ASSIGNMENT 3

Group Details:

Group - 002

18CS10069 - Siba Smarak Panigrahi

18CS30051 - Debajyoti Dasgupta

November 15, 2020

A. Procedure:

1. We have used both Binary SVM classifiers (with Linear, Quadratic, and Radial Basis Function as kernels) and Multi-Layer Perceptron classifier with different learning rates for the assignment. The data used for training was in .csv format. We have shuffled the data. We considered a split of 80:20 for the Train and Test set. It had 41 attributes and one target variable, i.e. experimental class.

2. Major functions:

`svm_classifiers`: This function takes a list of kernels, trains an SVM classifier, and returns the accuracy lists of train and test accuracies. It keeps the value of C to be constant i.e. C = 1.0

`model`: This function creates the MLP classifier and fits the model on the input training data.

3. Some important Helper functions:

`find_best_C`: This function returns the train and test accuracies for an input list of kernels with C values of 1.e-03 to 1.e+max_C. Here max_C is an input parameter to the function.

`tune_learning_rate`: This function returns the best MLP Classifier and its corresponding score along with all other scores for different learning_rates. It basically returns a dictionary for each hidden-layer, which in turn is a dictionary with learning-rate as keys and the respective scores as values.

`tune_model`: This function also returns the best MLP Classifier and its corresponding score along with all other scores for different learning_rates. But it

has a different return format of dictionary for each learning-rate, which in turn is a dictionary with different combinations of hidden-layers as keys and the respective scores as values.

`split_data:` This function splits the data into two sets train and test in 80:20 ratio.

`print_ker_acc:` This function prints the result of various kernels and corresponding C-values present in given dictionaries in a tabular form

`best_ker_acc:` This function prints the best C value for each kernel from input train accuracies (train_acc) and test accuracies' (test_acc) dictionaries.

`plot_scores:` This function plots learning-rate vs accuracy for each different hidden-layer combination provided.

`plot_scores_3:` This function plots different hidden-layer combination vs accuracy for each different learning-rate provided.

4. Support Vector Classifier

i. Binary SVM Classifier is a linear discriminant classifier.

ii. **Uses Vapnik's principle:** to never solve a more complex problem as a first step before the actual problem.

iii. After training the weight vector can be written in terms of training samples lying in class boundaries.

iv. The Primal problem for soft-margin hyperplanes and kernel function:

Minimize w.r.t **'w'**

$$L_p = \frac{1}{2}||w||^2 + C * \Sigma_t s^t - \Sigma_t \alpha^t [r^t (w^T \phi(x^t) + w_0) - 1 - s^t] - \Sigma_t \mu^t s^t$$

5. Multi-Layer Perceptron

i. It is a network of perceptrons arranged in a particular fashion, called the architecture of the model.

ii. We consider a multi-layered feed-forward network, this means there is no feed-back or loop in the network.

iii. In this algorithm we define an Error function, and we try to minimize w.r.t **W** parameter using either Gradient Descent or Stochastic Gradient Descent (SGD).

SGD is similar to Gradient Descent only in SGD we update the **W** after each training example.

iv. The back-propagation algorithm helps to determine the changes in each and every weight parameter at each level. The back-propagation algorithm is simply the chain-rule with partial differentiation.

Minimize: $J_n(W) = \frac{1}{N} \Sigma_i ||O_i - F(X_i; W)||^2$

We start with W_0. In SGD, we update the weights by the following rule:

$$W_i = W_{i-1} + \eta(i) \Sigma_k (O_k - F(X_k; W_{i-1})) \nabla F(X_k; W_{i-1})$$

B. Results

(Note: We have scaled the dataset with Standard Scaler to make the training faster)

1. Binary SVM Classification:

Best value of C:

TABLE 1. Values of C and corresponding train and test accuracies

Kernel Name	Value of C	Training Accuracy (in %)	Test Accuracy (in %)
Radial Basis Function	0.001	71.4	64.9
	0.01	71.4	64.9
	0.1	71.4	64.9
	1.0	81.0	76.8
	10.0	87.6	83.9
	100.0	**90.8**	**87.7**
	1000.0	94.2	86.7
	10000.0	97.5	85.8
	100000.0	99.2	85.3
Linear	0.001	78.1	70.6
	0.01	86.3	82.9

	0.1	89.0	84.8
	1.0	89.5	85.3
	10.0	89.9	85.3
	100.0	90.6	84.8
	1000.0	90.5	85.3
	10000.0	88.2	85.3
	100000.0	**88.5**	**86.7**
Quadratic	0.001	71.4	64.9
	0.01	71.4	64.9
	0.1	71.4	64.9
	1.0	79.5	73.0
	10.0	86.7	82.5
	100.0	**91.6**	**87.2**
	1000.0	92.5	87.2
	10000.0	95.3	87.2
	100000.0	97.3	85.3

(The best accuracies are highlighted in the above table.)

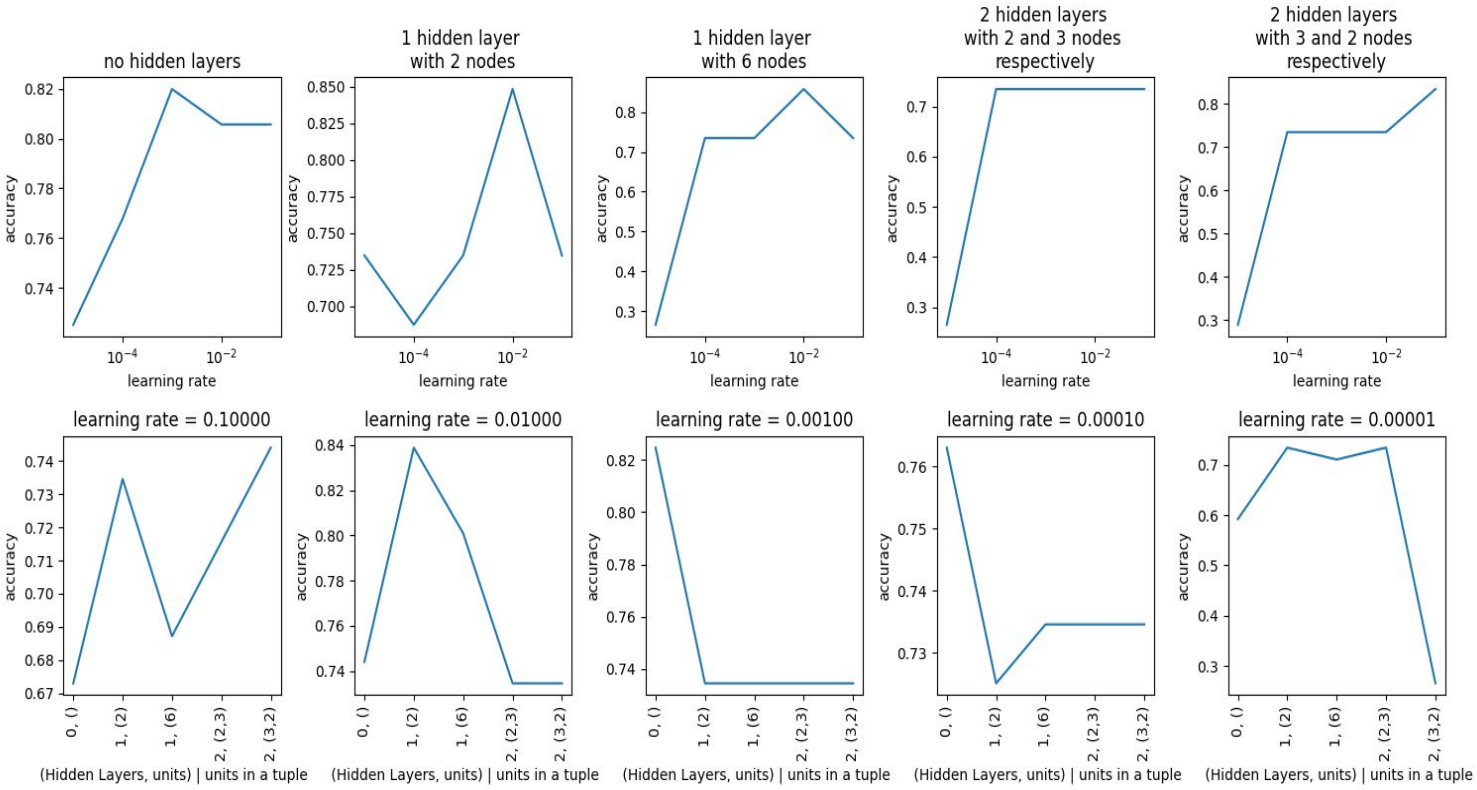
2. MLP Classifier:

Input size: 41

Output unit: 1

a. The plots of hidden-layer combination for different learning-rates:

FIGURE 1. The plots of learning rate vs accuracy for each model (top row); and model vs accuracy for each learning rate (bottom row)



b. Best model parameters:

i. 'activation': 'logistic',

ii. 'alpha': 0.0001,

iii. 'batch_size': 'auto',

iv. 'beta_1': 0.9,

v. 'beta_2': 0.999,

vi. 'early_stopping': False,

vii. 'epsilon': 1e-08,
viii. 'hidden_layer_sizes': [6],
ix. 'learning_rate': 'constant',
x. 'learning_rate_init': 0.01,
xi. 'max_fun': 15000,
xii. 'max_iter': 200,
xiii. 'momentum': 0.9,
xiv. 'n_iter_no_change': 10,
xv. 'nesterovs_momentum': True,
xvi. 'power_t': 0.5,
xvii. 'random_state': None,
xviii. 'shuffle': True,
xix. 'solver': 'sgd',
xx. 'tol': 0.0001,
xxi. 'validation_fraction': 0.1,
xxii. 'verbose': False,
xxiii. 'warm_start': False

Justification for the hyperparameters:

1. At a learning rate of 0.01, there is a smooth curve , indicating that the model is not overfitting on the the data
2. From the first part of Q2 we find that out of tanh, ReLU, and logistic, the logistic activation function performs best on this data
3. SGD Solver has not been tuned since in the problem statement it was asked to use stochastic gradient descent
4. For the model of 1 hidden layer with 6 neurons, we can make the same argument that it gives us a smooth gradient descent, which means it is not over fitting on data and provides the best test accuracy

Hence due to the above reasons, these above parameters for the model were selected.

3. Comparison of Performance

a. The dataset prefers the Binary SVM Classifier compared to the various architectures used for MLP Classifier. This simply means that SVM Classifier with soft-margin hyperplanes captures the input-output variations of this dataset better than the network of perceptrons.

b. But we could observe that the MLP Classifier with very few hidden-layers is performing comparably to Binary SVM Classifier. But as we increase the value of C in SVM, we observe that it beats the MLP Classifier.

c. The corresponding best accuracies for each of the model are mentioned below:

   i. For Binary SVM Classifier with **Radial Basis Function kernel**, we have **training accuracy = 90.8%**, **test accuracy = 87.7%**, when the **value of C is 100.0**

   ii. For the above MLP Classifier (best model; parameters mentioned above) we have **training accuracy = 70.0%**, **test accuracy = 70.6%**