SPR Final Proj

Param Dave

December 2022

1 SVM

Here we first apply PCA and MDA on the MNIST Dataset in order to reduce the number of features in the data. We use a similar method as done in project 1 for the face dataset. We use LIBSVM, an package in python to perform SVM. Firts we import the data using PyTorch's method and feed it into the package. The following table shows the results of algorithm.

Method	Data Preprocessing	Accuracy
Linear SVM	PCA	66
Polynomial SVM (d=2)	PCA	50
Polynomial SVM (d=4)	PCA	30
Polynomial SVM (d=6)	PCA	27
Kernel SVM (g = 1000)	PCA	11
Kernel SVM (g = 10)	PCA	11
Linear SVM (g = 1000)	MDA	22
Polynomial SVM (d = 2	MDA	11.35
Polynomial SVM (d = 4	MDA	11.35
Polynomial SVM (d = 6	MDA	11.35
Kernel SVM (g = 1000)	MDA	89
Kernel SVM (g = 10)	MDA	87

We can conclude that the SVM with PCA works fine for Linear SVM and Polynomial SVM while it deteriorates for RBF SVM. Similarly, SVM with MDA only works with RBF SVM and fails to give satisfactory results on others. The accuracies are shown in figure 1,2,3 and 4

2 Logistic Regression

For Logistic Regression for multiple epochs, we compute the posteriors using the equation given and compute loss thereafter. We compute the gradient and use the gradient using training samples and test data to update our thetas used to calculate posterior. This was the training round. In inference, we calculate posterior and compute the labels. https://www.overleaf.com/project/63999a2b4882752d375c6ec0 Finally running the algorithm with PCA we get 0.88 and 0.78 accuracies in

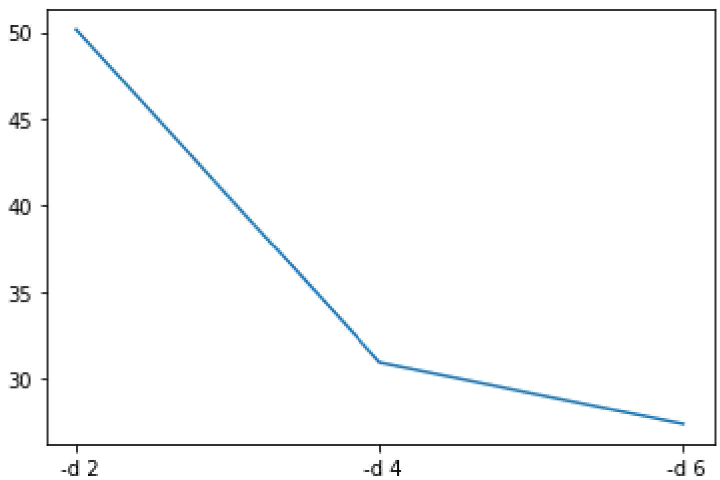

Figure 1: Polynomial SVM with PCA

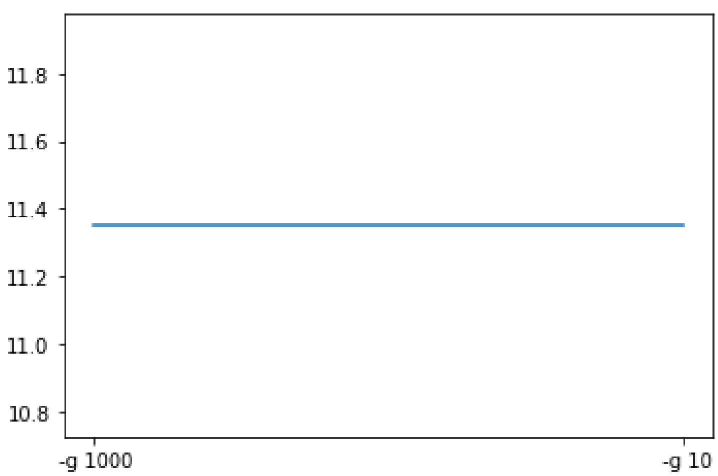

Figure 2: RBF with PCA

training and testing. For MDA we get 0.78 and 0.79 accuracies in training and testing.

Loss graphs for training are shown below

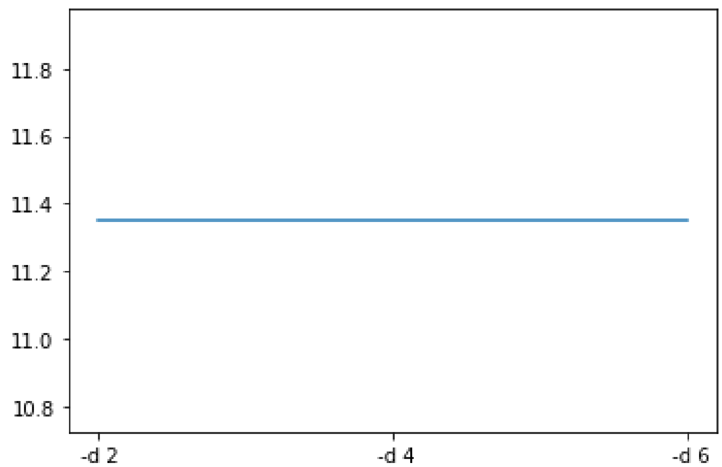

Figure 3: Polynomial SVM with MDA

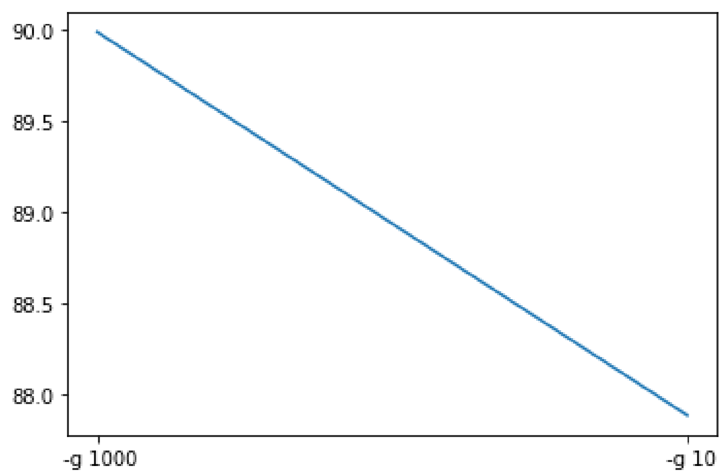

Figure 4: RBF with MDA

3 Deep Learning

3.1 MNIST Dataset

To train a network on MNISt we use a network similar to lenet as shown in the figure below. The accuracy of the model comes to around 0.98/1 for a batch size of 64 and a learning rate of 0.01. The learning rate does play a part in optimization. We found that the high value of the learning rate leads to no convergence and similarly too low value of the learning rate leads to slow

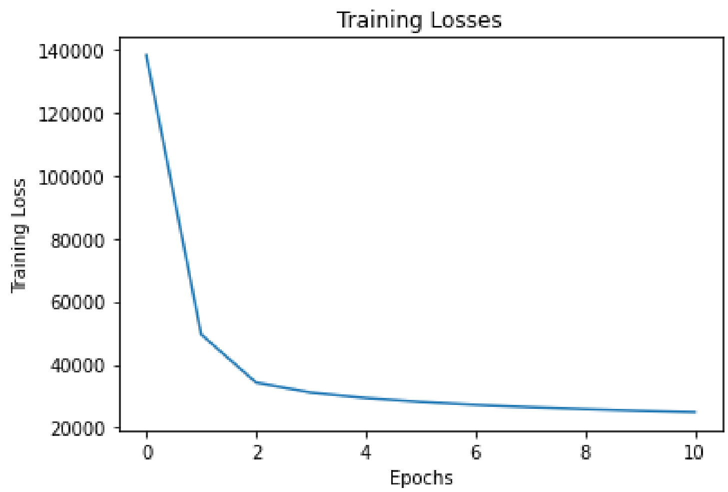

Figure 5: Logistic Regression Loss with PCA

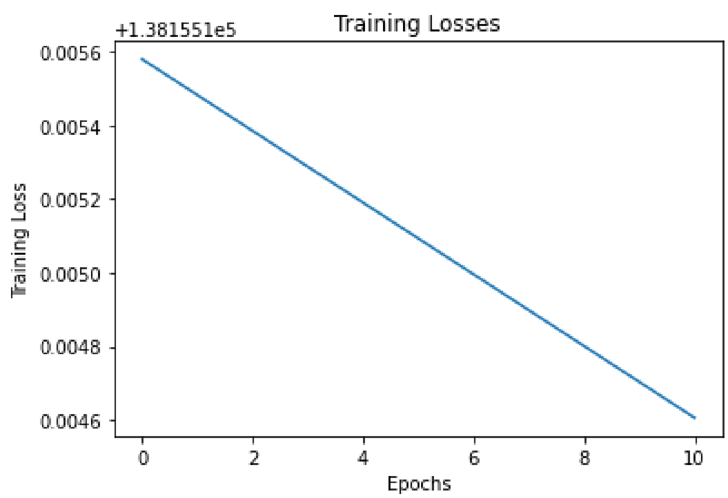

Figure 6: Logistic Regression Loss with MDA

convergence.

3.2 Monkey Dataset

We use a custom dataset for training as shown in figure below.

The accuracy comes to around 0.11/1.

To improve the accuracy we use tranfer learning. For fine-tuning and transfer learning we take ResNet-18. It gives us an accuracy of 0.97/1 for transfer

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1            [-1, 6, 28, 28]             156
       BatchNorm2d-2            [-1, 6, 28, 28]              12
               ELU-3            [-1, 6, 28, 28]               0
         MaxPool2d-4            [-1, 6, 14, 14]               0
            Conv2d-5           [-1, 16, 10, 10]           2,416
       BatchNorm2d-6           [-1, 16, 10, 10]              32
               ELU-7           [-1, 16, 10, 10]               0
         MaxPool2d-8             [-1, 16, 5, 5]               0
            Linear-9                  [-1, 120]          48,120
              ELU-10                  [-1, 120]               0
           Linear-11                   [-1, 84]          10,164
              ELU-12                   [-1, 84]               0
           Linear-13                   [-1, 10]             850
================================================================
Total params: 61,750
Trainable params: 61,750
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.00
Forward/backward pass size (MB): 0.16
Params size (MB): 0.24
Estimated Total Size (MB): 0.40
----------------------------------------------------------------
```

Figure 7: Deep Learning Architecture to train on MNIST Dataset

```
custom_model(
  (model): Sequential(
    (0): Conv2d(3, 8, kernel_size=(5, 5), stride=(1, 1))
    (1): ReLU()
    (2): Conv2d(8, 16, kernel_size=(5, 5), stride=(1, 1))
    (3): ReLU()
    (4): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (5): Conv2d(16, 32, kernel_size=(5, 5), stride=(1, 1))
    (6): ReLU()
    (7): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (8): Conv2d(32, 64, kernel_size=(5, 5), stride=(1, 1))
    (9): ReLU()
    (10): Conv2d(64, 128, kernel_size=(5, 5), stride=(1, 1))
    (11): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (12): Flatten(start_dim=1, end_dim=-1)
    (13): Linear(in_features=61952, out_features=1024, bias=True)
    (14): ReLU()
    (15): Linear(in_features=1024, out_features=512, bias=True)
    (16): ReLU()
    (17): Linear(in_features=512, out_features=10, bias=True)
  )
)
```

Figure 8: Deep Learning Architecture to train on Monkey Dataset

learning and accuracy of 0.99/1 for fine-tuning for 3 epochs.