Information – SL Assignment (Neural Networks)

OptDigits Dataset:

- Status - Completed.
- It was an easy implementation of Multi-Layer Perceptron on the OptDigits dataset. I have first given indices to all the attributes and labelled the class attribute as Target so that WEKA can understand it.
- Multi-Layer perceptron in WEKA is a Classifier that uses backpropagation to classify instances. This network can be built by hand, created by an algorithm or both. The network can also be monitored and modified during training time. The nodes in this network are all sigmoid (except for when the class is numeric in which case the the output nodes become unthresholded linear units).
- The number of hidden layers in the MLP is the most important parameter to classify the class values. There are wildcard values 'a' = (attribs + classes) / 2, 'i' = attribs, 'o' = classes, 't' = attribs + classes.
- The learning rate by default is 0.3, momentum rate is by default 0.2.
- For ‘a’ number of hidden layers WEKA gives accuracy of around 96% and the confusion matrix for each class portrays that classes 3, 7 and 8 are falling behind the classification when compared to other classes. A = (64+10)/2 = 36 hidden layers.

=== Detailed Accuracy By Class ===

TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
1.000	0.001	0.994	1.000	0.997	0.997	1.000	1.000	0
0.995	0.006	0.948	0.995	0.971	0.967	0.999	0.987	1
0.994	0.005	0.957	0.994	0.975	0.973	0.999	0.992	2
0.907	0.001	0.994	0.907	0.949	0.944	0.997	0.984	3
0.989	0.003	0.973	0.989	0.981	0.979	1.000	0.997	4
0.989	0.009	0.928	0.989	0.957	0.953	0.999	0.983	5
0.989	0.000	1.000	0.989	0.994	0.994	0.999	0.997	6
0.927	0.002	0.982	0.927	0.954	0.950	0.999	0.994	7
0.908	0.004	0.963	0.908	0.935	0.929	0.996	0.982	8
0.972	0.007	0.941	0.972	0.956	0.951	0.998	0.989	9

Weighted Avg. 0.967 0.004 0.968 0.967 0.967 0.964 0.999 0.990

- Let's take 'i' number of hidden layers. It also gives accuracy of around 96% and this time we can say that the classes 3,7 and 8 were again classified a little accurately than compared to other classes. I = 64 number of hidden layers.

=== Detailed Accuracy By Class ===

TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
0.989	0.001	0.994	0.989	0.992	0.991	1.000	1.000	0
0.995	0.006	0.953	0.995	0.973	0.970	0.999	0.991	1
0.989	0.002	0.983	0.989	0.986	0.984	0.999	0.995	2
0.934	0.000	1.000	0.934	0.966	0.963	0.999	0.994	3
0.989	0.002	0.978	0.989	0.984	0.982	1.000	0.998	4
0.989	0.009	0.928	0.989	0.957	0.953	0.999	0.996	5
0.983	0.001	0.989	0.983	0.986	0.985	1.000	0.999	6
0.933	0.001	0.988	0.933	0.960	0.956	1.000	0.997	7
0.943	0.004	0.959	0.943	0.951	0.946	0.996	0.984	8
0.972	0.006	0.951	0.972	0.962	0.957	0.999	0.992	9
Weighted Avg. 0.972	0.003	0.972	0.972	0.972	0.969	0.999	0.995	

- Let's take the number of hidden layers as 74 = Atrribs+Classes , we get an accuracy of around 97%.
- We can also see that the mean squared error has been reducing as we keep increasing the hidden layers. Accuracy increases as we increase hidden layers as well.
- But we need to stop increasing the number of hidden layers until we get the highest accuracy possible as we can overfit the model with training data.
- Overall, I can say that the neural networks algorithm was an improvement over the Decision Tree model but the time to train the model and predict values (150 seconds) is a lot more than training and predicting from decision tree model (20 seconds).

Amazon Reviews Dataset:

- Status – Completed.
- The challenging part dealing with Amazon Reviews Dataset is to understand how to classify text data.
- Implemented Bag of Words algorithm to the reviews column of the dataset. Cleaned the data from stopwords (using NlTK's corpus – stopwords), numbers and converted to a set of meaningful words.
- Count Vectorizer in scikit learn was used to create a bag of words model which will return binary value if that is word present in each review of the data set or not. The same will be iterative for all the words in the bag of words.
- Later I learned of something called TF-IDF, short for term frequency–inverse document frequency, is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus. It is often used as a weighting factor in information retrieval, text mining, and user modelling.
- Instead of using count vectorizer I used TF-IDF vectorizer in scikit-learn to give a vector for each review with which I performed decision tree classification.
- I used the MLP classifier from the scikit-learn's neural network library.
- The parameters for the default sklearn's MLP classifier were – hidden_layer_sizes = 100, activation function: rectified linear function, learning_rate_init = 0.001, etc.
- To reduce the time to execute the Neural Network algorithm on the entire data set, I've reduced the number of reviews to 8000 in train dataset and to 2000 reviews in test dataset.
- I initially chose to take 5000 features from the Tf-IDF vector and train it with at least one layer with 5000 nodes with sigmoid activation function on the Mamba Cluster, but three runs have gave me the same error message:

  PBS Job Id: 9999.mba-m1.uncc.edu
  Job Name: mLFirstJobScript
  Exec host: mba-c1.uncc.edu/0-3+mba-c2.uncc.edu/0-3+mba-c3.uncc.edu/0-3+mba-c4.uncc.edu/0-3
  Aborted by PBS Server
  Job exceeded its walltime limit. Job was aborted
  See Administrator for help
  Exit_status=-11
  resources_used.cput=05:17:42
  resources_used.energy_used=0
  resources_used.mem=12468316kb
  resources_used.vmem=13476264kb
  resources_used.walltime=01:00:33
- The problem that occurred was it was taking more than 1hr to run on the cluster with around 2500 hidden layer nodes which were too much.
- I had run with 5000 features and one hidden layer with 2500 nodes and the results were:

The accuracy score for the Neural Network is

0.564282141071

[[56 20 22 26 39]

 [33 22 25 15 25]

 [9 24 30 48 63]

 [14 12 36 105 178]

 [26 17 52 187 915]]

	precision	recall	f1-score	support
1	0.41	0.34	0.37	163
2	0.23	0.18	0.20	120
3	0.18	0.17	0.18	174
4	0.28	0.30	0.29	345
5	0.75	0.76	0.76	1197
avg / total	0.56	0.56	0.56	1999

- Hence, I reduced the number of features (frequent words) from the TF-IDF vector as 1000 features and chose the number of hidden layers as 1000, I got an accuracy of around 59% which was an increase from 54% using decision tree classifier.

0.590795397699

[[65 28 22 11 37]

 [31 17 24 20 28]

 [14 23 41 35 61]

 [10 20 43 94 178]

 [24 32 46 131 964]]

	precision	recall	f1-score	support
1	0.45	0.40	0.42	163
2	0.14	0.14	0.14	120
3	0.23	0.24	0.23	174
4	0.32	0.27	0.30	345
5	0.76	0.81	0.78	1197

avg / total 0.58 0.59 0.58 1999

Process finished with exit code 0

- The increase in the number of hidden layer sizes would give us an increase in the accuracy but it would take hours to run on the cluster itself. If we add another hidden layer it would take more time.
- So, the better idea would be to balance the number of features from the TF-IDF vector and the hidden layer sizes.
- The best accuracy I could get is when I took features = 500 and one hidden layer with nodes = 500, accuracy was around 61%, which was an increase.

0.611805902951

[[63 25 17 9 49]

[30 15 27 14 34]

[18 21 34 29 72]

[16 10 28 93 198]

[44 19 34 82 1018]]

	precision	recall	f1-score	support
1	0.37	0.39	0.38	163
2	0.17	0.12	0.14	120
3	0.24	0.20	0.22	174
4	0.41	0.27	0.33	345
5	0.74	0.85	0.79	1197

avg / total 0.58 0.61 0.59 1999

Process finished with exit code 0

- We should not decrease the number of features more than required as we would lose important features that would be important in classification.
- We can see that the classes 2,3 and 4 are less accurately classified than 1 and 5. Reasons:
    - 1 and 5 are extreme ratings and they would contain frequently used strong negative and positive words that could be common in almost all reviews.
    - 2,3 and 4 are moderate review ratings and they could contain a mixture of negative and positive words and it might be one of the reasons why our neural network was not able to classify those more accurately.

- A future improvement might be addition of Sentiment Analysis to the dataset as one of the features and training it along with bag of words features using an ensemble of methods like random forest , etc might improve the accuracy of the classification.