Exercises of SVM classifiers

Pablo García Fernández
(pablo.garcia.fernandez2@rai.usc.es)

Tuesday 18th January, 2022

Abstract

The aim of this report is to present the work carried out during the execution of the exercises of the unit 7 assignment, as well as the obtained results. In addition to this report, we also attach the developed .py code, the obtained figures and a .txt file with the results. Note that we use the **scikit-learn implementation of SVC**, so the results may vary from those of LibSVM.

Exercise 2 and 3: Compare SVM with linear kernel and gaussian kernel using the whole dataset as training and test set.

As in previous reports the starter point are the hepatitis and wine datasets. Note that since exercise 1 and exercise 2 ask us to re-run the SVM classifier using the whole dataset as training and test set, but alternating between a linear and a Gaussian kernel, we have decided to integrate both exercises in the same section.

The goal of this point is to compare both kernels not only in terms of accuracy, but also in terms of performance (measuring the computational time cost for each case). To this end, the methodology is as follows. We first standardize the data using the mean and standard deviation of all samples. Next, we train an SVM clasiffier on all the data with a linear kernel (no regularization) and with a Gaussian kernel (fixing $\lambda = 100$ and $\sigma = 1/no.\ of\ features$). Finally, the obtained models are tested on the entire dataset.

"Tab. 1" shows the obtained metrics and "Fig. 1", the obtained confusion matrices. We can observe that for this problem, where the number of patterns is much larger than the number of attributes, the projection of the Gaussian kernel to a higher dimensional hidden space increases the linear separability between classes and, consequently, better results are obtained than with the linear kernel. If the number of attributes were very high, we would see how this difference is reversed (since the linear separability is already very high in the input space). Regarding the computational cost, in the last column of the tables we can see how Gaussian computational cost is higher (more time consuming).

Finally, given that higher kappa values are reached in wine than in hepatitis, we can conclude that the classes comprising the hepatitis dataset are less linearly separable.

Recall that, as already stated in the LDA report, the results obtained do not constitute a good evaluation of the model, they are positively biased by training and testing on the same sets. A better evaluation is performed in the following section.

Table 1: Kernel comparison. Whole dataset as training and test

Hepatitis dataset

Classifier	Metrics					
	acc. (%)	**kappa (%)**	**precision (%)**	**recall (%)**	**f1 (%)**	**Time (s)**
Linear	89.68	68.50	93.50	93.50	93.50	0.0040
Gaussian	100	100	100	100	100	0.0044

Wine dataset

Classifier	Metrics					
	acc. (%)	**kappa (%)**	**precision (%)**	**recall (%)**	**f1 (%)**	**Time (s)**
Linear	100	100	-	-	-	0.0018
Gaussian	100	100	-	-	-	0.0022

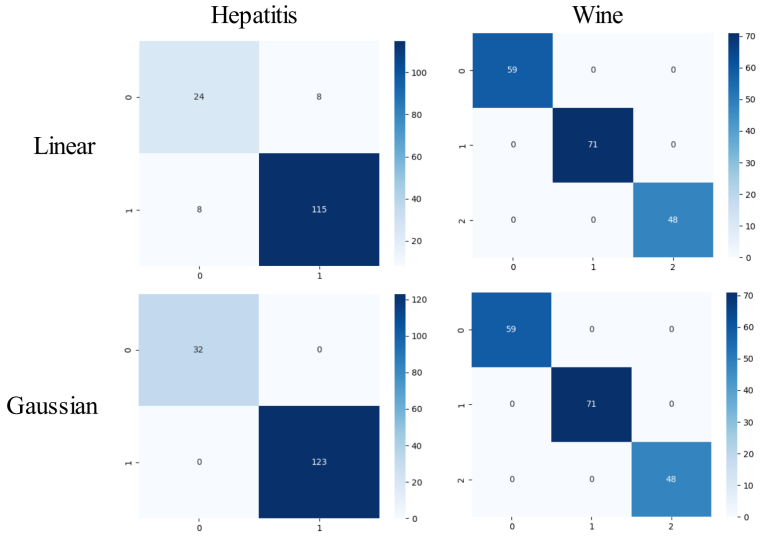

Figure 1: Kernel comparison. Whole dataset as training and test

Exercise 4: Repeat with 4,10-fold cross validation and parameter tuning

We start by using the function `crea_folds()` to obtain the partitions. Then we perform the tuning by training/testing over the train and validation partitions under different configurations using the K-folds cross-validation schema already explained in previous reports. Once the best hyperparameters are obtained, we re-train the model using both training and validation sets and evaluate its performance on the never-used test set (using again the K-folds cross validation approach). We repeat this process for both datasets and k=4,10.

First we present the results from **hepatitis** dataset:

- **Tuning step results**: "Fig. 2" (the detailed results can be seen in the attached .txt file).

– Linear k=4, best $\lambda = 0.125$ with 33.29% kappa.

– Linear k=10, best $\lambda = 8.0$ with 28.11% kappa.

– Gaussian k=4, best $\lambda = 8.0$, best $\sigma = 0.03125$ with 50.63% kappa.

– Gaussian k=4, best $\lambda = 32.0$, best $\sigma = 0.00781$ with 45.40% kappa.

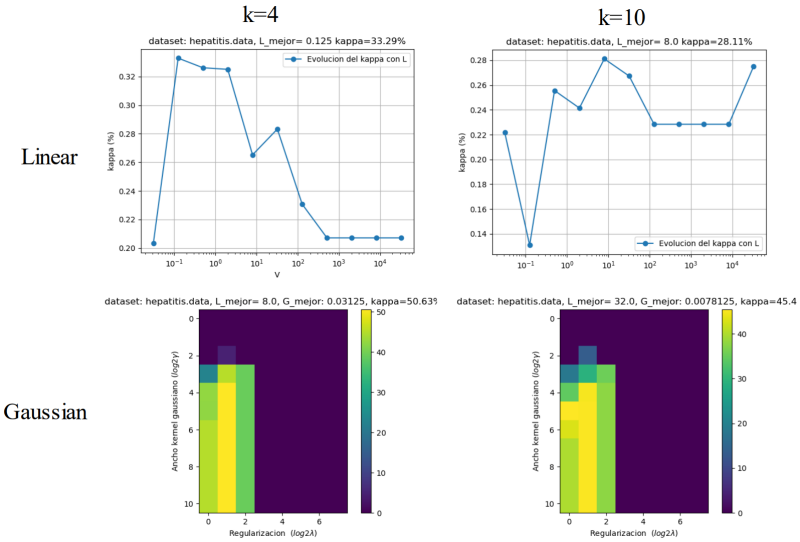

Figure 2: Tuning step in hepatitis dataset

- **Test results** with the tuned parameters: "Tab. 2" and "Fig. 3"

Table 2: Hepatitis test results after tuning

Hepatitis dataset

Method	**Metrics**				
	acc. (%)	**kappa (%)**	**precision (%)**	**recall (%)**	**f1 (%)**
Linear 4-folds	79.27	28.28	86.47	88.64	87.29
Linear 10-folds	80.50	47.99	87.92	86.67	86.81
Gaussian 4-folds	85.37	54.08	91.99	90.15	90.73
Gaussian 10-folds	80.50	46.84	86.49	88.00	87.08

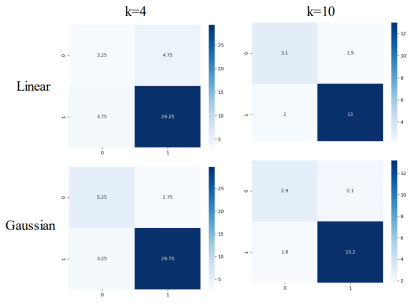

Figure 3: Hepatitis test results after tuning. Confusion matrices

We also present the results from **wine** dataset:

- **Tuning step results**: "Fig. 4" (the detailed results can be seen in the attached .txt).
  - Linear k=4, best $\lambda = 0.5$ with 95.63% kappa.
  - Linear k=10, best $\lambda = 0.03125$ with 98.14% kappa.
  - Gaussian k=4, best $\lambda = 2.0$, best $\sigma = 0.00781$ with 97.37% kappa.
  - Gaussian k=4, best $\lambda = 0.5$, best $\sigma = 0.03125$ with 98.08% kappa.

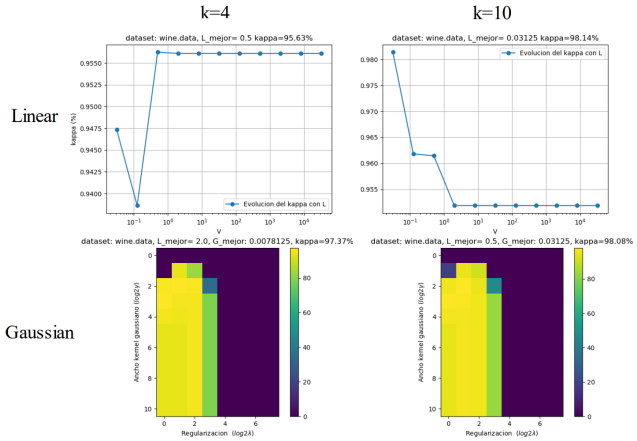

Figure 4: Tuning step in wine dataset

- **Test results** with the tuned parameters: "Tab. 3" and "Fig. 5"

Table 3: Wine test results after tuning

Hepatitis dataset

Method	**Metrics**	
	acc. (%)	**kappa (%)**
Linear 4-folds	96.94	95.30
Linear 10-folds	99.41	99.09
Gaussian 4-folds	98.47	97.67
Gaussian 10-folds	99.12	98.65

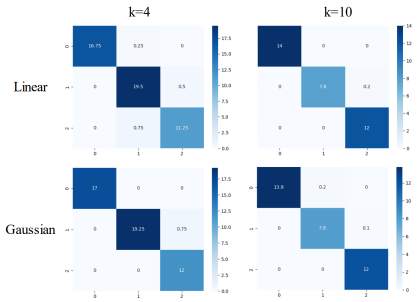

Figure 5: Wine test results after tuning. Confusion matrices

Exercise 5: Repeat for LBP and CooCur with OVO and OVA approaches

The methodology followed is the same as in the previous exercise, but using CooCur and LBP as datasets. As these are already divided into training/test splits, the first step is to unify them and perform the separation with the function `crea_folds()`. The following steps are exactly the same. Just to point out that in this exercise we are asked to evaluate the performance of the Gaussian kernel with the OVA (where as many SVCs are created as classes and each one learns to separate between one class and all the others) and OVO (where $C(C-1)/2$ classifiers are created and each one learns to separate between two different classes). To do this, we must measure both the accuracy values and the time spent by each.

Finally, it is worth mentioning that all evaluations are performed with a 4-folds cross-validation and tuning the hyperparameters of the models. All figures and detailed results can be found in the attached files (*4-svm/LBP_CooCur/*). Summarizing, "Fig. 6" shows the results of the tuning step. Table "Tab. 4" and "Fig. 7" show the test results.

As expected, the SVC OVO Gaussian is the most time-consuming approach. It is curious that, in terms of classification quality, the results are slightly better with the OVA approach than with the OVO approach. In addition, as always, better results are achieved in LBP textures.

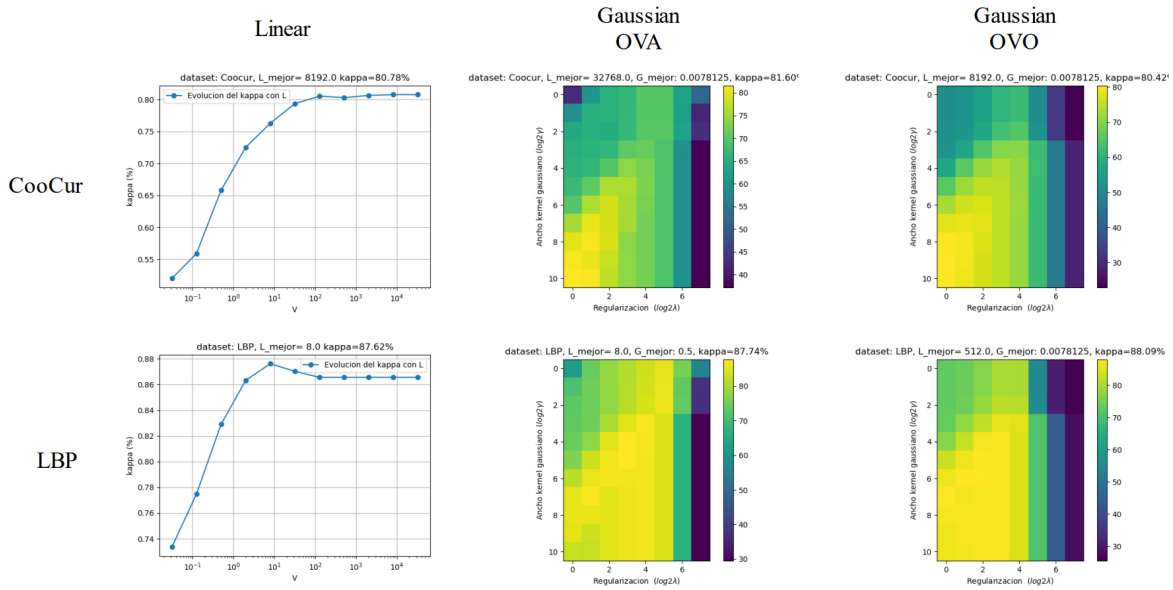

Figure 6: Tuning step in both datasets.

Table 4: Test results over CooCurr and LBP. 4-folds cross validation with tuned hyperparameters.

CooCur dataset

Classifier	**Results**		
	acc. (%)	**kappa (%)**	**Time (s)**
SVC linear	85.76	85.50	0.1452
SVC gaussian OVA	86.46	86.20	0.2477
SVC gaussian OVO	84.84	84.55	2.0641

LBP dataset

Classifier	**Results**		
	acc. (%)	**kappa (%)**	**Time (s)**
SVC linear	89.93	89.74	0.0601
SVC gaussian OVA	90.62	90.45	0.2208
SVC gaussian OVO	89.70	89.50	2.0758

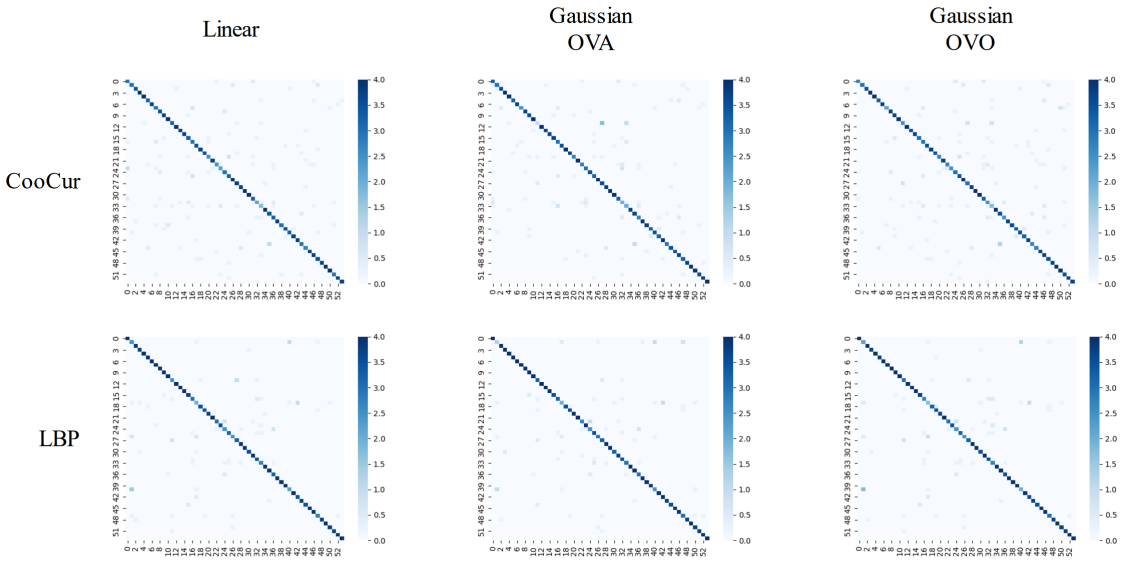

Figure 7: Test results, confusion matrices.