Paolo Alberto
Student ID: s278098

Machine Learning and Deep Learning
-
Politecnico di Torino

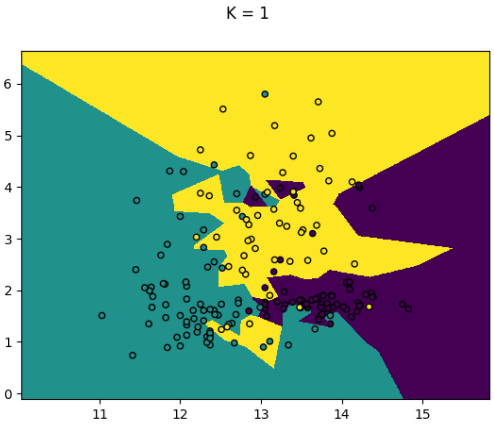
Homework 1 report

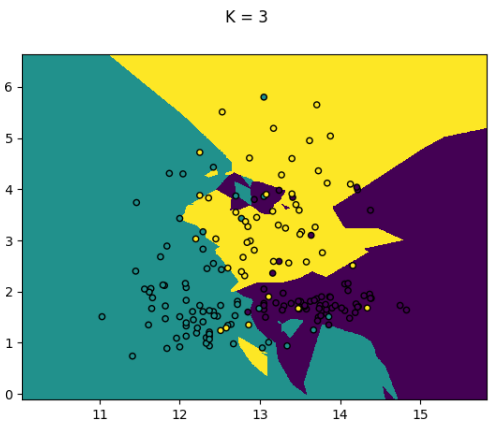
The purpose of homework 1 is to perform some analysis on the Wine dataset: a small collection of 178 instances, belonging to three different wine categories. Each entry has 13 features representing the different chemical constituents of the sample. In the following analysis, we will focus only on the first 2 attributes of the dataset (alcohol and malic acid), in order to have a 2D visual representation of the features. In order to make the experiment repeatable and consistent, the same seed (42) was used for all the random state variables in this homework.

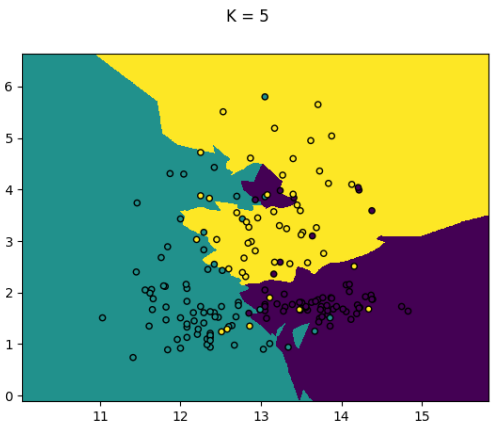
1 K-Nearest-Neighbors

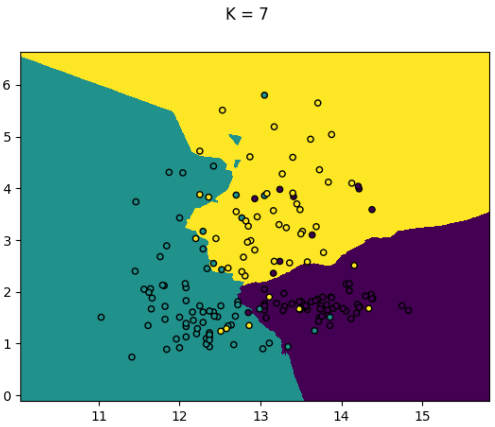
In the first step, the KNeighborsClassifier class from the sklearn.neighbors module is implemented for the 4 different values of K proposed. These models are used to plot the decision boundaries, by predicting the class for every pixel in a grid, with an ad hoc function.
As we can see from these plots, a smaller value of K creates more noise, while a larger value gives smoother decision boundaries. This is due to the fact that several training points contribute to the classification decision. On the other hand, a larger value of K is more computationally expensive, so the choice of K can be a trade-off.
The choice of K could also be seen from the point of view of the Bias-Variance tradeoff: an high value of K increases the Bias, while decreasing the variance and viceversa.

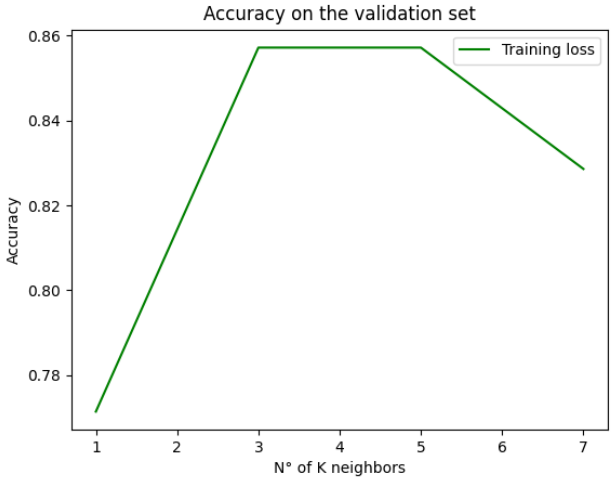
To show why the boundaries are noisier with a low K, let's consider now the case of K=1. In this case each training data point defines a region around it. All the test points in this region will be assigned to the same class as the training point. It means that boundaries will be given by the perpendicular bisector of the line segment connecting two points. This kind of partition leads to the so called Voronoi Diagram, which is responsible for the irregular shapes.


K = 1


K = 3


K = 5


K = 7


Accuracy on the validation set
Training loss
Accuracy
N° of K neighbors

Neighbors	K = 1	K = 3	K = 5	K = 7
Accuracy	0.771	0.886	0.8	0.829

The best accuracy on the validation was found to be 0.886, with K = 3. Evaluating the best performing model on the test set gives us an accuracy of 0.815. The model seems to be working fine on the test set. This kind of value can be considered a good result, taking also into account the fact that we are training on the first 2 features of the dataset, without any domain knowledge.

If we try to increase the number of features, KNN actually performs worse. This is caused by the fact that this algorithm relies on the calculation of the distance between points, and this metric becomes less meaningful when increasing dimensions.

2 Linear SVM

The first step is the normalization of the features. After this pre processing step, a model is defined for each of the different values of the parameter proposed. The classifier used here is the LinearSVC from the sklearn library. We use soft margin classification, adding slack variables ξ_i to allow for some outliers in our data. The minimization problem is:

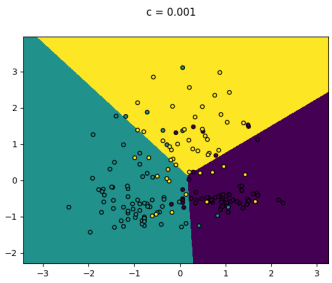
$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{m}\xi_i \\ \text{subject to} \quad & y_i[\langle \mathbf{w}, \mathbf{x}_i\rangle + b] \geq 1 - \xi_i, \quad \text{and} \quad \xi_i \geq 0; \quad (i = 1, \cdots, m) \end{aligned}$$

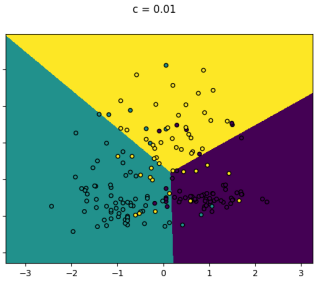
A smaller C increases the margin while ignoring the outliers in the training data. On the other hand, a larger value of this parameter creates a smaller margin, but it could possibly cause overfitting. When plotting the boundaries, the accuracy remains stable until $C = 100$, and the only relevant change can be seen with $C = 1000$, when the separation lines begin to worsen and the accuracy drops in a significant manner.

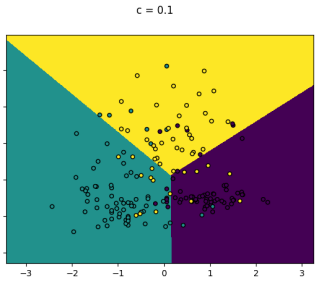
The more C increases, the more we are increasing the cost of misclassification (with C that tends to infinity we obtain an hard margin classifier), so we could argue that with such a big value, the classifier is too rigid to allow for the outliers, and for that reason it doesn't work well with this kind of non-linearly separable data. In particular, when faced with a multi class problem, the LinearSVC model applies a "one vs all" strategy for the boundaries decision, and this can lead to problems with such a rigid value for C.

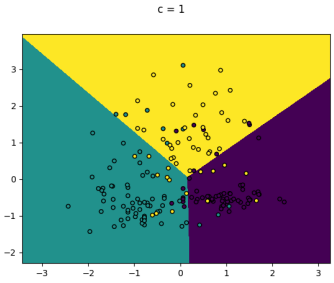
C	0.001	0.01	0.1	1	10	100	1000
Accuracy	0.8	0.8	0.828	0.828	0.828	0.828	0.6

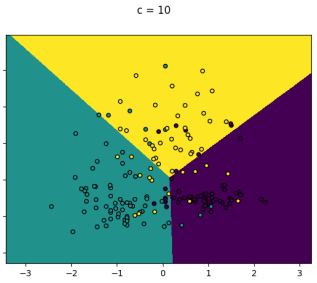
Evaluating the best performing model on the test set gives an accuracy of 0.76, wich is slightly worse with respect to the validation set.

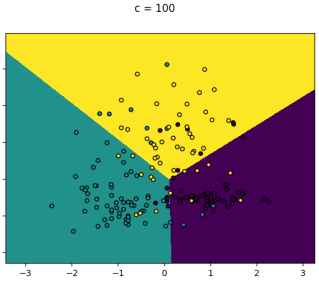

c = 0.001

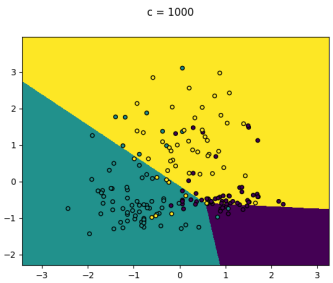

c = 0.01

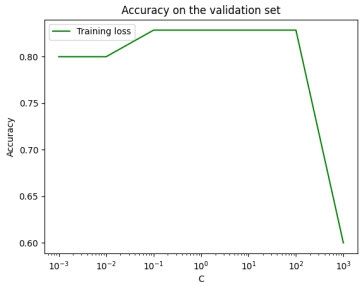

c = 0.1


c = 1


c = 10


c = 100

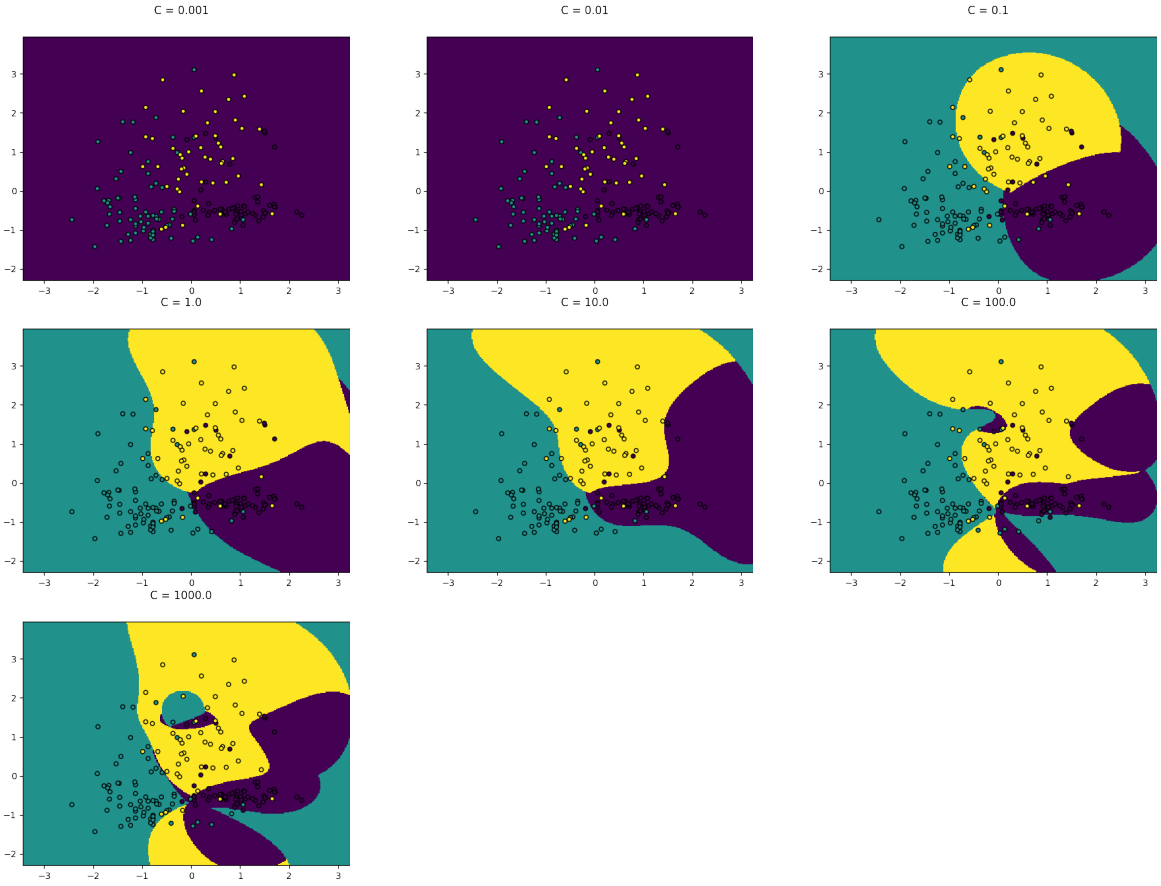

c = 1000


Accuracy on the validation set
Training loss
Accuracy
C

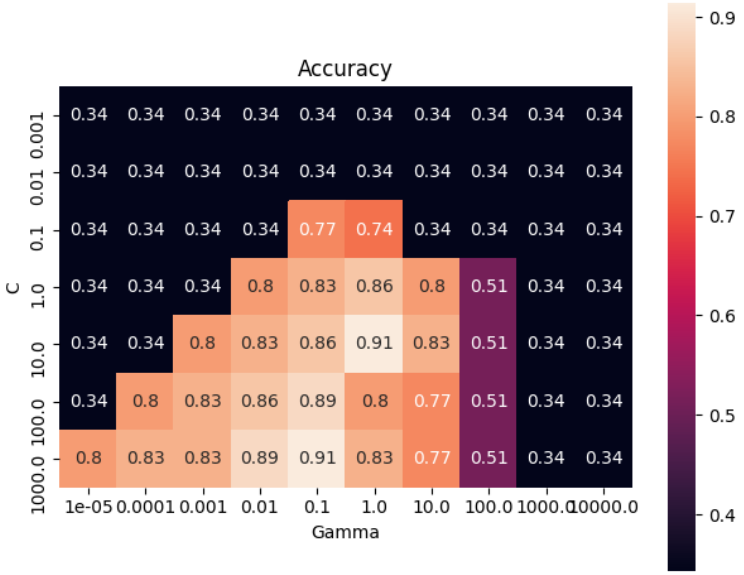
3 RBF Kernel

For this task, the SVC classifier from the sklearn.svm module was used. After repeating the same procedure done with the linear SVM classifier, we plot the boundaries for the different values of C, and the accuracy. The use of this algorithm gives an advantage on the linear SVM. This is pretty reasonable, since the provided features are not linearly separable. The boundaries obtained are different from the ones obtained with the linear SVM, and they reflect the non-linearity of this type of classifier.

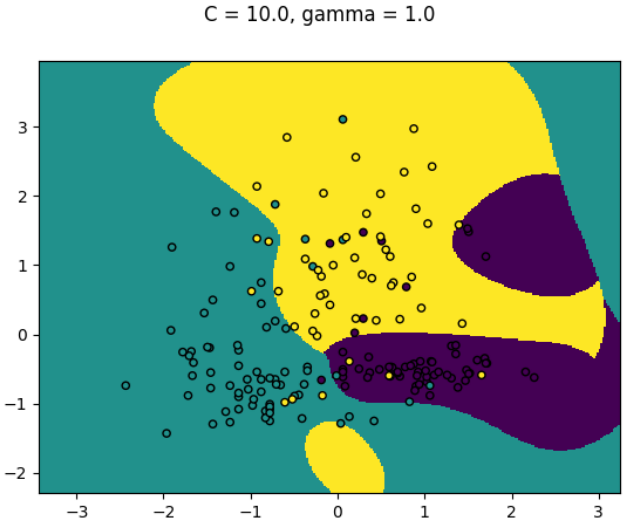
C	0.001	0.01	0.1	1	10	100	1000
Accuracy	0.342	0.342	0.828	0.885	0.914	0.8	0.8



Now we proceed to tune both gamma and C, with a grid search. The range for C is the same as before, while for gamma we used an array of 8 possible values, evenly spaced on a log scale from 10^{-5} to 10^{4}.



The best accuracy on the validation set was found to be 0.914, with $C = 10$ and gamma $= 1$. Evaluating these parameters on the test set gives an accuracy of 0.815. The decision boundaries for these parameters are then plotted:



4 K-Fold

After merging together training and validation splits, we make use of the GridSearchCV class provided by sklearn in order to perform a 5-fold validation on the training data.

The best performing parameters are in this case $C = 1000$ and gamma $= 0.01$, with an accuracy of 0.83. When evaluated on the test set, the accuracy becomes 0.778.

The final score on the validation is not so different from the accuracy that was found during the training phase. If we compare said difference with the one obtained using holdout in the previous section, we can see that there is less of a discrepancy between validation and test accuracies. This can be explained by the fact that the validation accuracy is calculated by averaging the accuracies obtained during the 5-fold validation procedure.

5 Conclusion

Various differences can be observed between KNN and SVM. The former is based on the calculation of distance between points, while the latter tries to find a separation between classes. Because of its distance based nature, the KNN algorithm has worse performances when we increase the dimensionality of the data. It is probably better to use it if we have lots of points in a low dimensional space. If instead, we have feature points in an high dimensional space, it is probably better to use an SVM. KNN will be better than a linear SVM if the data is not linear, but that is not true anymore if we use an RBF SVM instead.
Overall, SVM can be considered like an improved version of KNN in wich we only keep the points closer to the boundaries for classification (support vectors).
In conclusion, out of the three types of models proposed, the usage of a non-linear SVM would be the best solution, especially in a real life scenario, where we would use all of the 13 attributes available for this dataset.