Eeshaan Ravi Tivari (2019465)
Manvi Goel (2019472)

README
Assignment 1

CSE641- Deep Learning

Question 1

Content

1. Perceptron Class
   It implements the perceptron training algorithm. It also plots the decision boundaries.

2. Training data for AND, OR, and NOT
   It is given to the method manually

3. Training using PTA
   It constructs a perceptron object and trains the object using the training data. It finally outputs the new weights and decision boundary at every step of weight updates. (1a)

4. Training for XOR using PTA
   It constructs a perceptron object and trains for XOR data. It breaks out of the loop when the weights start repeating. (1c)

Libraries Used

1. Numpy for arrays
2. Matplotlib for plotting

Instructions

1. Make an array for the data with both X and y.

```
AND_data = np.array([[0,0,0],[0,1,0],[1,0,0],[1,1,1]])
OR_data = np.array([[0,0,0],[0,1,1],[1,0,1],[1,1,1]])
NOT_data = np.array([[0,1],[1,0]])
```

2. Initialize weights and create and perceptron class object.

3. Train the algorithm and plot the graphs using the perceptron class object created above.

```
weights= np.array([4,5,-1])
pta_and = Perceptron(weights,max_epoch=100)

flag = pta_and.fit(train_x,train_y)

if not flag:
    print("untrainable")
    print("steps for weight repetetion: ",pta_and.steps)

else:
    predictions = pta_and.predict(test_x)

    print("epoch required to converge: ",pta_and.steps)
    print("final weights: ",pta_and.wts)

    print('predicted:',predictions,'\nactual:',test_y)
```

Question 2

Content

1. Adaline class
   It contains the Adaline update, predict and fit method.

2. Madaline Class
   It contains the madaline fit and predict method. It makes use of adaline objects for its hidden layers

3. Generating training data
   It uses if-else conditions to generate a suitable dataset and plots it.

4. Running madaline algorithm for training data
   It creates a madaline object and trains for the dataset until it converges. It prints the final accuracy. (2a and 2b)

Libraries Used

1. Numpy for arrays
2. Matplotlib for plotting training data
3. Itertools for generating powerset

Instructions

1. Construct a Madaline object with the number of inputs, number of hidden neurons, learning rate, and threshold value.
2. Fit the model for the training set.
3. The output would be the number of training points correctly classified and the weights for the hidden layers

```
n = 2
h = 21
s = trainX
z = trainy
np.random.shuffle(s)
np.random.shuffle(z)
net = Madaline(n, h, 0.01, 0.5)
net.fit(s, z, True)
```

Question 3

Content

1. MLP class
   Class in toolkit.py to implement the multi-layer perceptron training algorithm.

2. Experimentation models
   It contains various experiment models for getting the best configuration using the gradient descent optimization technique.

3. Optimizer model
   Using the best model configuration achieved using gradient descent, more models are made and trained using different optimization techniques.

4. Accuracy plots and values for train, validation, and test sets

Libraries Used

1. Numpy for arrays
2. Matplotlib for plotting training data
3. Keras for loading dataset and to convert class label to binary encoded matrix
4. Pickle for model saving
5. Sklearn for accuracy metric

Instructions

1. Construct an MLP class object with the number of inputs, number of hidden neurons, learning rate, batch size and optimizer.
2. Fit the model for the training set.
3. The fit() method will return epochwise validation and train accuracy.
4. To make prediction pass the test samples into predict() function of MLP object which will return the predicted output.
5. To get the accuracy sklearn's accuracy_score() is used.

```
# activation function: tanh
epochs = 15

# MyNeuralNetwork(N_layers, Layer_sizes, activation, learning_rate, weight_init, batch_size, num_epochs)
myNet = MyNeuralNetwork(4, [784, 256, 64, 10], "tanh", 0.001, "random", X_train.shape[0], epochs,optimizers = "GD")
validationLoss, trainLoss = myNet.fit(X_train, y_train, X_validation, y_validation)
y_pred = myNet.predict(X_test)


y_test2 = np.argmax(y_test, axis = 1)
print("test accuracy:",100*accuracy_score(y_pred, y_test2))
```

Contribution

Eeshaan - Perceptron training, MLP and optimizers implementation
Manvi - madaline training, MLP and optimizers implementation