COL774 - Assignment 4

Ankit Solanki
2016CS50401

April 27, 2019

1 Part-A: Non-competitive

1.1 PCA-SVM

```
[#] Kernel -> Linear
[>] Best Parameters
[*] C: 10.0
[-]
[-] Ratio of train to test: 9
[*] Accuracy on train set: 0.93942
[*] Accuracy on test set: 0.34160
```

```
[#] Kernel -> RBF
[>] Best Parameters
[*] C: 5.0
[*] Gamma: 0.01
[-]
[-] Ratio of train to test: 9
[*] Accuracy on train set: 0.91381
[*] Accuracy on test set: 0.10500
```

RBF takes too much time to train, and it overfits. Linear kernel generalises well.

1.2 CNN

```
[#] Model defined [#]

ConvNet(
  (layer1): Sequential(
    (0): Conv2d(15, 32, kernel_size=(3, 3), stride=(2, 2))
    (1): MaxPool2d(kernel_size=(2, 2), stride=2, padding=0, dilation=1,
        ceil_mode=False)
```

```
  )
  (layer2): Sequential(
    (0): Conv2d(32, 64, kernel_size=(3, 3), stride=(2, 2))
    (1): MaxPool2d(kernel_size=(2, 2), stride=2, padding=0, dilation=1,
    ceil_mode=False)
  )
  (fc1): Linear(in_features=6912, out_features=2048, bias=True)
  (fc2): Linear(in_features=2048, out_features=2, bias=True)
)

[*] Ratio of train to test: 9
[*] Accuracy on train set: 0.96862
[*] Accuracy on test set: 0.44617
```

2 Part-A: Competitive

Tried a deep learning model: But it took too much time to train, and couldn't get the results on time. So, not submitted anything, but the output from a randomized algorithm.

```
 # Tried Deep learning model
 # Used conv nets from Pytorch:


# feature extraction from here
 self.layer1 = nn.Sequential(
     nn.Conv2d(15, 32, kernel_size=(3,3), stride=2),
     nn.BatchNorm2d(32),
     nn.ReLU(),
     nn.Dropout(0.5),
     nn.MaxPool2d(kernel_size=(2,2), stride=2))
 self.layer2 = nn.Sequential(
     nn.Conv2d(32, 64, kernel_size=(3,3), stride=2),
     nn.BatchNorm2d(64),
     nn.ReLU(),
     nn.Dropout(0.4),
     nn.MaxPool2d(kernel_size=(2,2), stride=2))

 # apply fully connected as nn
 self.fc1 = nn.Linear(256, 2048)
 self.do1 = nn.Dropout(0.4)
 self.fc2 = nn.Linear(2048, 512)
 self.do2 = nn.Dropout(0.4)
 self.fc3 = nn.Linear(512, 256)
 self.do3 = nn.Dropout(0.4)
```

```
self.fc4 = nn.Linear(256, 64)
self.fc5= nn.Linear(64, num_classes)
```