Mandatory Assignment 01

Machine Learning F2023

# Exercise 1 (Moons dataset)

## Subtask A

When working with a Mulit-Layer Perceptron Classifier (MLPC) you have parameters such as hidden\_layers and alpha. Hidden\_layers specify the number of layers and nodes per layer that you would like to process your inputs. The alpha parameter specifies the regularization term, simply a value that changes the answer to be simpler. This can be used to prevent overfitting.

Specifically for the moons dataset, I found that using the parameters hidden\_layers = (50, 25, 10), alpha = 0.01 works great as the input data is not restricted to too few nodes, which could lead to underfitting, yet you don’t have too many to produce an output that is in the category of overfitting. The alpha parameter helps to prevent the output data from being overfitted, and thus working with a higher node count, a higher alpha value works great.

## Subtask B

Standard logistic regression using the ‘lbfgs’ solver does not fit the moons dataset very well. This is more or so due to the curvature of two categories. However, using a polynomial logistic regression would probably be able to fit our dataset, however I feel that is out of the scope of this task.

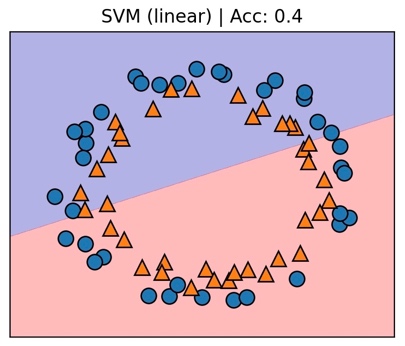
The exact same goes for k-means. As we are dealing with two categories, setting a cluster size of 2 gives an output where the edge-data for each group is categorized wrongfully. This is once again due to the coordinate intercept of the two categories.

Decision trees work a little bit differently. When having a max\_depth of 5, I feel you have a good balance between well fit and overfitted output. Overfitting is a big problem with decision trees and setting the max\_depth of the tree can help. The solution I found makes a good categorization about 50% of the time. The same goes when using a random forest classifier.

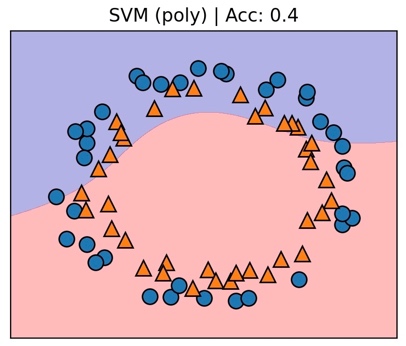
When using a support vector machine (SVM), I found that having the regularization parameter set to around 100, is plenty to output more normalized predictions, while it not being too “simple”. Using an SVM with the “RBF” kernel fits the moons dataset extremely well, while other kernels such as, “linear” or “sigmoid” don’t fit the shape. An RBF SVM is the most efficient and effective choice when categorizing the moons dataset.

## Subtask C

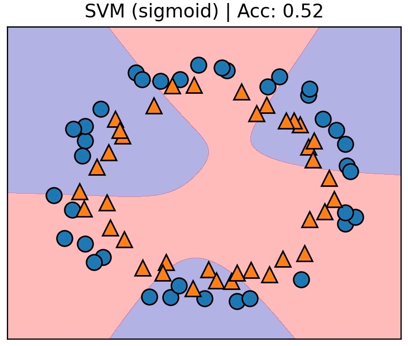
Looking at a couple different solutions using a Support Vector Machine Classifier (SVC) with various kernels, let’s start with the linear kernel.



The linear kernel tries to find a common line that separates our two categories in our dataset (circles). However, it’s clear to see that the linear kernel is not the best option for the circles-dataset. This is because our categories are mixed within each other and therefor basically impossible to find a straight line that separates the two. In this case, it does not matter what our other parameters for the SVC would be.

Another kernel we can use is the “poly” kernel. This is a kernel that tries to find a polynomial that separates our two categories.

Looking back at the linear kernel, this is a step up in terms of categorizing our dataset correctly, but there is still much to be desired. While it seems the poly kernel can pick up more regularization between our two categories, it still has an accuracy score of 40%, which is the same as our linear kernel.



Because it is challenging to have positive parameters, the sigmoid kernel is typically troublesome or invalid. As it has a significant flaw—namely, that its output value range is not centered on zero—the sigmoid function is no longer frequently utilized in research. Here we can see that the sigmoid kernel does exactly that, it has a hard time classifying our categories, yet it scores better than the linear and poly kernels.

Shape

Description automatically generated with medium confidence

Looking at the Radial Basis Function (RBF) kernel, it is the exact kernel that we are looking for when it comes to the circles-dataset. The circles are namely scattered at the perimeter of two slightly different sized circles and therefore, the RBF kernel can perfectly classify decisions for the dataset. It must be said that even though we got an accuracy score of 100% this time, that is not the case for every single render. However, it is always in the 95th percentile.

# Exercise 2

## Subtask A

asd