E1 213 – Pattern Recognition and Neural Networks

Assignment # 3

Codes available at: https://github.com/vineeths96/Pattern-Recognition-3*

Vineeth S **M.Tech Artificial Intelligence** **SR No: 16543**

Note: The results summarized in the tables in this report belongs to the 'best' performance classifier. The intermediate classifier results are omitted as they add no value.

Problem 1

Problem 1 is a 2-class classification problem with 2-dimensional feature space. We have to compare SVM and neural networks. We have 2000 samples with 0%, 20% and 40% label noise.

The standard ML practice is to split the dataset into 70% training set, 15% development set and 15% testing set. However, we are not performing any industry level training and hence the development set can be avoided. Hence, I decided to split the dataset into 80% training set and 20% testing set.

SVM

The following table summarizes the performance of SVM Classifier after cross validation with various kernels on the data with different label noise.

Kernel	Label Noise	Accuracy
Linear	0%	82.75%
Polynomial	0%	87.50%
RBF Gaussian	0%	98.25%
Sigmoid	0%	43.00%
Linear	20%	64.75%
Polynomial	20%	67.00%
RBF Gaussian	20%	77.50%
Sigmoid	20%	43.50%
Linear	40%	51.25%
Polynomial	40%	54.25%
RBF Gaussian	40%	56.50%
Sigmoid	40%	43.75%

As we can observe from the table RBF Gaussian kernel outperforms other kernels in their corresponding segments. Following our intuitive prediction, the classification accuracy drops with the increase in the label noise. This can be attributed to the fact that the data may not be separable when there is label noise and to its reliance on support vectors and the feature interdependence assumption. The value of 'C' for SVM was obtained from a Gaussian process

** GitHub repository access is private as of now. Access can be made available, if necessary.*

approximation using scikit-optimize library (similar to GridSearchCV, but less expensive). The input was pre-processed and evaluated with minmax scaling, z-score normalization.

The linear kernel works fine if our dataset is linearly separable. If not, we could try SVM with RBF kernel. The downside of the latter is that its complexity grows with the size of the training set. Other kernels like polynomial kernel are rarely used due to poor efficiency.

Neural Network

The following table summarizes the performance of Neural Network Classifier after cross validation with different label noise.

Label Noise	Accuracy
0%	97.25%
20%	74.25%
40%	54.50%

The models were implemented in Tensorflow. Since this problem deals with learning a non-linear classifier a simple dense net was used with binary cross entropy loss. Again, as per our intuition, the classification accuracy drops with the increase in the label noise. Interestingly, RBF Gaussian SVM classifier outperforms neural network based classifier for all levels of noise. The input was pre-processed and evaluated with minmax scaling, z-score normalization. The value of major hyperparameters like learning rate for DNN was obtained from a Gaussian process approximation using scikit-optimize library.

The deep network model needs the architecture to be defined appropriately to perform the task efficiently. This involves choosing the number of layers (apart from input and output layers), their hidden units, kernel filters and sizes etc. The network architecture should be deep enough to extract the key essential features from the input, which would help in properly learning the characteristics for each class. At the same time, the network should not be too deep so that the gradients diminish while moving towards the input layer, which would result in improper or no learning. The actual rule of thumb to select the number of hidden layers depends on the complexity and nature of the task that we are expecting the network to do.

Problem 2

Problem 2 is a 5-class classification problem with 2-dimensional feature space. We have to compare SVM and neural networks. In each case we have 8000 samples with 0%, 10% and 25% label noise. As for the reasons explained in Problem 1, I decided to split the dataset into 80% training set and 20% testing set.

SVM

The following table summarizes the performance of SVM Classifier after cross validation with various kernels on the data with different label noise.

Kernel	Label Noise	Accuracy
Linear	0%	41.00%
Polynomial	0%	56.44%
RBF Gaussian	0%	97.19%
Sigmoid	0%	1.94%
Linear	10%	32.38%
Polynomial	10%	49.38%
RBF Gaussian	10%	86.81%
Sigmoid	10%	4.06%
Linear	25%	29.00%
Polynomial	25%	41.88%
RBF Gaussian	25%	72.75%
Sigmoid	25%	9.81%

As we can observe from the table RBF Gaussian kernel outperforms other kernels in their corresponding segments by a huge margin. Sigmoid SVM performs the worst with accuracy in single digits. Following our intuitive prediction, the classification accuracy drops with the increase in the label noise. All other general observations about SVM from Problem 1 follows here as well. We can also observe that RBF SVM is more resilient to label noise for multi class classification. The value of 'C' for SVM was obtained from a Gaussian process approximation using scikit-optimize library. The input was pre-processed and evaluated with minmax scaling, z-score normalization.

Neural Network

The following table summarizes the performance of Neural Network Classifier after cross validation with different label noise.

Label Noise	Accuracy
0%	99.12%
10%	89.94%
20%	74.75%

The models were implemented in Tensorflow. Since this problem deals with learning a non-linear classifier a dense net was used with categorical cross entropy loss. Again, as per our intuition, the classification accuracy drops with the increase in the label noise. Interestingly, neural network based classifier outperforms RBF Gaussian SVM classifier for all levels of noise, opposite to the case with Problem 1. The value of major hyperparameters like learning rate for DNN was obtained from a Gaussian process approximation using scikit-optimize library. All other general observations about neural networks from Problem 1 follows here as well.

Problem 3

Problem 3 is a 10-class classification problem with the popular MNIST dataset. We have to compare SVM and neural networks. Each sample in MNIST dataset is a hand-written digit image of dimension 28x28. There are 50000 training and 10000 test samples in MNIST dataset and an additional 12000 training and 10000 test samples in MNIST-rot dataset.

SVM

The following table summarizes the performance of SVM Classifier after cross validation with various kernels on the data without using the MNSIT-rot dataset.

Kernel	Accuracy
Linear	90.67%
Polynomial	93.17%
RBF Gaussian	94.89%
Sigmoid	86.37%

The following table summarizes the performance of SVM Classifier after cross validation with various kernels on the data using the MNSIT-rot dataset.

Kernel	Accuracy
Linear	92.70%
Polynomial	95.79%
RBF Gaussian	96.52%
Sigmoid	89.48%

As we can observe from the table RBF Gaussian kernel outperforms other kernels by a small margin. Sigmoid SVM performs the worst, but much better than the multi class classification in Problem 2. All other general observations about SVM from Problem 1 follows here as well. We can also observe that there is a slight increase in the performance of the SVMs when MNSIT-rot dataset was used as well. This might be because of the added variety of data that the rotated data brings. The value of 'C' for SVM was obtained from a Gaussian process approximation using scikit-optimize library. The input was pre-processed and evaluated with minmax scaling, z-score normalization.

Neural Network

Two models were implemented for this problem – a dense neural network (DNN) and a convolutional neural network (CNN). The models were implemented in Tensorflow and were trained with categorical cross entropy loss. Interestingly, both the neural network based classifiers outperform RBF Gaussian SVM classifier, opposite to the case with Problem 1. All other general observations about neural networks from Problem 1 follows here as well. The value of major hyperparameters like learning rate for DNN and CNN was obtained from a Gaussian process approximation using scikit-optimize library.

The DNN achieved a cross validated accuracy of 97.87% and CNN achieved a cross validated accuracy of 96.21% without using MNIST-rot dataset. The DNN achieved a cross validated accuracy of 97.96% and CNN achieved a cross validated accuracy of 97.21% using MNIST-rot dataset. There is only a slight increase in performance because I have used image data augmentation in the deep networks. Hence, when training with MNIST dataset alone, the image data is augmented with horizontal flipping, rotation, cropping etc. which simulates the data in MNIST-rot. Augmentation techniques can create variations of the images that can improve the ability of the fit models to generalize what they have learned to new images.

Though in this problem, CNN achieves slightly lesser accuracy than a DNN generally CNNs outperform DNNs in image related tasks. CNNs captures spatial dependencies much better than DNNs and hence their performance. CNNs can learn features that are invariant to their location in the image. Another advantage of CNNs are that they are able to learn with far fewer parameters. A DNN with three hidden layers with 100 units each would have around ~100 * 100 * 100 (= 10^6) parameters where as a CNN with three hidden layers with 5x5 kernels would have around ~25+1 + 25+1 + 25+1 (=78) parameters (the +1 is for the bias term).

Problem 4

Problem 4 is a 2-class classification problem with the fMRI dataset. We have to compare SVM and neural networks. The data consists of (pre-processed) fMRI recordings of 34 patients with Alzheimer's disease and 47 normal subjects in rest state. The problem is to classify a fMRI recording as healthy or not. There are two different parcellation of the same recordings. As for the reasons explained in Problem 1, I decided to split the dataset into 80% training set and 20% testing set.

SVM

The following table summarizes the performance of SVM Classifier after cross validation with various kernels on the data.

Kernel	**Accuracy** (tc_rest_aal**)**	**Accuracy** **(**tc_rest_power**)**
Linear	58.82%	54.27%
Polynomial	47.06%	43.67%
RBF Gaussian	47.06%	44.56%
Sigmoid	47.06%	43.67%

This was the hardest problem in this assignment. As we can observe the linear kernel performs better than other kernels, relatively. But for a two-class problem, an accuracy around 50% is no better than random guessing. Gaussian SVM, Polynomial SVM, and Sigmoid SVM performs worse than this. Few researches also show that SVMs are not efficient if the number of features is very huge in number compared to the training samples – which clearly is the case here. I tried with both the parcellations but neither performed decent. I tried combining the two parcellations with subsampling and then using a Hadamard product but it did not perform well. All other general observations about SVM from Problem 1 follows here as well. The value of 'C' for SVM was obtained from a Gaussian process approximation using scikit-optimize library. The input was pre-processed and evaluated with minmax scaling, z-score normalization – but neither did have a significant impact in the performance.

Neural Network

The following table summarizes the performance of Neural Network Classifier after cross validation with different architecture.

Architecture	**Accuracy** **(**tc_rest_aal**)**	**Accuracy** **(**tc_rest_power**)**
DNN	61.18%	60.57%
CNN	68.21%	59.16%
LSTM	65.21%	61.39%

Numerous models were implemented for this problem – dense neural networks (DNN), convolutional neural networks (CNN), VGG, ResNet50, and a LSTM based recurrent network. The models were implemented in Tensorflow and were trained with binary cross entropy loss. As mentioned previously, this was the hardest problem in this assignment. I personally felt that the number of data samples was extremely low for a deep network to get trained properly. All the networks, I tried – even with the smallest one – suffered heavily from overfitting. Within few epochs, the training accuracy was upto 100% and validation accuracy was around 60%. Still the neural network based classifiers outperform SVM classifier by a small margin. All other general observations about neural networks from Problem 1 and about CNNs from Problem 3 follows here as well. The value of major hyperparameters like learning rate was obtained from a Gaussian process approximation using scikit-optimize library.

Though data augmentation should not be used in this case – since the data has both spatial and temporal dependencies – I tried it to explore what happens, but the performance of the networks dropped. I could have tried with an ensemble approach with models for two parcellations separately, with voting mechanism but could not do due to computational restrictions.

Problem 5

Problem 5 is a 5-class classification problem with the epileptic seizure dataset. We have to compare SVM and neural networks. From the EEG recording, a 178-dimensional feature vector is extracted. The data consists consist of 11500 training samples where each is a 178-dimensional feature vector. As for the reasons explained in Problem 1, I decided to split the dataset into 80% training set and 20% testing set.

SVM

The following table summarizes the performance of SVM Classifier after cross validation with various kernels on the data.

Kernel	**Accuracy**
Linear	24.91%
Polynomial	24.91%
RBF Gaussian	54.57%
Sigmoid	26.45%

As we can observe from the table RBF Gaussian kernel outperforms other kernels by a decent margin. Since this is a five-class classification random guessing would have an accuracy around 20% and hence we can say RBF SVM performs satisfactorily. All other general observations about SVM from Problem 1 follows here as well. The value of 'C' for SVM was obtained from a Gaussian process approximation using scikit-optimize library. The input was pre-processed and evaluated with minmax scaling, z-score normalization.

Neural Network

Two models were implemented for this problem – a dense neural network (DNN) and a recurrent neural network (LSTM network). The models were implemented in Tensorflow and were trained with categorical cross entropy loss. Interestingly, both the neural network based classifiers outperform RBF Gaussian SVM classifier. All other general observations about neural networks from Problem 1 follows here as well. The value of major hyperparameters like learning rate for DNN and LSTM was obtained from a Gaussian process approximation using scikit-optimize library. Two Bidirectional LSTM layers were used to capture the temporal dependencies in EEG signal.

The DNN achieved a cross validated accuracy of 55.87% and LSTM achieved a cross validated accuracy of 87.60%.