Project 01 – Math

Arthur J. Redfern
arthur.redfern@utdallas.edu
Sep 14, 2020

---

1 Logistics

- Assigned Sep 14, 2020 and due Sep 28, 2020
- This is an individual project, no help from others is allowed
- The use of any and all online resources is allowed
- **The use of xNN packages is not allowed (i.e., no PyTorch, no TensorFlow, …)**
- All code should be written by you – do not simply copy code from others

2 Goals

- The math section of the course provides the theory for understanding
  - Using layers to map input data to features to output class pmf
  - Computing an error by comparing the output class pdf with the true pdf
  - Back propagating the sensitivity of the error with respect to feature maps
  - Computing the sensitivity of the error with respect to parameters
  - Updating the parameters to minimize the error
- In this project you will put theory to practice and write Python (with NumPy) code to do all of the above for a simple neural network

3 Project

3.1 Neural Network (Max Grade 80/100)

Implement the following pseudo code in Python to define and train a traditional neural network for MNIST digit recognition and display key results:

```
# cycle through the epochs

    # set the learning rate

    # cycle through the training data
        # forward pass
        # loss
        # back prop
```

```
            # weight update

        # cycle through the testing data
            # forward pass
            # accuracy

        # per epoch display (epoch, time, training loss, testing accuracy, ...)

    # accuracy display
    # final value
    # plot of accuracy vs epoch

    # performance display
    # total time
    # per layer info (type, input size, output size, parameter size, MACs, ...)
```

For the forward pass use the following network structure with indicated output sizes:

Layer	Output
Data	1 x 28 x 28
Division by 255.0	1 x 28 x 28
Vectorization	1 x 784
Matrix multiplication	1 x 1000
Addition	1 x 1000
ReLU	1 x 1000
Matrix multiplication	1 x 100
Addition	1 x 100
ReLU	1 x 100
Matrix multiplication	1 x 10
Addition	1 x 10
Softmax	1 x 10

A Python template that works in Google Colab is provided to download and format MNIST as a NumPy array and provide some basic display functionality. Start from that code.

A suggestion is to do some planning before you start writing you own code. While you're not allowed to use a package like PyTorch, you are allowed to borrow aspects of its API where it makes sense. For example, perhaps it would be useful to define each layer as a class with parameters, initialization and forward and backward functions? As you're planning your code, think about what information do you need to save from the forward pass to implement the backward pass.

3.2 Convolutional Neural Network (Max Grade 100/100)

This class has CNN in its name so it wouldn't be right to get top marks on the 1st project without demonstrating some understanding of CNN layers and their use in a xNN.

To unlock a max grade of 100, do everything in 3.1 for a traditional neural network architecture then repeat it for the following convolutional neural network architecture:

Layer	Output
Data	1 x 28 x 28
Division by 255.0	1 x 28 x 28
3x3/1 0 pad CNN style 2D conv	16 x 28 x 28
Addition	16 x 28 x 28

```
ReLU                            16 x 28 x 28
3x3/2 0 pad max pool            16 x 14 x 14
3x3/1 0 pad CNN style 2D conv   32 x 14 x 14
Addition                        32 x 14 x 14
ReLU                            32 x 14 x 14
3x3/2 0 pad max pool            32 x 7 x 7
3x3/1 CNN style 2D conv         64 x 7 x 7
Addition                        64 x 7 x 7
ReLU                            64 x 7 x 7
Vectorization                    1 x 3136
Matrix multiplication            1 x 100
Addition                         1 x 100
ReLU                             1 x 100
Matrix multiplication            1 x 10
Addition                         1 x 10
Softmax                          1 x 10
```

3.3 Impress Me (Max Grade ???)

If you do something extra beyond what is required above for the neural network and convolutional neural network, maybe you can earn some extra points. As a general rule, the more impressed I am by what you do and the quality by which you do it, the more points you can get. If everyone does the same extra thing, I'll be less impressed by it.

Possibilities include:

- Using batches of data for training (vs 1 sample at a time)
- Adding additional functionality to the layers (down sampling CNN style 2D conv layer?)
- Enabling flexibility in the network definition via repeating certain blocks of layers multiple times (for examples of this see the MNIST and CIFAR-10 code on the class GitHub page)
- Auto generating the backward pass from the network definition
- Implementing and training a network on CIFAR-10
- Implementing and training a network which includes branching in the structure
- …

4 What To Turn In Via eLearning

- A Python file I can cut and paste into Google Colab, run and reproduce you results
    - For the traditional neural network call this file nn.py (required)
    - For the convolutional neural network call this file cnn.py (optional)
    - For the extra / impress me portion of things (optional)
        - If this is included in the nn.py or cnn.py code then be sure to describe what you did in the comment block at the top
        - If it's not included in the nn.py or cnn.py files then include and clearly comment the minimum set of files to describe and demo what you did
- I like lots of nicely formatted informative comments
    - At the top of the file describing things at a high level

- Throughout the code describing specific portions