ECE/CS 559: Neural Networks **Fall 2024**

Homework 6

Due: **Tuesday** October 29, 2024 (by 9pm, on Gradescope)

Note: *You may discuss these problems in groups. However, you must write up your own solutions and mention the names of the people in your group. Also, please do mention any solution manuals, online material, or other sources you used in a major way.*

Submit your code as a **two** `.py` **files**, one for each question, in Gradescope. Make sure that it **runs properly** and generates all the plots that you report on in the main submission.

1. **[Softmax Layer and Crossentropy Loss]**

   Consider the *softmax* layer:
   $$f_i = \frac{e^{z_i}}{\sum_{i'} e^{z_{i'}}}.$$
   Assuming y is non-negative vector with $\sum y_i = 1$ (e.g., a one-hot vector or a probability vector), define the crossentropy loss as:
   $$\ell(y, f) = \sum_i y_i \log \frac{1}{f_i}.$$

   (a) Calculate $\nabla_f \ell$.

   (b) Derive the backpropagation equation from $\nabla_f \ell$ to $\nabla_z \ell$.

   (c) Substitute (a) in (b) to obtain $\nabla_z \ell$ in terms of f and y directly.

2. **[PyTorch and MNIST]**

   (a) Read the PyTorch tutorial at `https://pytorch.org/tutorials/beginner/basics/intro.html`.

   (b) Follow similar steps to:

   - Load the MNIST dataset (not FashionMNIST).
   - Train a 784-200-200-10 fully connected feed-forward neural network with ReLU activation functions using the training data. Feel free to adjust hyperparamters (learning rate, number of epochs, batch size, etc.)
   - Report the results (accuracy and average loss per epoch), as well as all the hyperparameters you chose.

Note: The tutorial uses the crossentropy loss, but it ends the network with a linear layer. These outputs are interpreted as the z in Q1, and a softmax is taken inside the loss function. The softmax transforms z into a probability vector. The gradient is therefore the same expression as you found in Q1(c).

3. **[CNN]**

   (a) Modify the code from Q2 to instead have a convolutional neural network, with the following architecture:

   - Map the input to 20 channels using 4×4 linear filters followed by ReLU units, with stride 1.
   - Map those 20 channels into another 20 channels using 4×4 linear filters followed by ReLU, with stride 2.
   - Reduce the dimension of these channels further, using max-pooling within 2×2 blocks.
   - Flatten the channels into a vector, and follow these convolutional layers with a linear layer bringing the dimension down to 250, followed by a ReLU, then a final linear layer reducing the dimension to 10.

   Report your Python code snippet of this architecture in your main submission.

   (b) Train this neural network the same way as you did in Q2, without changes from the tutorial. Report on the results.

   (c) Add weight decay and dropout regularization, adjusting the corresponding hyperparameter to see if you can improve on the result in (b). Report on the final hyperparameters you use (namely, how much weight decay, how much dropout, and after which layers), along with the results. Note: In PyTorch, you add weight decay as a parameter to the optimizer. You add dropout to a layer by adding an extra "dropout layer" before it, which will zero-out some of the inputs during training, but will not do it during "evaluation mode". You can decide to add dropout after as many layers as you want.

   (d) One of the tweaks that has a great impact on the training of CNNs (and other neural networks) is the concept of "batch normalization". This is very related to how we justified the importance of centering and scaling the inputs and weights before training, to make sure, for example, that they're not all non-negative. Since ReLU produces non-negative outputs, we have the same issue. Batch normalization will attempt to do the centering and scaling as part of the neural network's architecture. Add batch normalization after every ReLU in your architecture, and train again. Also try to change the learning rate to a higher value and any other tweaks you can think of in order to get the best possible result. Report on the learning rate you used and any other tweaks you implemented, along with your results.