ECE542 – Backpropagation Assignment

The goal of this project is to gain a better understanding of backpropagation. At the end of this assignment, you would have trained an MLP for digit recognition using the MNIST dataset. The code requires a Python3 environment with Numpy and Matplotlib. You can find the MNIST dataset used for this project with the provided files. Follow the steps below after you have become familiar with the files provided.

Check the Data:

To ensure you're able to load the data correctly, run `experiment/mlp.py`:

```
python mlp.py –input
```

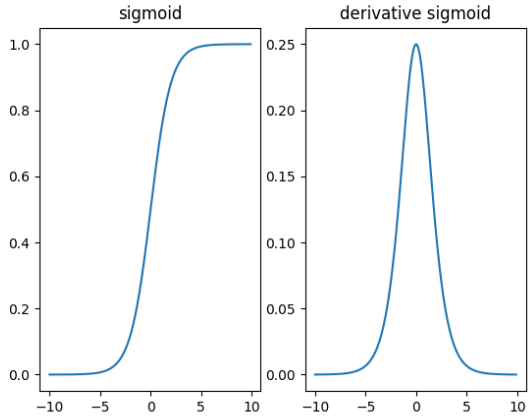
If everything is correct, you will see the output shows 3,000 samples in training, 10,000 samples in validation and 10,000 samples in testing.

Implement the Sigmoid Function:

Complete the implementation of the sigmoid function and derivative of sigmoid in `src/activation.py`. The function `sigmoid(z)` returns the value of the sigmoid function and `sigmoid_prime(z)` returns the value of the derivative. Test the sigmoid function by running:

```
Python mlp.py –sigmoid
```

If everything is correct, you will the following plots:



Implement the Backpropagation:

Follow the comments in `src/bp.py` to complete the implementation of backpropagation. You do not have to exactly follow the comments, but make sure the function returns each variable in the correct

format. To make things easier, you can use a very small network to test your implementation by modifying the input in the `experiment/mlp.py` file.

Use gradient check[1] to check whether your implementation is correct or not. We provide a simple implementation of gradient check for one of the weights which you can try by running:

```
python mlp.py –gradient
```

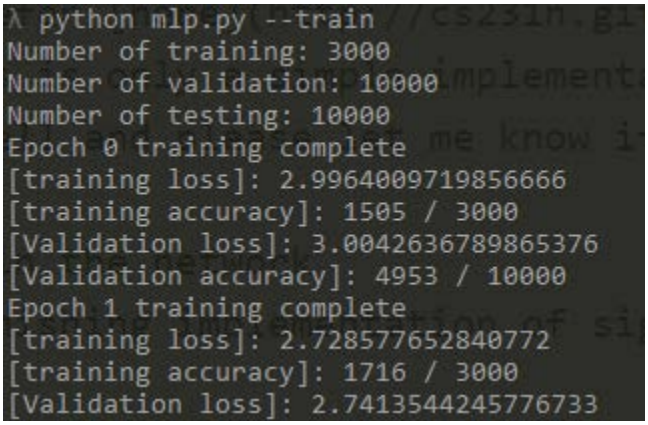
You should check the gradient for several weights before training the network. You can do this by changing the values of `layer_id`, `unit_id` and `weight_id` in `mlp.py`. You are welcome to modify the script as needed to make these checks easier for you.

Train the Network:

Next, you can train the network by running:

```
python mlp.py –train
```

You should see something like this:



You can play with the hyperparameters (number of layers, number of hidden units, learning rate, etc.) to get better results. Record the learning curves for both training and validation sets. You will need them as part of your report.

Deliverables:

You will have three deliverables: (1) your source code, (2) written report, and (3) your predictions of the test set. There will be a separate link for the predictions in order to automate the grading.

Report – Please use the IEEE conference template[2] to submit a PDF file with your source code. In your report, make sure to include:

1. The final network structure used and your choice of hyperparameters.

---

[1] http://cs231n.github.io/neural-networks-3/#gradcheck
http://deeplearning.stanford.edu/tutorial/supervised/DebuggingGradientChecking/

[2] https://www.ieee.org/conferences/publishing/templates.html

2. The learning curves on both training and validation set. Report the loss values and the accuracy.
3. Your final accuracy on the test set.

Predictions – Please submit a CSV file with your predictions for the test set. Note that there is a separate link for this submission. The data should be in one-hot encoding format. That is, each row should be a set of zeros with only a one on the value that you predicted (e.g., the value with the largest probability). The column entries should be in ascending order based on the label (i.e., the '0' label goes first all the way to '9'). Finally, each row should be the prediction for the corresponding test entry. Make sure they are all in the same order. If you did this correctly, you should end up with a table of size 1000 rows by 10 columns.