MACHINE LEARNING

Assignment 4

Mehak
2018MCS2143

Part 1 Non Competitive Part

A. SVM + PCA

Before applying PCA, there were two different ways of scaling the data. One was to just divide all the entries by 255 (effectively bringing all entries between 0 & 1) or we could normalize the data (zero mean-unit variance.) I have used scaling between 0-1.

Libraries Used:

PIL for Image Reading
Numpy and Pandas for array manipulation
Sklearn for PCA
Libsvm for SVM

Pre processing:

1. Converted RGB images to grayscale images
2. Normalization by dividing by 255 to convert the values to [0,1] range

Data generation : Include all ones data and zeros will be included as twice the number of ones . All possible combination of these zeros and ones are included in the train data set.

For Linear Kernel:

Training Set F1 score

[0.98657721 0.985112333]

Testing Set F1 score

[0.069404231]

<u>For Gaussian Kernel:</u>

Training Set F1 score

[0.99771569 0.99698902]

Testing Set F1 score

[0.071466177]

For finding out the optimal C value, I used the training data for training while the other validation data was used for cross validation.

<u>Cross Validation for linear kernel:</u>

I tried out C values: 0.01, 0.1, 1,5, 10 out of which, C=0.01 gave the best F1 score of 0.069404.

<u>Cross Validation for gaussian kernel:</u>

I tried out C values: 0.01, 0.1, 1,5, 10 and gamma values: 0.01, 0.05,0.1,0.3,0.5 out of which, C=0.01 and gamma value 0.05 gave the best F1 score of 0.071466.

<u>Observations:</u>

The PCA performed on the 50 episodes is not able to detect the main features required for prediction i.e. the ball object due to which it was observed no matter how much we tune the hyper parameters, the model is not able to make good prediction for the cases where reward is predicted to be one and hence F1 score is coming out to be very poor.

B. CNN

I have used Keras framework for CNN.
<u>Libraries used:</u>

PIL for Image reading
Keras for CNN
numpy/pandas for array manipulation
Gzip and pickle for loading dumping data

Preprocessing:

1. Converted RGB images to grayscale images
2. Normalization by dividing by 255 to convert the values to [0,1] range

To test out the implementation without having to submit the labels again and again, I used a validation set and during training, validation accuracy was calculated at each epoch.

I tried varying kernel size and stride but increasing these parameters resulted in the drop in validation accuracy.
Different experimentations with CNN:

(i) Changing number of kernels
1st conv. Layer -> 64 kernels
2nd conv. Layer -> 128 kernels

Training accuracy = 98.79

Validation accuracy = 96.99

(ii) Changing number of neurons in the hidden layer
Initially 2048 -> changed to 4096
Training accuracy = 95.89
Validation accuracy = 93.51

(iii) Changing the filter size
Initial filter size (3,3) -> changed to (5,5)
Training accuracy = 97.44
Validation accuracy = 94.33

Best model selected according to the validation data accuracy:

```
ConvNet(
(conv): Conv2d(1, 32, kernel_size=(3,3), stride=(2, 2))
(mp): MaxPool2d(kernel_size=(3, 3), stride=(2, 2))
(conv): Conv2d(1, 64, kernel_size=(3,3), stride=(2, 2))
(mp): MaxPool2d(kernel_size=(3, 3), stride=(2, 2))
(fc1): Dense(2048)
)
```

Learning Rate: Adaptive(initially 0.01)
Epochs: 350
Batch size: 50

Training accuracy = 99.57 Validation accuracy = 97.45

Testing set F1 score:
[0.45804]

Observations:

CNN gave the better F1 scores, since the training data was images, this sort of result was expected as CNNs are able to utilise feature locality.

Part 2 Competitive Part

I have used Keras framework for CNN.

Libraries used:

PIL for Image reading
Keras for CNN
numpy/pandas for array manipulation
Gzip and pickle for loading dumping data

Preprocessing:

1. Converted RGB images to grayscale images
2. Normalization by dividing by 255 to convert the values to [0,1] range

To test out the implementation without having to submit the labels again and again, I used a validation set and during training, validation accuracy was calculated at each epoch.

Architecture:

```
ConvNet(
(conv): Conv2d(1, 32, kernel_size=(3,3), stride=(2, 2))
(mp): MaxPool2d(kernel_size=(3, 3), stride=(2, 2))
(conv): Conv2d(1, 64, kernel_size=(3,3), stride=(2, 2))
(mp): MaxPool2d(kernel_size=(3, 3), stride=(2, 2))
(drop): Dropout(0.25)
(fc1): Dense(2048)
(LR): LeakyReLU
(drop): Dropout(0.5)
)
```

Batch Size: 50
Epochs: 500 (1 for each episode)
Learning Rate: Adaptive(initially 0.01)

Link for the saved model:

https://drive.google.com/open?id=1-0ou4OXjp5GnL1vE--Rss2re06nTKHWB

Testing set F1 score

[0.56404]

I have used google colab for training the network and predicting the labels.