Intro to Image Understanding (CSC420)
Assignment 2

Due Date: October 25^{th}, 2021, 10:59 pm
Total: 200 marks

General Instructions:

- You are allowed to work directly with one other person to discuss the questions. However, you are still expected to write the solutions/code/report in your own words; i.e. no copying. If you choose to work with someone else, you must indicate this in your assignment submission. For example, on the first line of your report file (after your own name and information, and before starting your answer to Q1), you should have a sentence that says: *"In solving the questions in this assignment, I worked together with my classmate [name & student number]. I confirm that I have written the solutions/-code/report in my own words".*
- Your submission should be in the form of an electronic report (PDF), with the answers to the specific questions (each question separately), and a presentation and discussion of your results. For this, please submit a file named **report.pdf** to MarkUs directly.
- Submit documented codes that you have written to generate your results separately. Please store all of those files in a folder called **assignment2**, zip the folder and then submit the file **assignment2.zip** to MarkUs. You should include a **README.txt** file (inside the folder) which details how to run the submitted codes.
- Do not worry if you realize you made a mistake after submitting your zip file; you can submit multiple times on MarkUs.

Part I: Theoretical Problems (80 marks)

[Question 1] Activation Functions (10 marks)

Show that in a fully connected neural network with linear activation functions, the number of layers has effectively no impact on the network.

Hint: Express the output of a network as a function of its inputs and its weights of layers.

[Question 2] Back Propagation (15 marks)

Consider the neural network architecture shown in the Figure below. Nodes denoted by

x_i are input variables, and $\hat{y}$ is the output variable). The node Σ takes the sum of its inputs, and σ denotes the logistic function

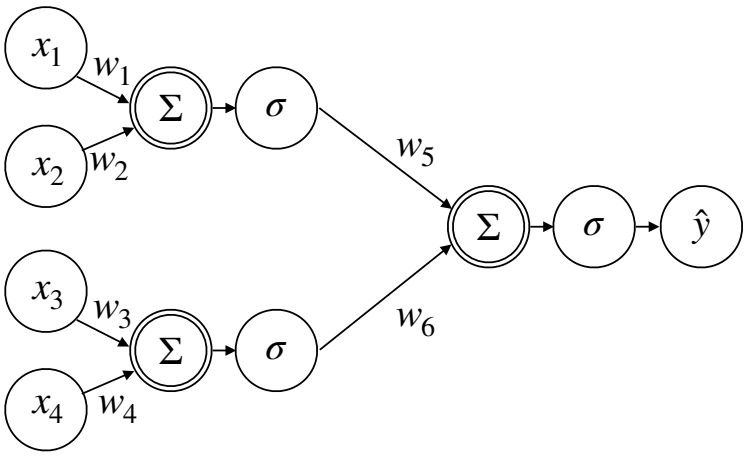
$$\sigma(x) = \frac{1}{1 + e^{-x}}.$$

Suppose the loss function used for training this neural network is **L2** loss, i.e. $L(y, \hat{y}) = ||y - \hat{y}||_2^2$. Assume that the network has its weights set as:

$$(w_1, w_2, w_3, w_4, w_5, w_6) = (0.75, -0.63, 0.24, -1.7, 0.8, -0.2)$$

Given an input data point $(x_1, x_2, x_3, x_4) = (0.9, -1.1, -0.3, 0.8)$ with true label of 0.5, compute the partial derivative $\frac{\partial L}{w_3}$, by using the back-propagation algorithm. Round all your calculations to 4 decimal places.

Hint: For any vector (or scalar) $\mathbf{x}$, we have $\frac{\partial}{\partial \mathbf{x}}(||\mathbf{x}||_2^2) = 2\mathbf{x}$. Also, you do not need to write any code for this question! You can do it by hand.



[Question 3] Convolutional Neural Network (15 marks)

In this problem, our goal is to estimate the computation overhead of CNNs by counting the FLOPs (floating point operations). Consider a convolutional layer C followed by a max pooling layer P. The input of layer C has 50 channels, each of which is of size 12×12. Layer C has 20 filters, each of which is of size 4×4. The convolution padding is 1 and the stride is 2. Layer P performs max pooling over each of the C's output feature maps, with 3×3 local receptive fields, and stride 1.

Given scalar inputs x_1, x_2, ..., x_n, we assume:

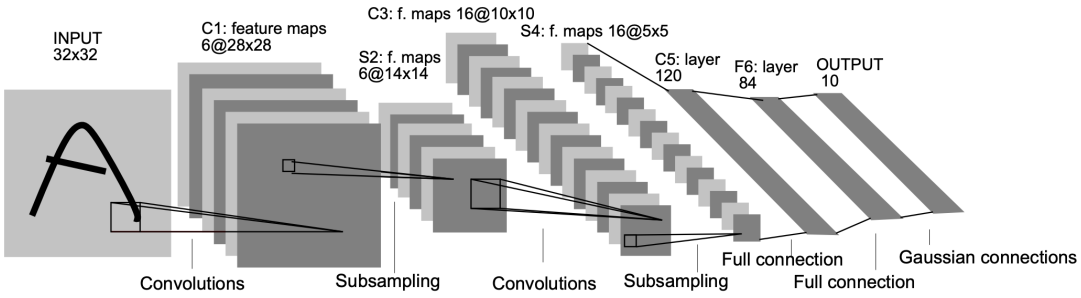
- A scalar multiplication $x_i.x_j$ accounts for one FLOP.
- A scalar addition $x_i + x_j$ accounts for one FLOP.

- A max operation $\max(x_1, x_2, ..., x_n)$ accounts for $n - 1$ FLOPs.
- All other operations do not account for FLOPs.

How many FLOPs layer C and P conduct in total during one forward pass, with and without accounting for bias?

[Question 4] Trainable Parameters (15 marks)

The following CNN architecture is one of the most influential architectures that was presented in the 90s. Count the total number of trainable parameters in this network. Note that the Gaussian connections in the output layer can be treated as a fully connected layer similar to $F6$.



[Question 5] Logistic Activation Function (10 marks)

For backpropagation in a neural network with logistic activation function, show that, in order to compute the gradients, as long as we have the outputs of the neurons, there is no need for the inputs.

Hint: Find the derivative of a neuron's output with respect to its inputs.

[Question 6] Hyperbolic Tangent Activation Function (15 marks)

One alternative to the logistic activation function, is hyperbolic tangent function:

$$\mathbf{tanh}(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}}.$$

- (a) What is the output range for this function, and how it differs from the output range of logistic function?
- (b) Show that its gradient can be formulated as a function of logistic function; and
- (c) When do we want to use each of these activation functions?

Part II: Implementation Tasks (120 marks)

In this question, the goal is to investigate the performance of a simple neural network in classifying different letters. All the tasks should be implemented using only Python, NumPy, and a deep learning package of your choice; e.g. PyTorch, TensorFlow, etc.

Task I - Data Preparation (15 marks):

The dataset that we will use in this assignment is a permuted version of **notMNIST**[1], which contains 28×28 images of 10 letters (A to J) taken from different fonts. This dataset has 18720 instances which you should divide them into 3 different sets: 15000 images for the training set, 1000 instances for the validation set and the remaining part for the test set. It is important to have balanced divisions for each of your classes to avoid any bias. Load the data into proper data structures; e.g. if you are using PyTorch, load them into tensors. Apply any required transformations to the images. Make sure to include enough justification for your choice of preprocessing pipeline. You may want to come back to this step as you go through the training. Remember that preprocessing may have enormous impact on the training results. Include your code snippet for this part in the report.

For all the tasks that follow, use RELU as your activation function, and cross entropy for your cost function. You should also apply a softmax layer to your output layer. To have an estimation about the running time, training the following neural networks, will not take more than an hour on an Intel core i7 CPU @1.73 GHz with 4GB RAM. That means you should be able to quickly train and test the networks using your own personal machine, Google Colab, or teach.cs machines.

Task II - Neural Network Training (25 marks):

Implement a simple neural network with one hidden layer and 1000 hidden units. Train your neural network with the aforementioned characteristics. For training the network, you are supposed to find a reasonable value for your learning rate. As we always do for picking the best value for a hyperparameter (which learning rate is one of them), you should train your neural network for different values of learning rate and choose the one that gives you the least validation error. Trying 5 different values will be enough. You may also find it useful to decay the weights as training procedure goes on. Plot the number of training, validation, and test error vs. the number of epochs. Make sure to use early stopping for your training to avoid overfitting. After stopping the procedure of training, what are the training, test, and validation errors?

[1] http://yaroslavvb.blogspot.ca/2011/09/notmnist-dataset.html

Task III - Testing the Number of Hidden Units (25 marks):

Instead of using 1000 hidden units, train three different neural networks with $100, 500$, and 1000 hidden units, respectively. Find the best validation error for each one. Choose the model which gives you the best result, and then use it for classifying the test set. Include all the validation errors and the final test error in your report. In one sentence, summarize your observation about the effect of the number of hidden units in the final results.

Task IV - Testing the Number of Layers (25 marks):

For this task fix the number of hidden units to 1000. This time, train a neural network with two hidden layers with the same number of hidden units. So each layer has 500 units. Plot the number of training and validation errors vs. the number of epochs. What is the final validation error when the training is done? Using the test set, compare this architecture with the one-layer case.

Task V - Dropout (30 marks):

Dropout is a powerful technique to decrease the overfitting and enhance the overall performance of the neural network. Using the same architecture in Task 2, introduce dropout on the hidden layer of the neural network (with rate 0.5) and train the network. You should use the dropout only in the training procedure, and not in the forward propagation. You can use any existing implementation for dropout; e.g. PyTorch has functions that incorporates dropout in the training. Plot the training and validation error vs. the number of epochs. Compare the results with the case that we do not have any dropout. In one sentence, summarize your observation about the effect of dropout.