Machine Learning: Exercise Set VII (a)

Georgios Manos
csd4333

24 December 2022

1 Support Vector Machine (Theoretical)

1.1 Decision Function

The decision function is:

$$f(x_{test}) = w \cdot x_{test} + b$$

In the dual form of the support vector machine (SVM), the weight vector w is given by:

$$w = \sum_{i}^{N_s} a_i y_i x_i$$

Where N_s represents the number of support vectors. Essentially, those are the samples that satisfy the condition $0 < a_i < C \implies 0 < a_i < 10$, hence x_1 and x_2. Substituting the values from the given training data into the above equation, the weight vector w becomes:

$$w = (1 \cdot (-1) \cdot (1, 0, 1)) + (1 \cdot 1 \cdot (0, -1, 0)) = (-1, -1, -1)$$

To compute the bias term b, we can use the KKT conditions. We know that $\xi_i = 0$ for $a_i < C$. Thus we can take all training points that satisfy that condition to use in the following KKT condition equation, and keep the average out of them*:

$$a_i y_i (x_i \cdot w + b) - 1 + \xi_i = 0 \underset{\xi_i = 0}{\implies} y_i (x_i \cdot w + b) = 1 \implies b = y_i - w \cdot x_i{}^{\dagger}$$

Substituting again the values from the given training data into the above equation, the bias term b becomes:

$$b = 1/2 \cdot ((-1 - (-1, -1, -1)(1, 0, 1)^T) + (1 - (-1, -1, -1)(0, -1, 0)^T)) = \frac{1}{2}$$

*C. Burges on "A Tutorial on Support Vector Machines for Pattern Recognition" suggests taking the average over all such training points is numerically wiser

†$y_i = \pm 1 = \frac{1}{y_i}$

Finally, the decision function $f(x_{test})$ is expressed as:

$$f(x_{test}) = w \cdot x_{test} + b = (-1,-1,-1) \cdot x_{test} + \frac{1}{2}$$

Note that optionally we could normalize the weight vector w prior to the bias calculation to get margins of 1.

1.2 Lagrange Multipliers

There are 2 ways to show that these Lagrange multipliers cannot really be the solution to the proposed problem.

First of all, the Lagrange multiplier of the 3rd sample, where $a_3 = 10 = C$ shows that the specific sample will be missclassified and won't be a support vector. However, using our function:

$$f(x_3) = (-1,-1,-1) \cdot (1,1,-1) + \frac{1}{2} = -1 - 1 + 1 + 0.5 = -0.5$$

Hence $sign(f(x_3)) = -1 = y_3$.

Secondly, from the Primal formulation we have the following constraint:

$$y_i(wx + b) \geq 1 - \xi_i, \forall x_i,\ \xi_i \geq 0$$

But for the samples x_4 and x_5, it is:

$$-1(-2 + 0.5) \geq 1 - \xi_i \implies 1.5 \geq 1 - \xi_i$$

and

$$1(1 + 0.5) \geq 1 - \xi_i \implies 1.5 \geq 1 - \xi_i$$

Which are both impossible for $\xi_i \geq 0$. So those Lagrange multipliers violate the KKT conditions for the primal problem.

1.3 Gaussian Kernel

Assuming Gaussian Kernel, we will refer to the kernel trick for our decision function. Hence, $f(x_{test})$ now will be:

$$f(x_{test}) = \sum_{i}^{N_s} a_i y_i K(s_i, x_{test}) + b$$

over all support vector samples s_i. For b, we will again take the average of the solutions for the following equation over all support vectors:

$$a_j(y_j \sum_i a_i y_i K(x_i, x_j) + b - 1) = 0$$

So:

$$b = \frac{1}{N_s}(\sum_{j}^{Ns} 1 - y_j \sum_i a_i y_i K(x_i, s_j))$$

Where $K(x_i, x_j) = e^{\frac{-\|x_i - x_j\|}{2\sigma^2}}|_{\sigma=1} = e^{\frac{-\|x_i - x_j\|}{2}}$ (hence, if $i = j$ then $K(x_i, x_i) = 1$). Therefore:

$$b = \frac{1}{2}(1 - (-1)(-K(x_1, x_1) + K(x_2, x_1)) + 1 - 1(-K(x_1, x_2) + K(x_2, x_2)))$$

$$= \frac{1}{2}(2K(x_1, x_2)) = K(x_1, x_2) = e^{\frac{-\|x_1 - x_2\|}{2}} = e^{-\frac{\sqrt{3}}{2}} \implies b \approx 0.42$$

Finally, the decision function $f(x_{test})$ is:

$$f(x_{test}) = \sum_i^{N_s} a_i y_i K(s_i, x_{test}) + b = -K(x_1, x_{test}) + K(x_2, x_{test}) + 0.42$$

To predict the class of the new input vector x_{test}, we use the sign function, both in this part and the first one (section 1.1):

$$\hat{y} = sgn(f(x_{test}))$$

2 Support Vector Machine (Programming)

2.1 Implementation Details

The python script would take quite some time to execute. This is due to that SVC as implemented in sklearn takes a lot of time to fit for linear kernel and especially C = 1 or C = 10, for any fold (about 4 minutes on my PC, and sometimes does not converge).

I found online that its best practice to Scale my training and test input, and for this purpose I also used StandardScaler in this exercise. With StandardScaler, the program runs in 0.64 seconds(!) for all configs, and while the resulting AUC is the same, the best configuration is different.

Finally, on the previous exercise I used the sklearn StratifiedKFold implementation as well, and I'm reusing that function as it is on this exercise too.

2.2 Results

The best configuration without scaling the data I found is C=0.01, with Linear kernel and gamma=0.1 [‡] and then there is the probability = True argument that is required to calculate the predicted class probabilities, in order to use them for the AUC score computation.

When Using the StandardScaler, the best configuration found is C=10, with RBF kernel and gamma = 0.1. Also note that the Standard scaler is fit on the training data and applied on both training and test data, without violating the Golden Rule.

[‡]although gamma is only used on rbf kernel and not on the linear one, and the results indeed with linear kernel are not affected by the gamma paramater

The Cross Validation performance (average AUC over the 5 test folds) is approximately 0.924, while the hold-out AUC score is approximately 0.941 (both with and without the scaler). Once again, we see that the cross validation performance is a conservative estimate of the actual model's performance as the estimation is taken from a model that wasn't trained on all available data, unlike the final model that was used to predict on the hold-out set. I also plotted the ROC curve on the hold-out set, for reference.

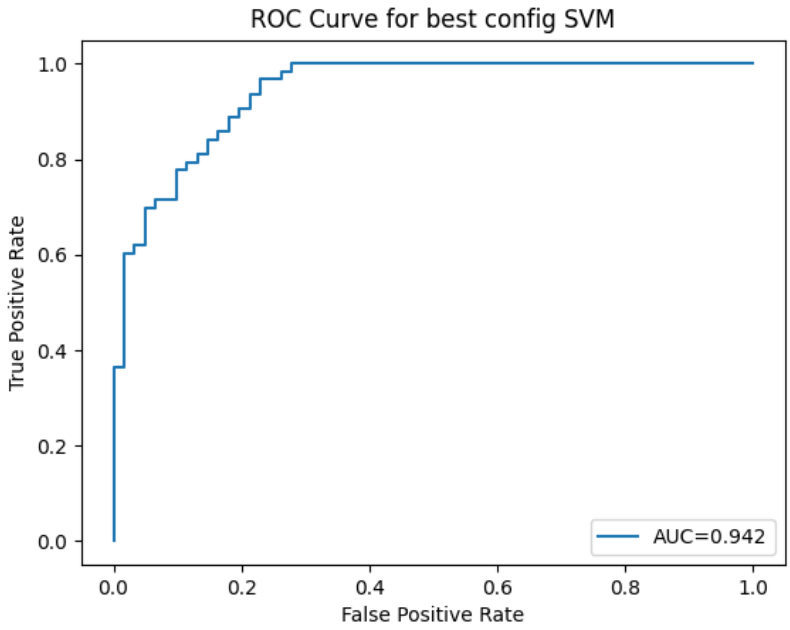

Figure 1: ROC Curve from the best configuration model applied on Standardized Data.