Assignment Report

Problem 1: Perceptron

Q1. In how many steps perception learning algorithm will converge.

A1. The algorithm converged in 9 steps, assuming that we define steps as the number of times the model sees any data point.

Q2. What will be the final decision boundary? Show step-wise-step update of weight vector using computation as well as hand-drawn plot.

A2. The Weight vector of the final decision boundary is W=[0.9, -0.5] , step wise vector and the hand drawn plot is shown in the pdf document “Perceptron.pdf”.

Assumptions while solving the problem:

It was assumed that the bias weight was not taken into consideration since the initial given weight vector had the same dimension(R^2) as the input vector X(R^2). And bias wasn't mentioned.

But if we wanted to consider bias term very little modification is required i.e first we would have to initialize a bias, then in the algorithm while making a prediction instead of comparing w*x >=0 , we would have to change it to w*x>=b. And finally whenever a data point was misclassified, update the bias term along with the weight vector.

Problem 2 : Learning to implement Neural Network

Steps taken:

- **Division Of Data**
    - Provided Train Data - 1000 images
        - Used as Train Data - 800 images
        - Used as Validation Data - 200
    - Provided Validation Data - 178 images
        - Was used as Test data
    - These steps were initially used for selecting the model
    - Once the model was selected it was trained on the Provided Trained Data and evaluated on the Provided Validation Data
- **First Model** - A simple fully connected neural network was created and used.
- Experiments were conducted for different numbers of hidden layers(from 1 - 4) along with experiments with their respective number of neurons in the layers (within this list [32, 64, 128, 256, 512]) .
    - The model with the combination of (1024(Input Size) ->512->256->10) was chosen based on the validation loss.

Steps Taken to Improve Generalization:

- Experimented with Image Augmentation
- Saved Models with best validation loss
- Experimented with Batch Normalization and Dropout (together and individually) as both have regularization effects
- Experimented with L1 and L2 regularization for weights.

Observations:

- L2 Regularization performed better than L1.

- The combination of BatchNorm and Dropout were helpful for generalization.
- Image Augmentation did not help much
    - Possible Reasons could be
        - Very Few Augmentations possible so not much helpful variations (can't use big rotations or flips or shifting)

Results:

For Final Model :

Accuracy on Provided Training Dataset - 1.0

Accuracy on Provided Validation Dataset - 0.985

Steps For Running The Model on Test Set on Your End :

1. Run all the cells till the section " Augmented Data Pipeline"
    a. Change the value of the variables "train_dir" and "val_dir" with the paths of the directory containing training and validation images respectively.
2. Then jump to the last section of the notebook "For Evaluation On Test Side"
    a. Here insert the path to the test directory
3. Then run the cell after that

Problem 3: Chart Image Classification using CNN

Steps taken:

- **Division of Data:**
  - The Provided Training data was split into training and validation in the ratio of 80:20 for each class to maintain the balance.
  - The Provided Test Data was not used because the labels were not provided for it.
- **Models**
  - 2 Layer CNN Model and Accuracy and Loss were calculated and plotted
  - Pretrained Network - A pre trained model of VGG16 network was used and Accuracy and Loss were calculated and plotted.
    - Model - (VGG16 -> 64 -> 5)

Results:

- 2 Layer CNN Model -
  - Training Accuracy - 1.0
  - Validation Accuracy - 0.985
- Pre Trained Model -
  - Training Accuracy - 1.0
  - Validation Accuracy - 1.0

Observations:

- The pretrained model reached a higher validation accuracy in fewer epochs compared to the 2 Layer CNN Model.