CIS 6930/4930: Deep Learning for Computer Graphics, Fall’ 20
Project 1: Tic-Tac-Toe

Team Members:

Sanjay Reddy Banda, UFID: 5878-2239

Chandra Sekhar Deverakonda, UFID: 9092-2981

Suprith Reddy Gurudu, UFID: 9961-2134

Implementation:

As mentioned in the project description, we have used three classifier models namely K-Nearest Neighbors, Multi-layer Perceptron, and Support Vector Machine on all three datasets and three regressor models includes K-Nearest Neighbors, Multi-layer perceptron, and Linear Regression on the intermediate (multi-label) dataset only. We have calculated the accuracies and normalized confusion matrices using the predicted and true labels from the models. A detailed explanation is provided below for each model.

Multi-layer Perceptron (MLP) Classifier:

A multi-layer perceptron is a class of feed forward artificial neural networks. Multi-layer perceptron contains one input layer, several hidden layers and one output layer. They are sometimes referred to as Vanilla neural networks. ScikitLearn library has MLPClassifier function which will allow us to implement MLP classifier on the data. It has certain parameters that help us in specifying number of hidden layers, activation function etc. Particularly, for this data set, we have implemented 3 hidden layer architecture. The first two hidden layers contain 256 nodes each and the final layer contains 128 nodes followed by Rectified Linear Unit (ReLu) activation function. We trained our model using StratifiedKFold cross validation with a 98% validation score and achieved 95% accuracy on test set.

We estimated the true performance of the classifiers using Confusion Matrix and Accuracy score.

Multi-layer Perceptron (MLP) Regressor:

A multi-layer perceptron regressor is similar to MLP regressor except that it produces continuous values unlike classifier which produces nominal variables. Regressor was trained using Mean Squared Error (MSE) over training data. MLPRegressor also has 3 hidden layers – two layers with 256 nodes each and final layer with 128 nodes. As the regressor outputs continuous values, we have implemented another helper function to fit the data properly. MLPRegressor accuracy stands at around 95%, which is less than MLPClassifier accuracy. This is mainly due to the implementation of non-linear activation functions in the classifier. The regressor scores are reported using ScikitLearn.Metrics library.

K-Nearest Neighbors (KNN):

An KNN is a supervised learning model used for both classification and regression tasks by labeling the samples according to the nearest neighbors. The Hyper parameters in this learning model are:

1. Number of Neighbors (k)
2. Distance function (i.e. Euclidean, Manhattan, Hamming, etc.)

As we increase the k value the accuracy of the model increases but it also comes with the requirement of more computational power. So, I have analyzed the model performance and execution found a sweet spot for both classification and regression. I have implemented K-fold strategy with 10 folds to fit the data to the model.

Classifier: For KNN classification model I have used “9” neighbors and “Euclidean Distance” as hyper parameters.

Regressor: For KNN regressor model I have used “9” neighbors and “Manhattan Distance” as hyper parameters.

Support Vector Machine (SVM) Classifier:

An SVM is a supervised machine learning model used for classification tasks by constructing a hyperplane or set of hyperplanes that separates the data into classes.

Prior to model building, we have loaded the data as features and labels, assigned the test size to 20% of original dataset, applied a function to split the data into train and test subsets, and initialized k-fold strategy to 10 to overcome the challenge of overfitting.

Once the preprocessing steps are completed, the model is built based on the hyper parameters passed into the SVM classifier function. Among the parameters, kernel is a significant one which is set to ‘linear’ (final dataset) and ‘rbf’ (intermediate datasets) to reduce the computations involved in higher dimensions and get better predictions.

Finally, the predicted labels and accuracy is calculated using *classifier.predict()* and *classifier.score()* functions along with the confusion matrix.

Linear Regressor using normal equations:

The objective of a linear regression function is to fit the best line/hyper-plane to a set of observations by computing the parameters that minimizes the cost function, for instance, mean squared error (MSE).

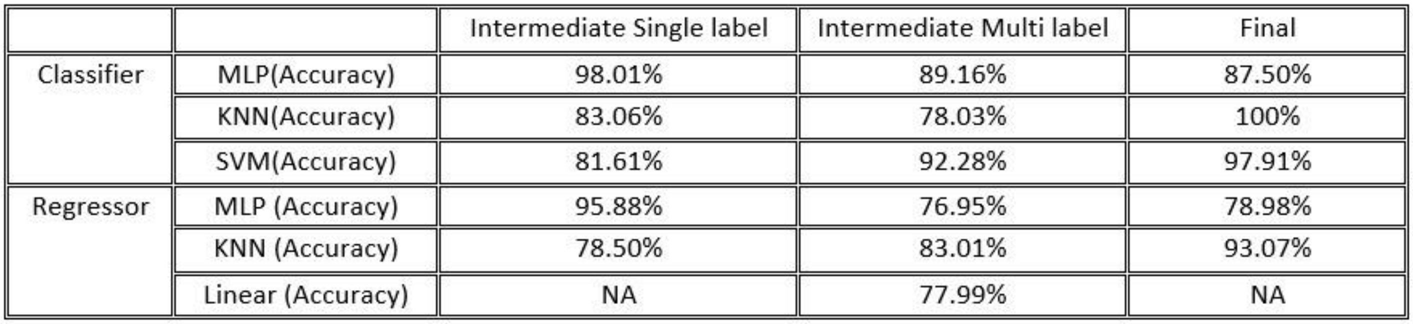
Prior to model building, we have loaded the data as features and labels, assigned the test size to 20% of original dataset, applied a function to split the data into train and test subsets, and initialized k-fold strategy to 10 to overcome the challenge of overfitting.

After preprocessing, the procedure of applying regression on the training set involves five different steps: (i) transposing the training feature matrix, (ii) multiplying the transposed matrix with the original training feature set, (iii) inverting the product from the previous step, (iv) multiplying the inverted matrix with transposed matrix, and (v) multiplying the resultant from previous step to the training label vector that produces theta which describes slope and intercept of the line/hyper-plane.

Finally, the predicted labels are calculated by multiplying theta with testing sample of features. As the predicted values are continuous, we cannot define accuracy like classification. So, we have rounded off the predicted values to the nearest integer that further used to compute the accuracy using *metrics.accuracy_score()* function.

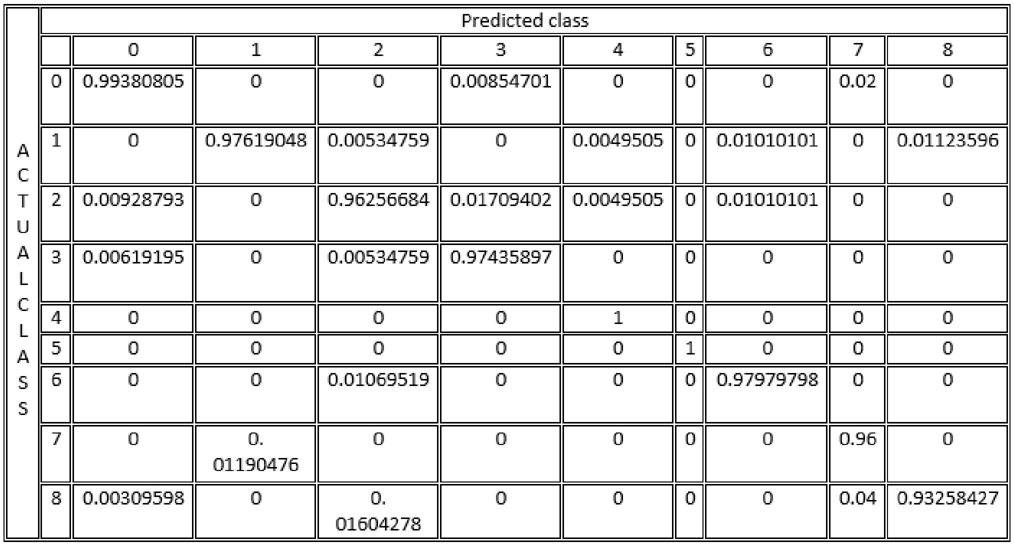
Evaluation:

Accuracy and normalized confusion matrices:



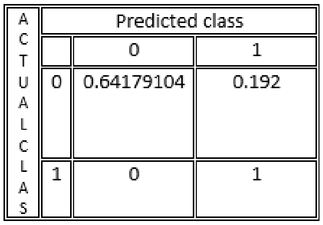
		Intermediate Single label	Intermediate Multi label	Final
Classifier	MLP(Accuracy)	98.01%	89.16%	87.50%
	KNN(Accuracy)	83.06%	78.03%	100%
	SVM(Accuracy)	81.61%	92.28%	97.91%
Regressor	MLP (Accuracy)	95.88%	76.95%	78.98%
	KNN (Accuracy)	78.50%	83.01%	93.07%
	Linear (Accuracy)	NA	77.99%	NA

MLP “Intermediate Single Labeled Data set”



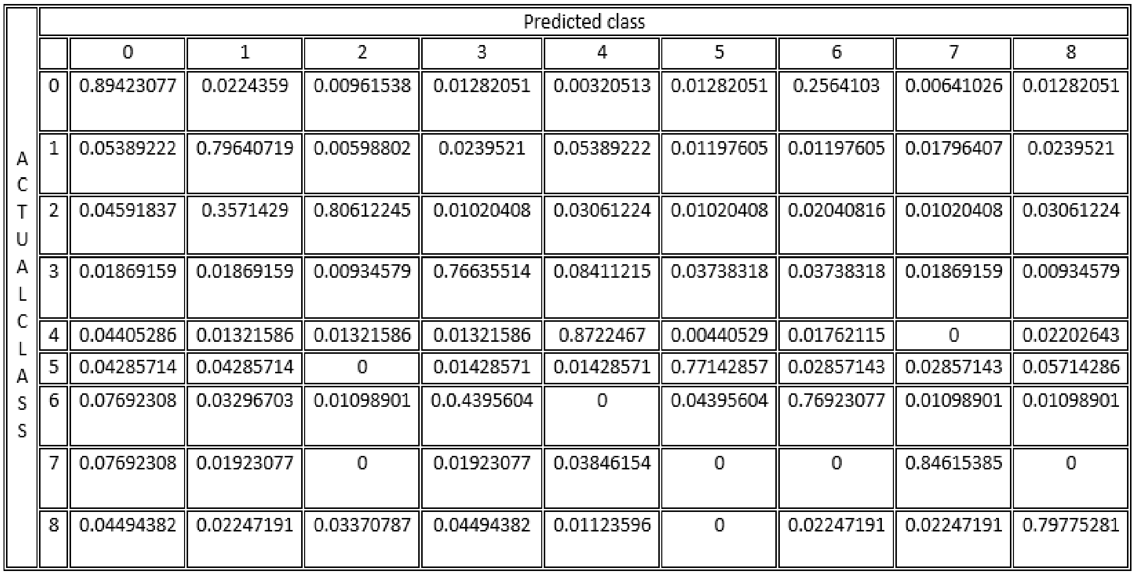
		Predicted class								
		0	1	2	3	4	5	6	7	8
ACTUALCLASS	0	0.99380805	0	0	0.00854701	0	0	0	0.02	0
	1	0	0.97619048	0.00534759	0	0.0049505	0	0.01010101	0	0.01123596
	2	0.00928793	0	0.96256684	0.01709402	0.0049505	0	0.01010101	0	0
	3	0.00619195	0	0.00534759	0.97435897	0	0	0	0	0
	4	0	0	0	0	1	0	0	0	0
	5	0	0	0	0	0	1	0	0	0
	6	0	0	0.01069519	0	0	0	0.97979798	0	0
	7	0	0. 01190476	0	0	0	0	0	0.96	0
	8	0.00309598	0	0. 01604278	0	0	0	0	0.04	0.93258427

MLP “Final Single Labeled Data set”



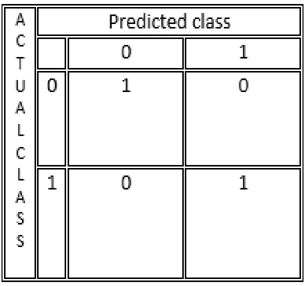
		Predicted class	
		0	1
ACTUALCLASS	0	0.64179104	0.192
	1	0	1

KNN Intermediate board Single labeled Data set:

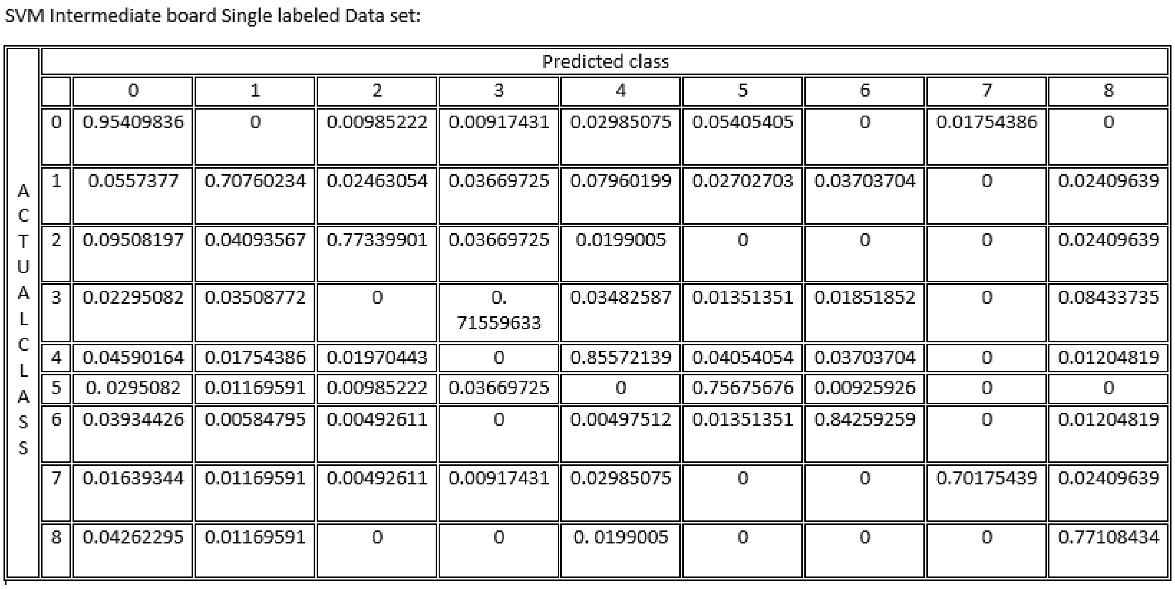


		Predicted class								
		0	1	2	3	4	5	6	7	8
ACTUALCLASS	0	0.89423077	0.0224359	0.00961538	0.01282051	0.00320513	0.01282051	0.2564103	0.00641026	0.01282051
	1	0.05389222	0.79640719	0.00598802	0.0239521	0.05389222	0.01197605	0.01197605	0.01796407	0.0239521
	2	0.04591837	0.3571429	0.80612245	0.01020408	0.03061224	0.01020408	0.02040816	0.01020408	0.03061224
	3	0.01869159	0.01869159	0.00934579	0.76635514	0.08411215	0.03738318	0.03738318	0.01869159	0.00934579
	4	0.04405286	0.01321586	0.01321586	0.01321586	0.8722467	0.00440529	0.01762115	0	0.02202643
	5	0.04285714	0.04285714	0	0.01428571	0.01428571	0.77142857	0.02857143	0.02857143	0.05714286
	6	0.07692308	0.03296703	0.01098901	0.0.4395604	0	0.04395604	0.76923077	0.01098901	0.01098901
	7	0.07692308	0.01923077	0	0.01923077	0.03846154	0	0	0.84615385	0
	8	0.04494382	0.02247191	0.03370787	0.04494382	0.01123596	0	0.02247191	0.02247191	0.79775281

KNN “Final Single Labeled Data set”

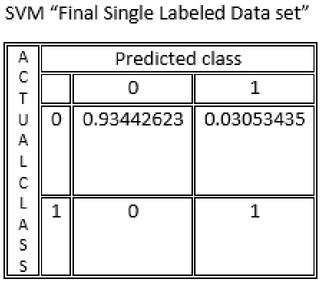


		Predicted class	
		0	1
ACTUALCLASS	0	1	0
	1	0	1



SVM Intermediate board Single labeled Data set:

		Predicted class								
		0	1	2	3	4	5	6	7	8
ACTUAL CLASS	0	0.95409836	0	0.00985222	0.00917431	0.02985075	0.05405405	0	0.01754386	0
	1	0.0557377	0.70760234	0.02463054	0.03669725	0.07960199	0.02702703	0.03703704	0	0.02409639
	2	0.09508197	0.04093567	0.77339901	0.03669725	0.0199005	0	0	0	0.02409639
	3	0.02295082	0.03508772	0	0. 71559633	0.03482587	0.01351351	0.01851852	0	0.08433735
	4	0.04590164	0.01754386	0.01970443	0	0.85572139	0.04054054	0.03703704	0	0.01204819
	5	0. 0295082	0.01169591	0.00985222	0.03669725	0	0.75675676	0.00925926	0	0
	6	0.03934426	0.00584795	0.00492611	0	0.00497512	0.01351351	0.84259259	0	0.01204819
	7	0.01639344	0.01169591	0.00492611	0.00917431	0.02985075	0	0	0.70175439	0.02409639
	8	0.04262295	0.01169591	0	0	0. 0199005	0	0	0	0.77108434



SVM “Final Single Labeled Data set”

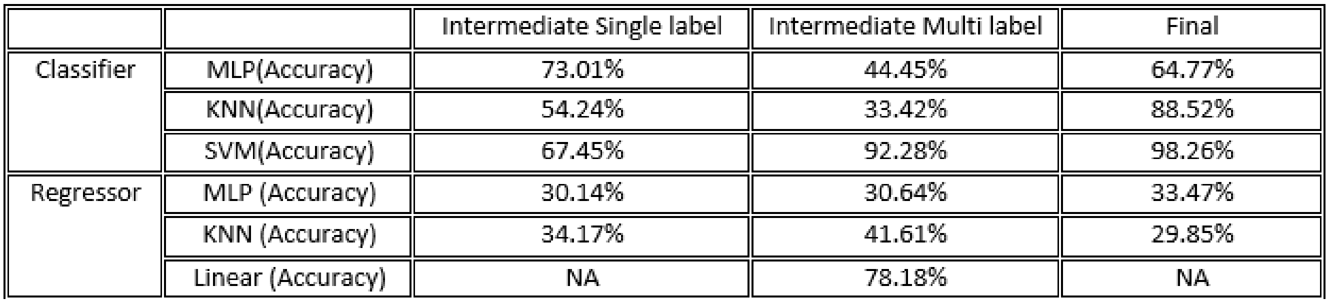
		Predicted class	
		0	1
ACTUAL CLASS	0	0.93442623	0.03053435
	1	0	1

Best methods in classification and regression:

On final dataset, KNN classifier and regressor works best as the accuracy reaches 100 and 93 percent, respectively. Similarly, SVM classifier is the best method applied on intermediate multilabel dataset and MLP classifier on intermediate single label dataset. MLP classifier predicts more accurate values on testing samples for intermediate single label dataset and KNN regressor for intermediate multilabel dataset.

Investigation:

On running the classifiers on 1/10th of the data we got following results:



		Intermediate Single label	Intermediate Multi label	Final
Classifier	MLP(Accuracy)	73.01%	44.45%	64.77%
	KNN(Accuracy)	54.24%	33.42%	88.52%
	SVM(Accuracy)	67.45%	92.28%	98.26%
Regressor	MLP (Accuracy)	30.14%	30.64%	33.47%
	KNN (Accuracy)	34.17%	41.61%	29.85%
	Linear (Accuracy)	NA	78.18%	NA

Scaling:

MLP performs well with having more and more data because it has deeper network and are capable of making strong correlations between the data and relying less on assumptions. But there will be a limit on size of data where it starts overfitting the model. KNN performance will not be impacted on the size of data set. Rather its performance depends on the distributedness of the labels in the dataset. Similarly with SVM performance is not much impacted with the size of the dataset.

This problem is not open ended in terms of board states and outcomes as it has finite set of inputs and outcomes.