1: 4 class classification - Noisy Data

Solution:

DESCRIPTION

The data set consists of 2 dimensional points with labellings associated to each of the data points. The labels range from 0 - 3. Hence consist of four labels.

The dataset consists of 5 noisy variations of the same data set:

1. Clean Data, 0 noise
2. 10% noise in the labelling
3. 20% noise in the labelling
4. 40% noise in the labelling
5. 60% noise in the labelling

VISUALIZATION

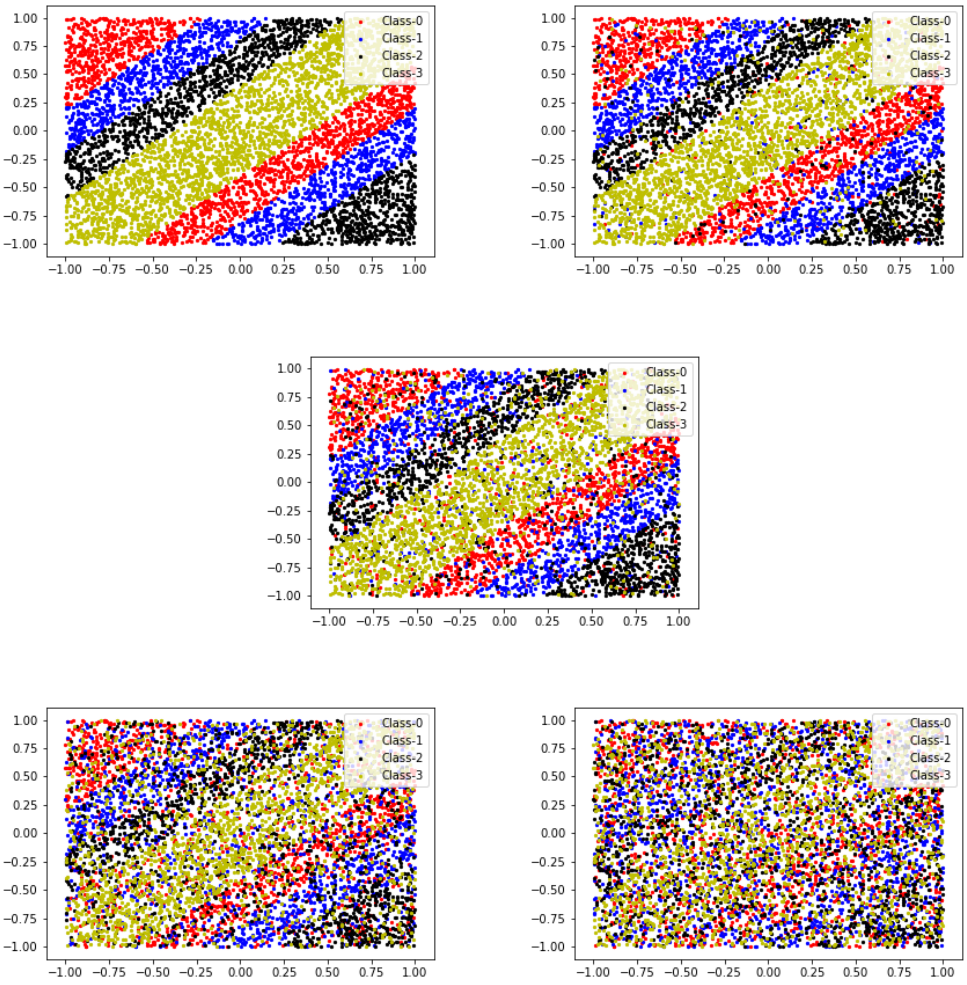

Figure 1: **Visualization of data points**:- **Row 1 (L-R)**: 0 Noise, 10% Noise, **Row 2**: 20% Noise, **Row3 (L-R)**: 40% Noise, 60% Noise

Experiments

The data was visualized using matplotlib library to see the patterns of general data points. It can be seen that there is a clear separation for non-noisy data points however the demarcation of boundaries becomes more questionable as the noise increases. At 60% it is almost impossible to construct a good separation boundary.

The following variations were trained:
1. SVM
2. Neural Network (Single hidden layer, 2 hidden layers)

The data was split into training and test data randomly on the 0% noise data set and the same splitting was done on the noisy data sets. The following variations were tried:
1. 20% test, 80% train
2. 40% test, 60% train
3. 60% test, 40% train
4. 80% test, 20% train

Apart from this 3 different SVM kernels - Linear, Poly and RBF were tried. In the neural networks, activation layers 'relu' and 'logistic' were tried

Observations

Test Data	Model	0	10%	20%	40%	60%
20%	SVM	99.68%	96.4%	95.7%	92.03%	85%
	Neural Network	99.01%	95.5%	95.8%	93.2%	78.20%
40%	SVM	99.29%	97.34%	96.71%	91.03%	82.4%
	Neural Network	98.9%	96.5%	97.65%	84%	67%
60%	SVM	99.47%	96.27%	96.7%	90.6%	75.8%
	Neural Network	99.03%	95.95%	95.15%	87.9%	64.3%
80%	SVM	99.06%	97.4%	94.9%	89.20%	73%
	Neural Network	98.60%	95.4%	92.9%	84%	54.6%

The above are the accuracy measures on held out test data. The first row shows readings for 20% held out test data, second row for 40% and so on.

The following observations were seen:

1. SVM seems to be more robust than neural networks on very noisy data sets.

2. It can be seen that, in general, the accuracy decreases as more noise is added to the training data. This is because the separation boundary becomes more difficult to obtain and is more convoluted, hence generalization is difficult. In some cases, however, the accuracy increases for some increase in noise. This may be perhaps because the boundaries obtained may be suitable for that setting of test data.

3. For linear SVM there is substantial decrease in performance, which is evident from the visualization of data set itself. The data is not strictly linearly separable.

4. Neural networks required a lot of tuning. Two variations, i.e. Single hidden layer and 2 hidden layers were tried. No substantial improvement was observed. There was very slight change in the test accuracies

5. Different values of C parameters were tried for SVM and 1×10^3 performed best in all scenarios. Performance was tested using cross-validation - 5 folds.

6. Neural network hyper-parameters were tuned using cross validation for number of layers, and hidden units. On other parameters like learning rate and regularization constant, random search first to obtain high accuracy and then a grid search on the surrounding values. Final values: 1 hidden layer, 80 hidden units, 0.01 learning rate, and 10 as the regularization constant

7. It took more time to train SVM for noisy data sets than clean. This was because clean data contained lesser support vectors than noisy and obtaining those was computationally expensive

8. As the test data size is increased we see that for clean data, there is not much change in the accuracy, since by removal of data points, if the support vectors are not removed, then we essentially get the same boundaries. However in-case of noisy data, since there more support vectors, if we remove more points, it is more probable that the support vectors are also removed during the process, hence the boundary changes.

9. Final accuracies for neural networks were obtained by training the model multiple times, then calculating labels each time, and finally taking the mode of the values.

References:

1. Code written in python notebook format with the help of matplotlib and scikit-learn libraries. Available upon request