Consolidated Report

Prithvi Poddar 17191

Note: Report 2 and 3 are the reports on CNN and autoencoder respectively. Those were made on Google Colab as it provides GPU facility. Thus, the .ipynb files regarding the same might not run on the local jupyter notebook. Although I've added those python notebooks in the report directory, I'll leave a link to the Colab notebook here as well . You can visit the link to view the code in Colab.
To run the code, you'll have to make a copy of it in your own drive as my codes are in view only format.

Report 2 (CNN) link:
https://colab.research.google.com/drive/1GxHUK_lYsqA3YGl_Ts4SltuB5q7tVE4L?usp=sharing

Report 3 (Autoencoder) link:
https://colab.research.google.com/drive/1_qcXU7SAz_ZQjyYU9bw3jxThv6E6d30V?usp=sharing

Starting with Report-1 now.

<u>Report-1 (Neural network)</u>
Files: Prithvi_Poddar_17191_Report_1.ipynb and Prithvi_Poddar_17191_Report_1.pdf

<u>Part -1:</u>
I have defined the network as a class in the python file named **network.py** in the parent directory. The code to load the data is in **data_loader.py** in the parent directory.
In part 1, I start off with using the neural network with normal training and testing data splits and do the hyper parameter tuning for the network. At the end, we find out the the best parameter choices are:
Weight_initializer = default
Cost = CrossEntropy
Activation = sigmoid
Epochs = 50
Batch_size = 10
learning_rate = 0.1
Regularization_parameter = 7.0

With this we got an accuracy of around 95% on the training data and 88% on the testing data that was provided.

Finally we used **10-fold cross validation** and got training accuracy of 97% and testing accuracy of 96% on the given data.
Then I ran the model on the entire MNIST dataset with cross validation, getting an accuracy of 95% in both training and testing sets.

Part-2 (PCA data):
Here we ran the same model on the PCA data provided and ended up getting a maximum accuracy of 88% on the test set and 87% on training set. This showed that the PCA data wasn't as good as the original data.

P.S. I'd suggest you to look at the python notebook for report 1 as the pdf is too long because the pdf contains the outputs in their entirety. The python notebooks have the outputs too but in smaller windows and hence is easy to follow.

Report-2 (CNN):

Files: Prithvi_Poddar_17191_report_2_CNN.ipynb and Prithvi_Poddar_17191_report_2_CNN.pdf

Colab link:
https://colab.research.google.com/drive/1GxHUK_lYsqA3YGl_Ts4SltuB5q7tVE4L?usp=sharing

In this report, for the architecture of the CNN, I took help form
https://www.kaggle.com/cdeotte/how-to-choose-cnn-architecture-mnist
as this site has already done some of the experiments to find the optimal parameters. I avoided doing these experiments manually as the CNN in itself takes lot of time to train and hence it would be a highly time taking process to train the model for each parameter.
The architecture of the model is:

```
Network_CNN(
  (conv1): Conv2d(1, 32, kernel_size=(5, 5), stride=(1, 1))
  (conv2): Conv2d(32, 64, kernel_size=(5, 5), stride=(1, 1))
  (fc1): Linear(in_features=1024, out_features=500, bias=True)
  (dropout1): Dropout(p=0.4, inplace=False)
  (fc2): Linear(in_features=500, out_features=10, bias=True)
)
```

At the end of the report, I have also derived the outputs of individual convolutional filters of the CNN to look at what kind of information does the CNN learn.

Report-3 (CNN):

Files:
Prithvi_Poddar_17191_Report_3_Autoencoders.ipynb and
Prithvi_Poddar_17191_Report_3_Autoencoders.pdf
Colab link:
https://colab.research.google.com/drive/1_qcXU7SAz_ZQjyYU9bw3jxThv6E6d30V?usp=sharing

In this, I built a sparse autoencoder inspired by the notes of Andrew NG on sparse autoencoder.
The model has the following parameters:

```
SparseAutoencoder(
  (en1): Linear(in_features=784, out_features=256, bias=True)
  (en2): Linear(in_features=256, out_features=128, bias=True)
  (en3): Linear(in_features=128, out_features=64, bias=True)
  (en4): Linear(in_features=64, out_features=32, bias=True)
  (en5): Linear(in_features=32, out_features=16, bias=True)
  (de1): Linear(in_features=16, out_features=32, bias=True)
  (de2): Linear(in_features=32, out_features=64, bias=True)
  (de3): Linear(in_features=64, out_features=128, bias=True)
  (de4): Linear(in_features=128, out_features=256, bias=True)
  (de5): Linear(in_features=256, out_features=784, bias=True)
)
```

At the end, I've regenerated 20 input images from the validation set, using the autoencoder.