ENGR-421 HW-6

Berke Can Rizai – 69282

We import libraries and generate the array from the given images, labels.

Text

Description automatically generated with low confidence

We split the data for train, test.

Graphical user interface, text, application, chat or text message

Description automatically generated

We define the gaussianity function that is provided in the lab and take the number of data points and classes.

Text

Description automatically generated

Since we will do one-versus-all classification, I define all the label arrays here. We will use this for traning support vectors. I change the target class labels as 1 and all other labels are -1.

Text

Description automatically generated

I define the support vector machine function as follows, it takes labels as arguement and C (punishment factor), S values are optional arguements that are set to 10.

Text

Description automatically generated

This follows the exact same logic as we did in the lab, first, we get the K\_train by giving the gaussian\_kernel the x\_train arrays as well as the s, width.

We then, create the arrays to feed our optimizer and run it with cvx.solvers.qp.

In the alpha, if values are really low that means we don’t care about them and they are disregarded by setting them to 0.

I return the alpha and w0.

Then, for each class, I run the binary classifier and store their values as follows,





And such.

I then, calculate the predicted value for the entire training dataset.

This is done by “np.matmul(ktrain1, y1[:,None] \* a1[:,None]) + w01”.A picture containing graphical user interface

Description automatically generated

We just multiple trainset, labels and alpha vector and add the w0 values.

We also get y\_predicted but they are not really relevant here. We don’t use them.

Others are calculated as same, 

Then, I get the max of each row with the np.argmax with all the f\_predicted values and since here, array start with 0, I add 1 to make it for our case.

Training performance is as follows,

Text

Description automatically generated

That is calculated with, 

We then, run the recognizer in the test data with the Text

Description automatically generated

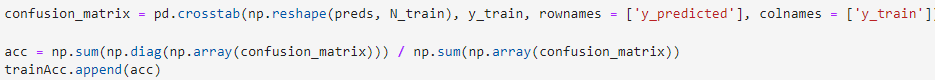
Where y1 was labels for class 1 vs others, a1 and w01 were values for support vectors.

For the different c values to try, I created a list for these values and from this list, called the svm and trained for each class then calculated f\_preds for the training. Then, I get the y\_pred with argmax again.

Table

Description automatically generated

To calculate the accuracy, I sum all correct classifications with the np.diag as all diagonal values are correct labels and divide them with total number. Then append this to error list for train.



Later, I do the same for test data.

Text

Description automatically generated

Here everything is same as before, we calculate f\_preds then get max, error and then add error to list.

Then we have the,

Graphical user interface, text, application

Description automatically generated

For plotting, I set the size to 10-6 and I plot train errors with blue dots and test with red.

I add legend to up left and set the scale for x as log so that it looks same as the requested logarithmic ticks. Then we are done with it.

Text

Description automatically generated