AIML - CS 335

Lab 12: Feed-Forward Neural Network

27 October 2022

Lecturer: Abir De

Important: Please read the instructions mentioned in the questions carefully. We have provided boilerplate code for each question. Please ensure that you make changes in the areas marked with TODO. Please read the comments in the code carefully for detailed information regarding input and output format.

1 Feed-Forward Neural Network

In this problem, we will implement a feed-forward network for the classification of two datasets.

1. **MNIST dataset**: Predict a digit (from 0 to 9) from an image of the digit. Each data point in the MNIST dataset is a 28x28 2D array.
2. **Flowers dataset**: Predict flower type (labeled from 0 to 4) from a feature vector of length 2048.

In `assignment.py`, complete the functions where `TO DO` has been marked. These are:

1. The class `FlattenLayer` with forward and backward functions. This class converts a multi-dimensional array into 1D vector.
2. The class `FCLayer` with init, forward and backward functions. This class implements a fully connected layer in a feed-forward network.
3. The class `ActivationLayer` with forward and backward functions. This class invokes a non-linear activation layer that applies the activation function given as argument on the inputs.
4. Three different activation functions can be passed as argument to the `ActivationLayer` class. These are sigmoid, tanh and ReLU. Implement the forward pass for them in functions `sigmoid()`, `tanh()` and `relu()` respectively. Implement the backward pass for them in the functions `sigmoid_prime()`, `tanh_prime()` and `relu_prime()` respectively.
5. The class `SoftmaxLayer` with forward and backward functions. This class applies softmax function on the inputs.
6. Use cross-entropy loss to train your network. Implement the forward and backward pass in the functions `cross_entropy()` and `cross_entropy_prime()`.

7. Finally, implement the function `fit()` to design and train a feed-forward network for each dataset. Save your trained model/weights in the 'models' directory.

8. In the `predict()` function, load your trained model/weights, compute the predicted labels for the test data and return it. We will be calling this function from the auto grader with a test set that has not been shared with you, so please ensure your code is fully functional and this function is returning the desired output.

In addition to the above functions, you have two additional functions that you can utilize in your network training.

1. `preprocessing()` can be used to perform pre-processing steps like data normalization.

2. `split_data()` can be used to split the provided training data into two disjoint sets, where one can be used for training and the other for validation.

Design and train a separate feed-forward network for each of the two datasets. Save your trained model/weights in the 'models' directory of your submission.

2 Submission instructions

Make changes only in the places mentioned in comments. Do not modify anything else. Finally, place your solution inside a folder named `<ROLL_NUMBER>_L12` and compress it to a tar file named `<ROLL_NUMBER>_L12.tar.gz` using the command
`tar -zcvf <ROLL_NUMBER>_L12.tar.gz <ROLL_NUMBER>_L12`
Submit the tar file on Moodle. The directory structure should be -

```
<ROLL_NUMBER>_L12
| - - - - data
| - - - - models
| - - - - -|- - - - mnist_model.pkl
| - - - - -|- - - - flowers_model.pkl
| - - - - assignment.py
```

Replace `ROLL_NUMBER` with your own roll number. If your Roll number has alphabets, they should be in *"small"* letters.