SVM Report – Ian Valle

Program details

First, I made sure to import the right modules I would need for the program (sklearn, matplotlib, csv, and warnings) as well as setup the functions. One to load the data which converts the dataset file into a list we can work with and one to setup folds of data for training, validating and testing. After loading the data, I also initialized the variables determining the total accuracy over five iterations for all svm types and setup the weights dictionary for later use.

Folds = for j in range len(data)

1/5 = test set

1/5 = val set

4/5 = training set

Then for five iterations for the rest of the program I initialize the empty sets to be used for training, validating and testing as well as their labels, then I setup the list of all the hyperparameters, with the argmax variables and max function value variable. Now the data is divided into the proper folds and the labels lists are setup for easier use later.

Now we start working our way through each one of the svm kernels and types by first predicting the data in the validation set for all values of the hyperparameters and determining their best values by averaging the number of correct values over all of them. Then we use these best hyperparameters to predict the data on the testing set by again averaging the number of correct values over all of them and add the data up from the five iterations. We do all these steps for all the svm kernels / types: RBF, Linear, Polynomial, Sigmoid, One v.s. All, RBF weighted, Linear weighted, Polynomial weighted, and Sigmoid weighted

Optimal Hyperparamaters = For I in range (each hyperparameter)

… iniitialize prediction, count

...check accuracy for all combinations of hyperparameters

… return the parameters that have maximized the accuracy

For all the kernel types...

Predict data = Initialize prediction, count

Take accuracy with the best parameters

Return accuracy to a variable that adds up the average over all iterations

Finally, we output the accuracy data as strings, that are easy to read, for each kernel remembering to divide it by the 5 times we accumulated data. There is also plot setup to show this data as a bar graph for easier comparison between all the svm kernels and types.

Results

Hyperparameters tested

C = [1e^-5, 1e^-4, 1e^-3, 1e^-2, 1e^-1, 1, 10, 100, 1000]

Gamma = [3, 2, 1, 1e^-1, 1e^-2, 1e^-3, 1e^-4, 1e^-5, 1e^-6]

Degree = [2, 3, 4, 5, 6, 7]

Coef = [1e^-2, 1e^-1, 0.5, 1, 2, 3, 4, 5, 7, 10]

For RBF the most accurate hyperparameters were C = 1 for all iterations, gamma = 0.1 for 4 out of 5 iterations

For Linear the most accurate hyperparameter was C = 1e^-5 for all iterations

For Polynomial the most accurate hyperparameters were C = 1e^-5 for all iterations, gamma = 0.1 for 4 out of 5 iterations, degree = 2 for all iterations

For Sigmoid the most accurate hyperparameters were C = 1000 for 4 out of 5 iterations, gamma = 1e^-5 for all iterations, coef = 0.01 for all iterations

One v.s. One against One v.s. All

Label	Accuracy	Time
OvO RBF	98.22%	0:00:00.33
OvO Linear	82.83%	0:00:00.13
OvO Polynomial	99.41%	0:00:00.64
OvO Sigmoid	98.82%	0:00:02.55
One v.s. All	83.44%	0:00:00.22

All but one of the kernels in OvO have better accuracy than the OvA SVM by about 15% however the OvA time is the second best only off by 0:00:00.09 from the linear kernel which happens to have the worst accuracy. This table shows that taking more time in the kernel will considerably increase the accuracy for only a small increase in time.

Weighted

Label	Accuracy
W RBF	91.69%
W Linear	92.91%

W Polynomial	99.41%
W Sigmoid	98.82%

The weighted SVM prediction on the data did not have the effects I was expecting when comparing it to the unweighted data. RBF seems to be the only kernel to decrease in performance, meanwhile the Linear kernel shows considerably improvement. Both the polynomial and sigmoid kernels show no difference between weighted and unweighted data. I suspect this is because there are not enough samples to work with in this data set and would need over well over 1000 samples.