Universitat de Barcelona

Introduction to Machine Learning

Work 4: SVM

Authors: Emer Rodriguez Formisano, Georgina Ballbè i Serra, Jorge Alexander
Supervisor: Maria Salamó
Date: 7th January 2017

Introduction

The Support Vector Machine (SVM) classifier, is a supervised learning algorithm that generates a hyperplane between labelled data points, maximizing the average distance from the data points at the edge of the set (the support vectors), in order to classify future data points depending on where they lie in relation to the previously generated hyperplane.

In this work we will study its performance on: a linearly separable problem; a nonlinearly separable problem, which can be addressed using a linear kernel with soft margin; and a nonlinearly separable problem, which requires using nonlinear kernels.

The analysis of the performance of the algorithm consists of two parts. The first part with randomly generated datasets that meet the three cases described, and the second part using different configurations of the SVM algorithm applied to two datasets from the UCI repository[1], the Hepatitis and Pen-based ones.

The empirical results are exposed in the results section together with the main conclusions.

In all cases the SVC implementation from sklearn python library is used.

Exercise 1

The Python programming language was used for the implementation in file: **`exercise1_svm.py`**.

In this exercise we evaluate the performance of SVM in three cases:

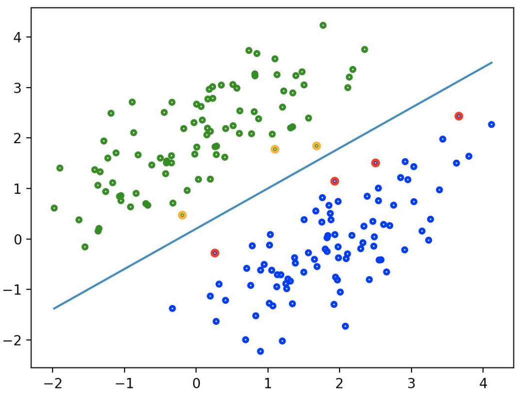
Linearly separable data (dataset 1)

In this section, SVC is set up to work with a linearly separable problem. Therefore a linear Kernel is used [SVC(kernel='linear')].

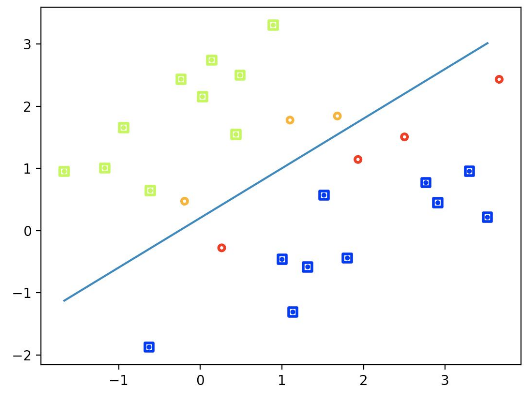
Function `generate_data_set1()` generates a datasets of two classes which are linearly separable.

In order to test the performance of SVM with this kind of datasets, we build and run the following function `run_svm_dataset1()`.
:

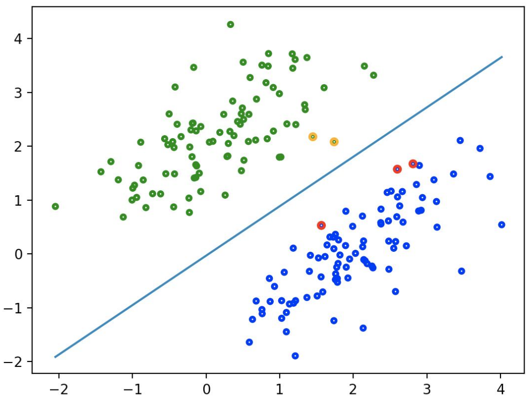
The following figures contain the results of two runs as example. Left column figures show the training data, the selected support vectors for each class and the hyperplane. On the right column figures show the support vectors, and the test data points with the classified class (color of the circle) and the ground truth of that observation (color of the square), for both examples all observations are correctly classified.



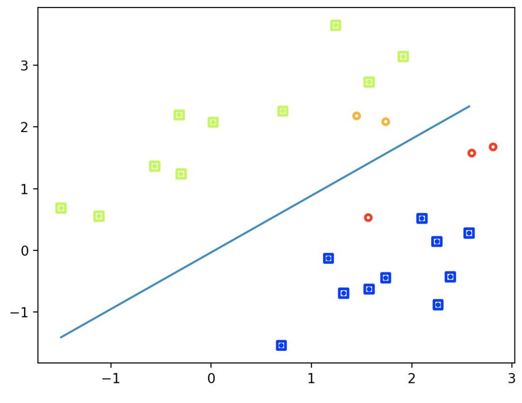
Example 1. Training data with two classes (green, blue) and selected support vectors for each class (yellow, red)



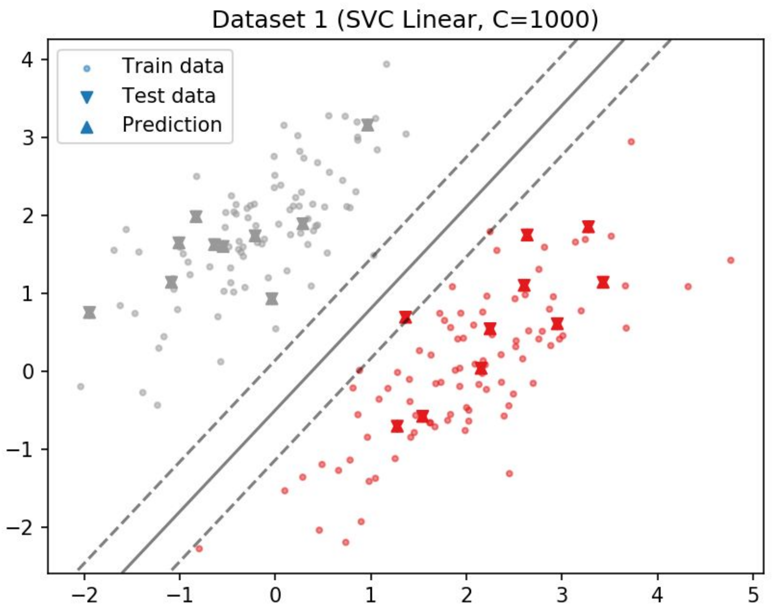
Example 1. Support vectors for each class (yellow, red)
Test data prediction: green, blue outer square
Test data ground truth: green, blue inner circle



Example 2. Training data with two classes (green, blue) and selected support vectors for each class (yellow, red)



Example 2. Support vectors for each class (yellow, red)
Test data prediction: green, blue outer square
Test data ground truth: green, blue inner circle



The following figure shows another run example of Dataset 1 with a high C value which aims at classifying all training examples correctly by giving the model freedom to select more samples as support vectors. The figure also show the test data and its predicted values.

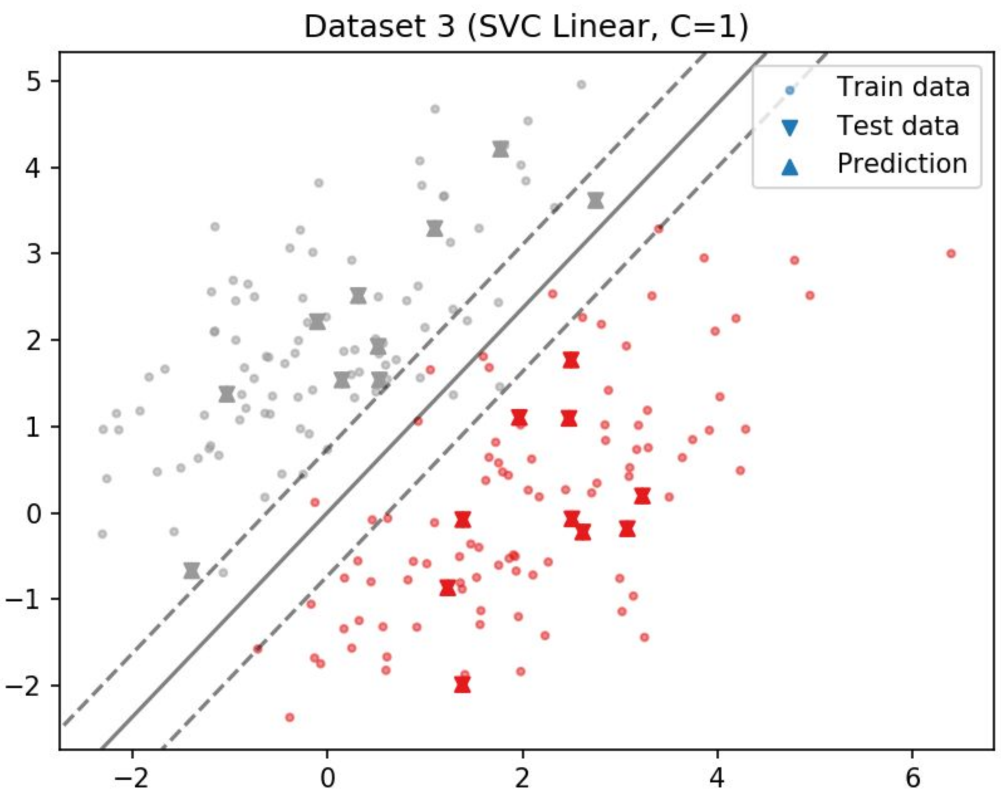
Console Output Statistics

```
correct prediction: 20
incorrect prediction: 0
total prediction: 20
score: 1.0
```

Nonlinearly separable data, soft margin (dataset 3) - Linear kernel with error margin

The second example presented is the analysis of dataset 3 which has some overlapping data points in the area between the two groups. A SVC with linear kernel and a low C parameter was used so that the hyperplane fitting is more flexible.

The following figure shows the data points and the hyperplane drawn. The split leaves instances of the two classes in both sides. However, as the test data tend to be outside the margin area, the predictions are correct giving a high accuracy value as shown in the console output statistics.

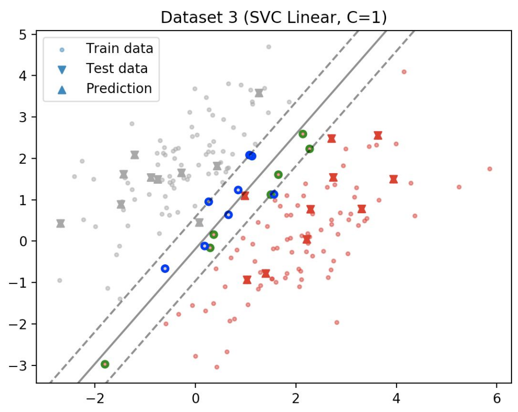


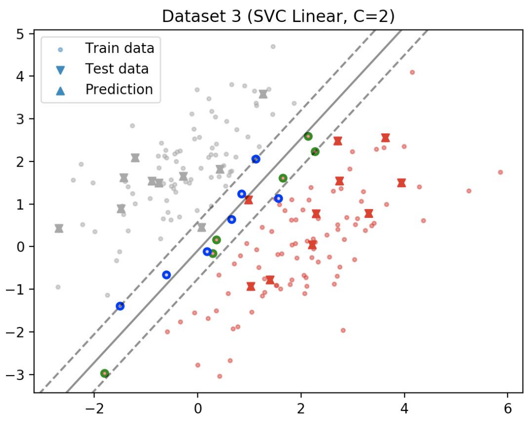
Console Output Statistics

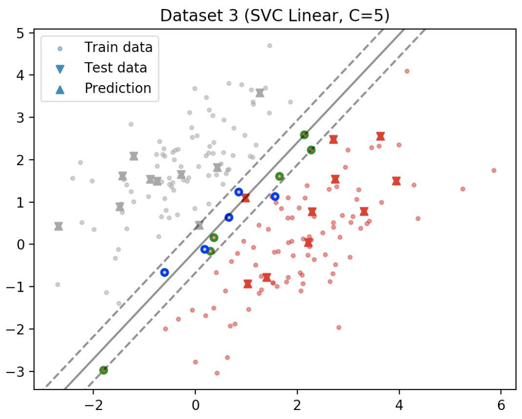
```
correct prediction: 20
incorrect prediction: 0
total prediction: 20
score: 1.0
```

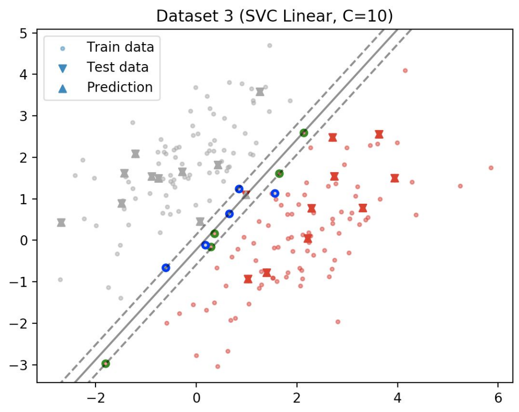
If the C is modified, the restriction on the error allowed during the calculation of the hyperplane varies. The following figure shows the effects of this variation. As C increases, the model tries to fit the training set with smaller error, leading to overfitting and worse result in accuracy. Note that the hyperplane margin area is reduced, the selected SV are different and the total number of SV decreases. Support Vectors are represented with blue and green circles for each class.

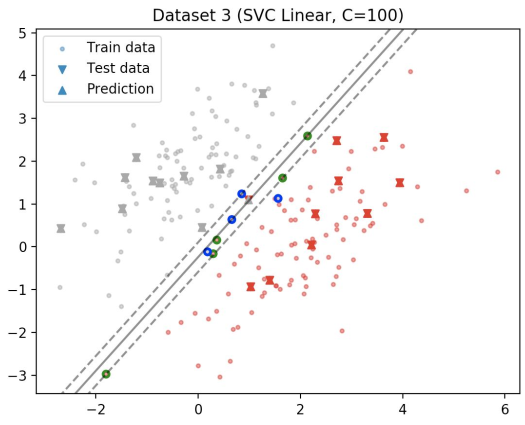
C	1	2	5	10	100	1000
Accuracy	1	1	1	0.95	0.95	0.95
# SV	14	13	11	10	9	9

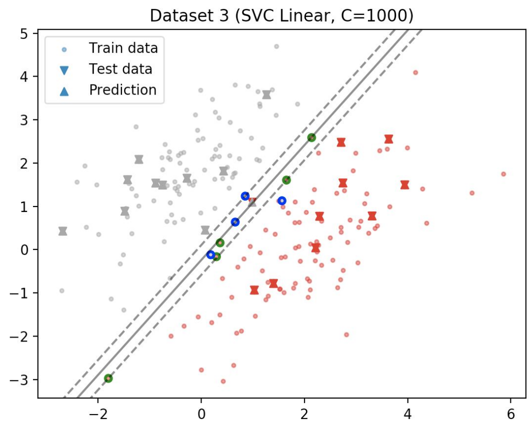








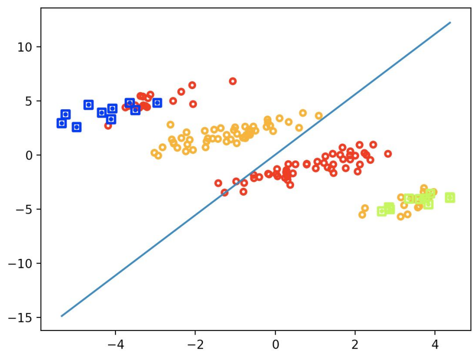




Nonlinearly separable data (dataset 2) - Gaussian kernel

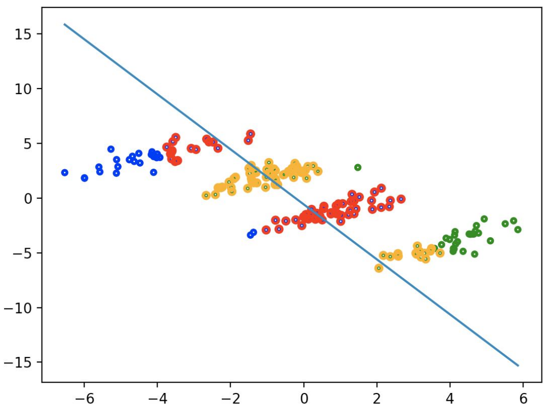
In third and last case presented, the data is not linearly separable. The following figures show the observations distribution in XY, the support vectors and the hyperplane obtained using a linear kernel in the SVC. With this configuration, most of the variables in the dataset are selected as SV.

The first examples show the selected support vectors, hyperplane and the classification results for test data. This figures also shows that there is a great amount of training points selected as SV. When running the test, all samples are correctly classified, please not that the test samples are all located in the extremes, this is not representative.

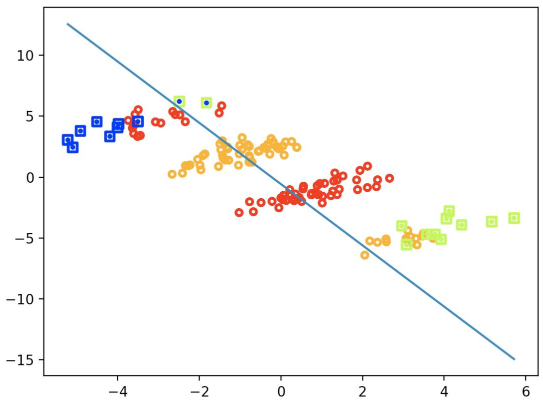


*Example 1.**Linear Kernel -** Support vectors for each class (yellow, red)*
Test data prediction: green, blue outer square
Test data ground truth: green, blue inner circle

The second example, even though the training data distribution is very similar to the previous one, the SVM calculates a completely different hyperplane, this also leads to consider that the linear kernel is not suitable. In this case, two test points are misclassified (green squares with blue inner circles). Accuracy score = 0.9.

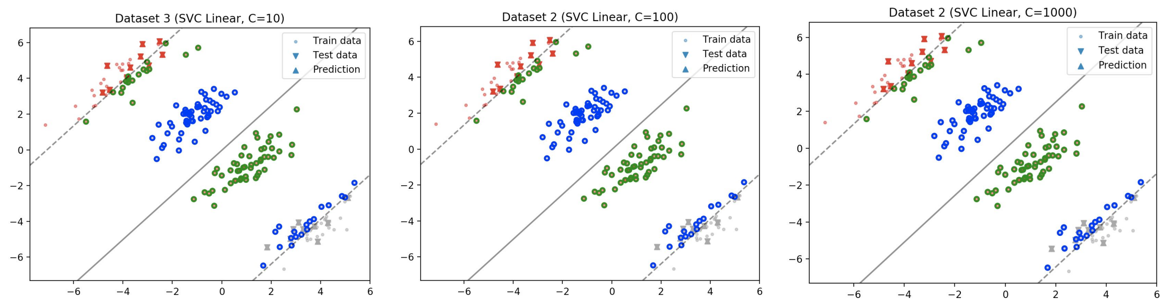


*Example 2. **Linear Kernel -** Training data with two classes (green, blue) and selected support vectors for each class (yellow, red)*

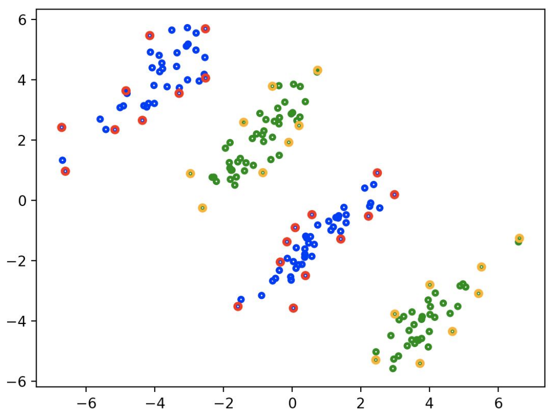


*Example 2. **Linear Kernel -** Support vectors for each class (yellow, red)*
Test data prediction: green, blue outer square
Test data ground truth: green, blue inner circle

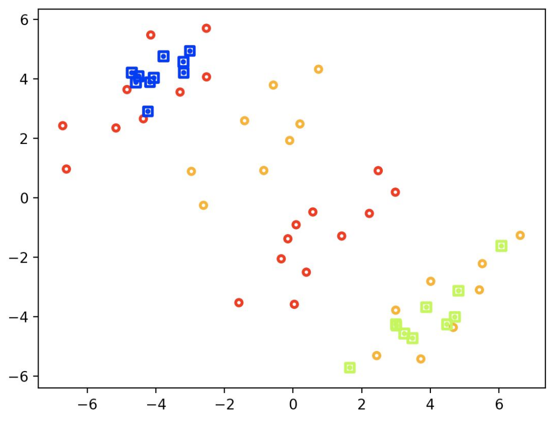
In the figures below show that even when changing the values of C, there is no variation in the hyperplane margin nor in the number and selection of support vectors (blue and green circles).



Changing the SVC linear kernel to a gaussian kernel [SVC(kernel='rbf')], then the number of SV decreases and the separation of the space is optimal, see red and yellow circles in the figures below, how only some of the ones in the outer perimeter are selected as SV to define the region.

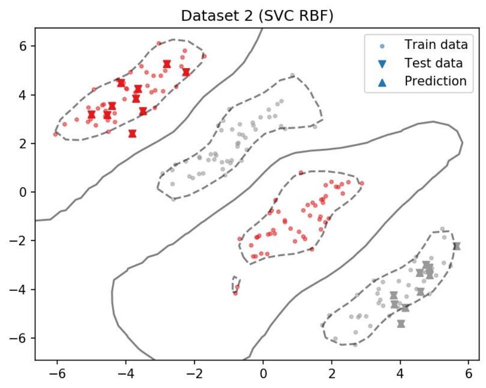


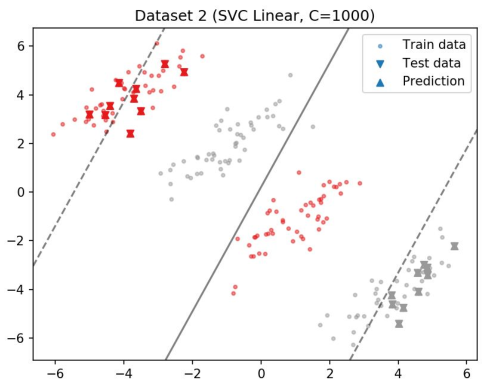
Example 3. ***Gaussian Kernel -*** *Training data with two classes (green, blue) and selected support vectors for each class (yellow, red)*



Example 3. ***Gaussian Kernel -*** *Support vectors for each class (yellow, red)*
Test data prediction: green, blue inner circle
Test data ground truth: green, blue outer square

A second comparison between SVC RBF and linear partitions is presented next. Note how the partitions provided by RBF are more robust than the one provided by the linear method. Both approaches have the same performance just because the test data is far away from the partition and it is not showing up the limitations of the linear method.





Console Output Statistics

```
correct prediction: 20
incorrect prediction: 0
total prediction: 20
score: 1.0
```

Exercise 2

Datasets

The analysis was performed using two datasets individually, without mixing the results. The intention of the analysis was to find good hyperparameters and test their performances with a given dataset, rather than finding robust hyperparameters across multiple datasets. The Hepatitis and Pen-based datasets from the UCI machine learning repository[1] were used and were provided having already been split into groups of 10 train and 10 test folds each.

Hepatitis

The Hepatitis data-set contains data regarding 155 patients who had the hepatitis disease. It includes several features such as age, sex,... (see the features list below). The feature to be classified is named “Class”, and represents whether that particular subject lived or died.

Features / dimensions

1. Class: DIE, LIVE
2. AGE: 10, 20, 30, 40, 50, 60, 70, 80
3. SEX: male, female
4. STEROID: no, yes
5. ANTIVIRALS: no, yes
6. FATIGUE: no, yes
7. MALAISE: no, yes
8. ANOREXIA: no, yes

9. LIVER BIG: no, yes
10. LIVER FIRM: no, yes
11. SPLEEN PALPABLE: no, yes
12. SPIDERS: no, yes
13. ASCITES: no, yes
14. VARICES: no, yes
15. BILIRUBIN: 0.39, 0.80, 1.20, 2.00, 3.00, 4.00 -- see the note below
16. ALK PHOSPHATE: 33, 80, 120, 160, 200, 250
17. SGOT: 13, 100, 200, 300, 400, 500,
18. ALBUMIN: 2.1, 3.0, 3.8, 4.5, 5.0, 6.0
19. PROTIME: 10, 20, 30, 40, 50, 60, 70, 80, 90
20. HISTOLOGY: no, yes

Training Size (Rows x Columns)
155 x 20

Pen-based

The pen-based dataset is a database of handwritten digits. The data contains 16 transformed coordinate values of a writing-tablet as writers wrote different individual digits on them. There is also one classification feature which is the digit written from 0 to 9.

Features
16 transformed coordinate values (between 0 to 100) plus one classification feature (0 to 9).

Training Size (Rows x Columns)
7494 x 17

Implementation

This implementation parser, normalization, cross-validation and statistical measurements as in previous knn work3.

For the main support vector machine algorithm, it uses the SVM module from Sklearn python library.

Performance evaluation

In order to evaluate the performance of the algorithm we calculate the accuracy for each fold, using the accuracy function available in sklearn.metrics module. It simply calculates the percentage of data points **correctly classified over the total classified**.

In order to decide whether the performance of a certain configuration of the algorithm was significantly better or worst than others, a **Friedman test was used**[2,4]. The Friedman test calculates the p-value across groups of results, which can then be tested for rejecting the null hypothesis (there is statistical differences among the mean accuracies of each group).

The **significance level** for the hypothesis rejection was set to **0.05**. In case the null hypothesis is rejected, the **Nemenyi test**[2] was performed which checks the differences pairwise.

Results & Discussion

The section contains the empirical results obtained from running the algorithms on different datasets. The first part of the analysis is focused on finding good hyperparameters over a wide range of settings, mainly kernels functions and C values.

Once a good candidate is identified, then the second part tries to refine or tune the parameters of the selected method. For example, if a RBF method is selected, the second part tries to find a best C and Gamma parameters that are robust across the folds.

The default parameters used for the first part of the analysis are described on Tab.1

Tab.1 - Hepatitis & Pen-based parameters of the Part I analysis

Parameter	**Values**
Significance level (alpha)	0.05
Kernel	Linear, Gaussian, Sigmoid, Poly
C	1, 10, 100, 1000

Hepatitis

Part I

The table 2 shows the output obtained from running the analysis with the default parameters on the Hepatitis dataset. The table is followed with the boxplot representation of the accuracies (Fig.1).

Tab 2. Mean and standard deviation of the accuracies

kernel	linear	linear	linear	linear	poly	poly	poly	poly	rbf	rbf	rbf	rbf	sig	sig	sig	sig
C	1	10	100	1000	1	10	100	1000	1	10	100	1000	1	10	100	1000
mean	0.822	0.819	0.756	0.743	0.793	0.832	0.807	0.799	0.793	0.822	0.785	0.656	0.793	0.822	0.813	0.718
std	0.098	0.095	0.100	0.100	0.027	0.087	0.095	0.099	0.027	0.098	0.125	0.122	0.027	0.103	0.106	0.121

The figure shows some interesting patterns. The performance of the SVC when using different kernels is similar. When the C parameter increases, the boxplots describe an arc, which maximum is observed using a C value of 10. After that point, there is tendency of decreasing performance when the C value increases. However, polynomial method is less sensitive to C changes compared to RBF method.

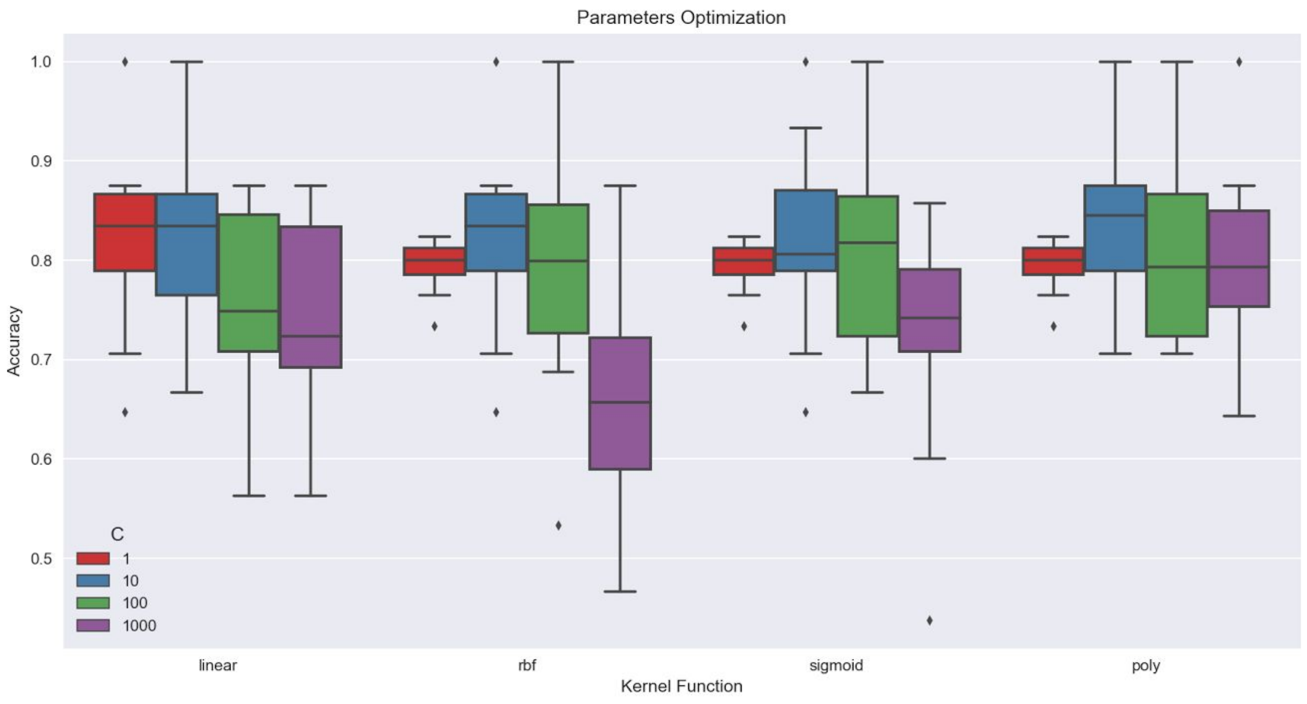
The Friedman test shows statistical differences among them with a p-value of 0.00836. Thus a post hoc Nemenyi test was used in order to show the pairwise comparisons among the cases. The statistically relevant p-values are presented next:

```
Statistically relevant Nemenyi p-values
rbf-1000  linear-1     0.043
          linear-10    0.043
          poly-10      0.015
          rbf-10       0.043
```

As initially spotted on the boxplot, the RBF with C value of 1000 is the worst performant combination of settings compared to many others. Then the rest of methods can be considered statistically equivalent.

The final selected model in phase I was RBF with C value of 10. The main reason is to study the sensitivity of gamma in the analysis and RBF tends to captures better nonlinear relationships.

Fig.1 - Hepatitis part 1 box plots



Part II

As commented in the previous section, RBF with C value of 10 was selected. The aim of the second phase of the analysis is to tune the settings in order to obtain better accuracy. The parameters used in the second round are summarised in Tab. 3.

Tab.3 - Hepatitis parameters of the part II analysis

Parameter	**Values**
Significance level (alpha)	0.05
Kernel	Gaussian
C	1, 5, 15, 30, 60, 75, 90
Gamma	0.01, 0.05, 0.1, 0.15, 0.2

The results obtained from the runs are detailed in Tab. 4. Its visual representation using boxplot are presented in Figure 2.

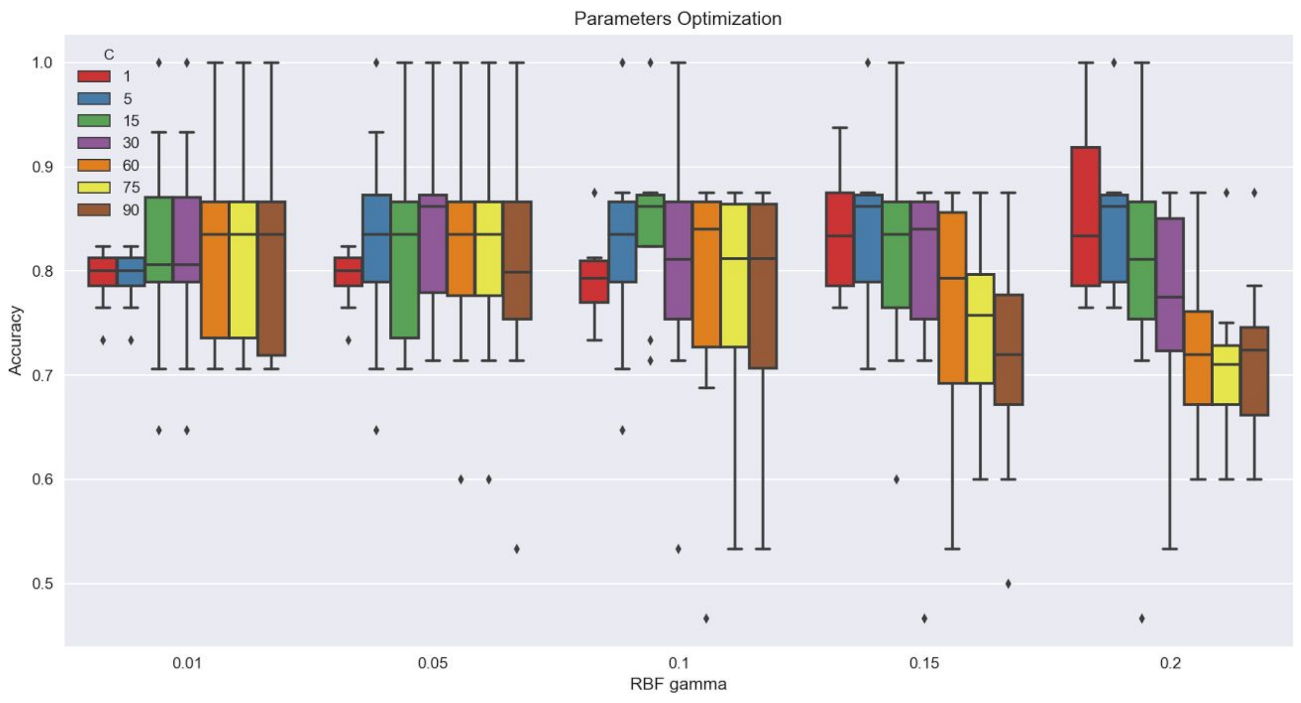
Tab.4 - Mean and standard deviation of the accuracies obtained with parameters of Part II

gamma	**0.01**	**0.01**	**0.01**	**0.01**	**0.01**	**0.01**	**0.01**	**0.05**	**0.05**	**0.05**	**0.05**	**0.05**	**0.05**	**0.05**	**0.1**	**0.1**
C	**1**	**5**	**15**	**30**	**60**	**75**	**90**	**1**	**5**	**15**	**30**	**60**	**75**	**90**	**1**	**5**
mean	0.793	0.793	0.822	0.822	0.820	0.820	0.814	0.793	0.828	0.820	0.838	0.818	0.818	0.798	0.793	0.822
std	0.027	0.027	0.103	0.103	0.094	0.094	0.098	0.027	0.103	0.094	0.084	0.108	0.108	0.124	0.038	0.098
gamma	**0.1**	**0.1**	**0.1**	**0.1**	**0.1**	**0.15**	**0.15**	**0.15**	**0.15**	**0.15**	**0.15**	**0.15**	**0.2**	**0.2**	**0.2**	**0.2**
C	**15**	**30**	**60**	**75**	**90**	**1**	**5**	**15**	**30**	**60**	**75**	**90**	**1**	**5**	**15**	**30**
mean	0.844	0.799	0.780	0.780	0.773	0.839	0.834	0.812	0.786	0.760	0.741	0.713	0.851	0.846	0.793	0.767
std	0.080	0.125	0.130	0.109	0.116	0.067	0.088	0.109	0.127	0.114	0.082	0.109	0.084	0.071	0.142	0.103
gamma	**0.2**	**0.2**	**0.2**													
C	**60**	**75**	**90**													
mean	0.721	0.709	0.717													
std	0.079	0.073	0.081													

Best selection of parameters are RBF with values gamma 0.1 and C 15. The boxplot shows lower IQR with high median accuracy.

We expected to see a positive effect with Low gamma and high C values (restrictive settings) but the results shows the accuracy is not that sensitive to the mentioned settings with an average values of 0.82. However, the variance is slightly high, by showing a large IQR.

Fig.2 - Hepatitis part 2 box plots



As the Friedman test was rejecting the null hypothesis with p-value of 1.4e-07, a post hoc analysis was done too. The Nemenyi test shows statistical differences between C parameters when gamma is fixed at 0.2. Those are evidences that changes in the C parameter results in a negative effect when they are combined with high gamma values. The same as the arc pattern described previously.

```
Statistically relevant Nemenyi p-values
0.2-75  0.2-1    0.049
        0.2-5    0.039
```

Pen-based

Part I - Finding good hyperparameters

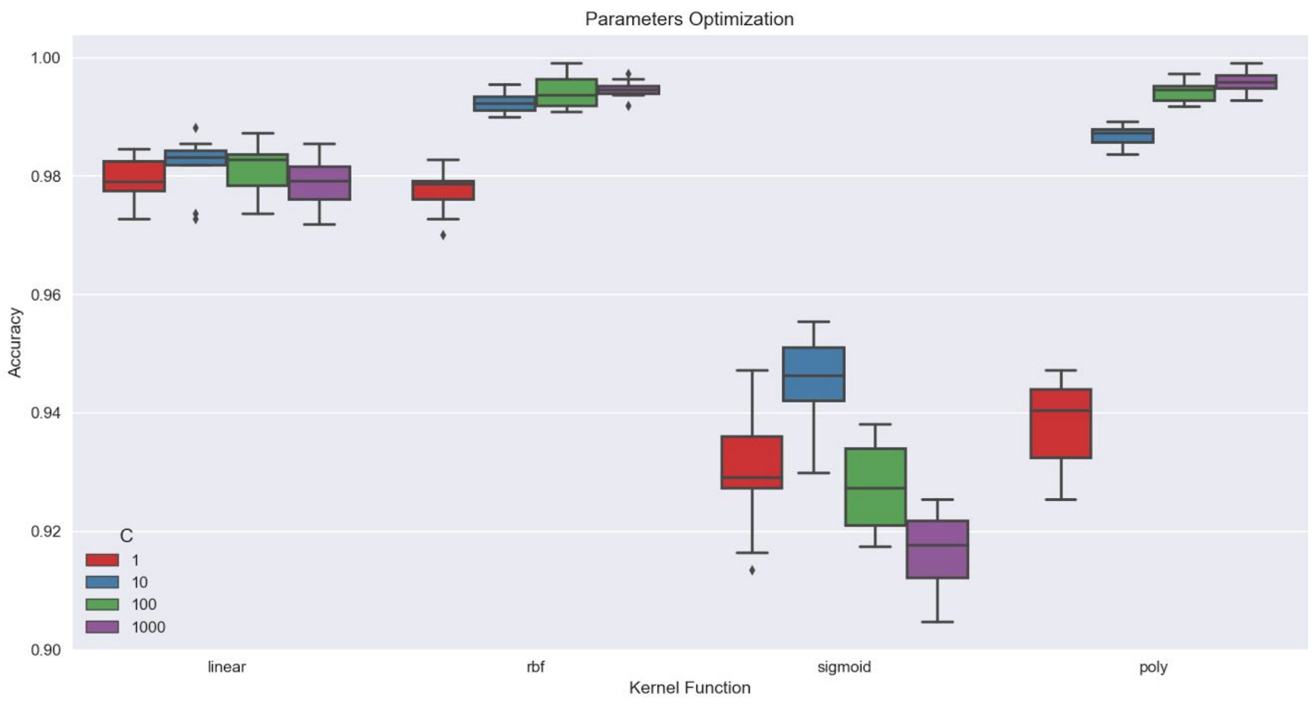
Table 5 shows the output obtained from running the SVC algorithm with Pen-based dataset with four different types of kernels (linear, gaussian, sigmoid and polynomial) and different values of C (1-1000) [see Table 1]. The rest of the parameters are the default ones. Figure 3 shows a boxplot representation of the accuracies.

Results

Tab.5 - Mean and standard deviation of the accuracies

kernel	linear	linear	linear	linear	poly	poly	poly	poly	rbf	rbf	rbf	rbf	sig	sig	sig	sig
C	1	10	100	1000	1	10	100	1000	1	10	100	1000	1	10	100	1000
mean	0.979	0.982	0.981	0.979	0.938	0.987	0.994	0.996	0.978	0.992	0.994	0.995	0.930	0.946	0.927	0.916
std	0.004	0.005	0.004	0.004	0.008	0.002	0.002	0.002	0.004	0.002	0.003	0.002	0.011	0.008	0.007	0.007

Fig.3 - Pen-based part 1 box plots



From Tab.6 and Fig.4, it is observed that the **rbf kernel** produces the highest overall accuracies and that a higher C has a better overall performance. In addition, sigmoid kernel shows a much worse performance, therefore it is not considered for the next steps of the statistical significance analysis (Nemenyi). The most relevant cases for the rest of the kernels are:

```
Statistically relevant Nemenyi p-values
poly-10       poly-1         0.043
poly-100      poly-1         0.000
poly-1000     linear-1       0.015
              linear-1000    0.020
              poly-1         0.000
rbf-1         poly-100       0.028
              poly-1000      0.006
rbf-10        poly-1         0.001
rbf-100       poly-1         0.000
              rbf-1          0.039
rbf-1000      poly-1         0.000
              rbf-1          0.033
```

When comparing the **rbf kernel** to the **poly kernel**, it has a slightly lower accuracy for C=1000 (0.994 vs 0.996) that is not a statistically significant difference. However, the nemenyi test results show that for rbf C=1 and poly C=1, the accuracy difference is statistically different and in favour of the rbf kernel. Therefore the **rbf (gaussian) kernel** was selected for further tuning.

Part II - tuning hyper parameters

On the second part of this analysis the kernel type is set to **rbf**. Due to the similarity of performance with different values of C, in this second part, C range is augmented and gamma parameter is added, in order to study the effects of the combination of the two. Table 6 provides the information of the settings of the SVC.

Tab.6 - Pen-based parameters of the part II analysis

Parameter	**Values**
Significance level (alpha)	0.05
Kernel	Gaussian
C	10, 100, 500, 1000, 5000, 10000
Gamma	0.01, 0.05, 0.1, 0.2, 0.5, 1.0

Results

Table 7 shows the results of accuracy obtained for the different combinations of gamma and C values, and Figure 4 a boxplot representation of these results.

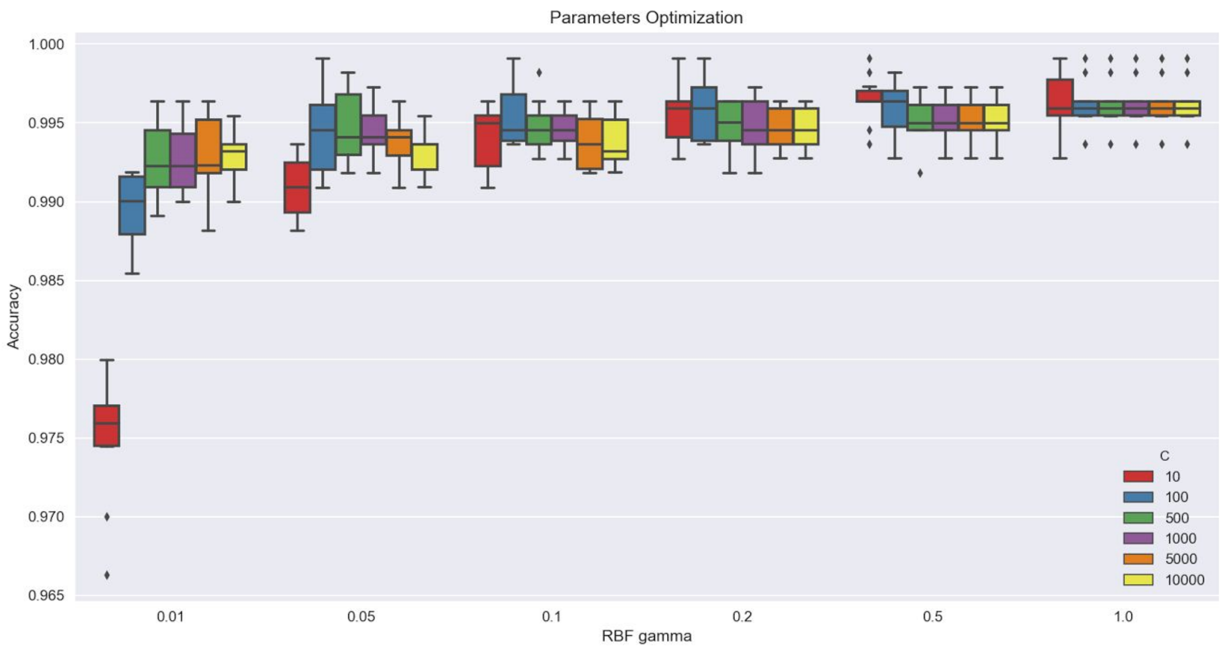
Tab.7 - Mean and standard deviation of the accuracies

gamma	0.01	0.01	0.01	0.01	0.01	0.01	0.05	0.05	0.05	0.05	0.05	0.05
C	10	100	500	1000	5000	10000	10	100	500	1000	5000	10000
mean	0.975	0.990	0.993	0.993	0.993	0.993	0.991	0.995	0.995	0.994	0.994	0.993
std	0.004	0.002	0.003	0.002	0.003	0.002	0.002	0.003	0.002	0.002	0.002	0.001

gamma	0.1	0.1	0.1	0.1	0.1	0.1	0.2	0.2	0.2	0.2	0.2	0.2
C	10	100	500	1000	5000	10000	10	100	500	1000	5000	10000
mean	0.994	0.996	0.995	0.995	0.994	0.994	0.996	0.996	0.995	0.995	0.995	0.995

gamma	0.5	0.5	0.5	0.5	0.5	0.5	1	1	1	1	1	1
C	10	100	500	1000	5000	10000	10	100	500	1000	5000	10000
mean	0.996	0.996	0.995	0.995	0.995	0.995	0.996	0.996	0.996	0.996	0.996	0.996
std	0.002	0.002	0.001	0.001	0.001	0.001	0.002	0.002	0.002	0.002	0.002	0.002

Fig.4 - Pen-based hyper-parameter tuning box-plots



It is observed from the results that for a high gamma there is no variation in the median accuracy, for the same range of C values. However, for lower values of Gamma, the C value has a greater effect over the accuracy. This is just the opposite pattern observed in the previous dataset. The lowest IQR with highest median accuracy is seen for gamma=0.5, and C=10, which is why it is chosen as the optimum parameter (highlighted in yellow in Fig.5).

Finally, looking at the statistical relevance of the performance with the selected parameters, the Nemenyi results shows that there is not statistical significance comparing to the results obtained with parameters from the first section (with C=1000 and gamma=1/number of features)
Furthermore, the models which have a statistical difference perform worse than the selected model because the difference of the medians is negative.

Statistically relevant Nemenyi p-values

0.5-10	0.01-10	0.000*
	0.01-100	0.000*
	0.01-500	0.023
	0.01-1000	0.002
	0.01-5000	0.008
	0.01-10000	0.001
	0.05-10	**0.000***
	0.05-10000	**0.009**

* 0.000 is a decimal number smaller than $0.5*10^{-4}$

However, although the best model does not have a statistically significant performance difference than other models, the higher median accuracy of the best model will mean that it performs better.

Conclusions

For the first exercise, using the first dataset it was found that the data was linearly separable, therefore a linear kernel suited dataset 1 well, and allowed a perfect prediction. In the second case, the data was not linearly separable, it was found that a perfect prediction could be found using a linear kernel with a correctly adjusted C, that is a linear kernel with soft margin. It is worth mentioning that a high C led to overfitting of the model to the data and lowered the accuracy of the test data prediction. Finally, the last dataset gave also nonlinearly separable data, but in this case a linear kernel with soft margin was not the a good solution either, even though the test data was found to be classified correctly by a linear kernel, it was only because the test data was located at the extremes. The best kernel for this dataset was a gaussian kernel, which allow a perfect separation of the classes.

In the second exercise, the analysis of the Hepa dataset concluded that a Gaussian RBF kernel with C=10 produced the best results, and that the Poly kernel was less sensitive to changes in C. However, the RBF kernel captured the nonlinear relationships better, and was more robust across both the Hepa and Pen-based dataset.

For the Pen-based dataset it was found that the Gaussian kernel with C=1000 produced the best results. It also had a similar performance as the Poly kernel, but performed better for low values of C, with statistical significance according to the Nemenyi tests. When tuning the hyper-parameters for the selected kernel, it was found that the higher gamma made the accuracy less sensitive to changes in C. The best result was judged to be when gamma=0.5 and C=10, which gave a mean=0.996, std=0.002.

Which is the best kernel function for a SVM classifier?

From the work done in exercise 1 and 2 we conclude that the best kernel depends on the distribution of the data of the problem. The datasets in Exercise 1 were easy to plot and allowed a better understanding of the shapes and margins of the hyperplane defined by the SVM. In Exercise 2, the best kernel was obtained by looking at the statistical analysis of the results obtained with each kernel, and for both datasets RBF provided best results for both cases.

Did you find differences in performance among the different kernel functions used at the SVM algorithm?

The SVM performance could show great variations in its performance depending on the kernel, for example, for both datasets in exercise two, sigmoid kernel failed to achieve accuracies similar to the other kernels. Also, for pen-based dataset, the difference in performance of the linear kernel and the gaussian kernel had statistical relevance.

According to the data sets chosen, in both exercises, which combination of hyper-parameters let you to improve at the maximum the accuracy of the SVM?

The combination of parameters chosen were described in detail in the results section. As a general overview, the default parameters set by the algorithms are a decent starting point. Default kernel is RBF with gamma in ‘auto’ (1/num. of features) and C value of 1. The only problem with them is that the C parameters had to be set to 10 in the Hepatitis and 1000 in the Pen-based dataset in order to return good results, far away from value of 1 used by default. As it is dataset dependent, it is hard to set a default value that works in all scenarios.

References

[1] Lichman, M. 2013. UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

[2] Janez Demšar. 2006. Statistical Comparisons of Classifiers over Multiple Data Sets. J. Mach. Learn. Res. 7 (December 2006), 1-30.

[3] R. Kohavi. 1995. A study of cross-validation and bootstrap for accuracy estimation and model selection. In Proceedings of the International Joint Conferences on Artificial Intelligence IJCAI-95.

[4] Wikipedia contributors. 2017. 'Friedman test', *Wikipedia, The Free Encyclopedia,* 24 November 2017, 11:13 UTC, <https://en.wikipedia.org/w/index.php?title=Friedman_test&oldid=811846519> [accessed 17 December 2017]

[5] SciKit documentation for sklearn svm. 2017. <http://scikit-learn.org/stable/modules/svm.html> [accessed 3rd January 2017]

[6] Pedregosa *et al.* 2011. Scikit-learn: Machine Learning in Python. JMLR 12, pp. 2825-2830.

Code execution

Exercise 1

Code can be executed runing “exercise1_svm.py”
Go to the bottom of the file to run different options. (comment / uncomment)

Exercise 2

Code can be executed running the “main.py”.

The following files contain all the necessary functions:

`main.py`: main script to obtain performance data of KNN algorithms. Calls all necessary functions implemented in the files mentioned below.
`plot_svm.py` : implements auxiliary functions to plot dataset data and results algorithm.
`utils.py` : implements auxiliary functions to the algorithm.
`nemenyi.py`: implements Nemenyi's multiple comparison test.
`Parser.py`: implements the parser of the dataset files.

Extra packages may need to be installed, such as `seaborn` for the plots