Homework 2

Deadline: Sunday, Feb 19th, at 11:59pm.

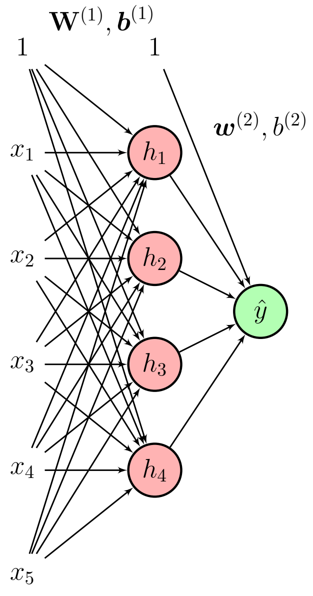
Submission: You need to submit a `APL405_HW2_<RollNumber>_<FirstName>.zip` file on Microsoft Teams

- Put all your solutions and answers to all questions as a PDF file titled `hw2_writeup.pdf`. You can produce the file however you like (e.g. LATEX, Microsoft Word, scanner), as long as it is readable.
- Your final codes for Question 4 and 5 should be submitted as python notebook `.ipynb` files.

Late Submission: 50% of the marks will be deducted for any submission beyond the deadline. No submissions will be accepted after two days past the deadline.

Collaboration: Homeworks are individual work. Please refrain from copying.

1. [**3 marks**] **Hand-design a neural network**: In this problem, you will manually decide on a set of weights and biases for a two-layer neural network (1-hidden layer) which determines if a list of five numbers are in the descending order or not. More specifically, you receive five inputs x_1, x_2, x_3, x_4 and x_5, where $x_i \in R$, and the network must output 1 if $x_1 > x_2 > x_3 > x_4 > x_5$ and 0 otherwise. You will use the following architecture consisting of an input layer, one hidden layer with four hidden units (or nodes) and one output layer with one node.


$\mathbf{W}^{(1)}, \boldsymbol{b}^{(1)}$
1
1
$\boldsymbol{w}^{(2)}, b^{(2)}$
x_1
x_2
x_3
x_4
x_5
h_1
h_2
h_3
h_4
$\hat{y}$

Assume the activation function $\sigma(z)$ of the hidden units and the output unit to be a hard threshold function, such that,

$$\sigma(z) = \begin{cases} 1 & \text{if} \quad z \geq 0 \\ 0 & \text{if} \quad z < 0 \end{cases}$$

Find a set of weights and biases for the network which correctly implements this function (including cases where some of the inputs are equal). Note, in some cases, some of the inputs may equal to each other for which your output should be 0. Your answer should include:

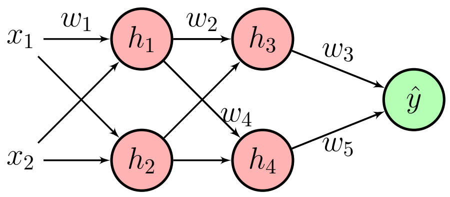
(a) A 4×5 weight matrix $\mathbf{W}^{(1)}$ for the hidden layer.

(b) A 4-dimensional vector of biases $\boldsymbol{b}^{(1)}$ for the hidden layer.

(c) A 4-dimensional vector of weights $\boldsymbol{w}^{(2)}$ for the output layer.

(d) A scalar bias $b^{(2)}$ for the output layer.

2. **[2 marks] Gradients with ReLU activation**: An interesting feature of the ReLU activation function is that it *sparsifies* the activations and the derivatives, that is, it sets a large fraction of the values to zero for any given input vector. Consider the following network, with all hidden units having ReLU activation functions:



Note that each w_i refers to the weight of a single connection, not the whole layer. Suppose we are trying to minimize a loss function L which depends only on the activation of the output unit $\hat{y}$ (For instance, L could be the squared error loss $\frac{1}{2}(y-\hat{y})^2$).

Suppose the unit h_1 receives an input of -1 on a particular training case, so the ReLU evaluates to 0. Based only on this information, which of the weight derivatives

$$\frac{\partial \mathcal{L}}{\partial w_1}, \frac{\partial \mathcal{L}}{\partial w_2}, \frac{\partial \mathcal{L}}{\partial w_3}$$

are guaranteed to be 0 for this training case? Write YES or NO for each. Justify your answers.

3. **[3 marks] Computational graph for backpropagation**: Consider a two-layered neural network consisting of N inputs, K hidden nodes and N output nodes. The forward computation of the output is done in the following way:

$$\begin{aligned}\boldsymbol{z} &= \mathbf{W}^{(1)}\boldsymbol{x} + \boldsymbol{b}^{(1)}\\ \boldsymbol{h} &= \sigma(\boldsymbol{z})\\ \hat{\boldsymbol{y}} &= \boldsymbol{x} + \mathbf{W}^{(2)}\boldsymbol{h} + \boldsymbol{b}^{(2)}\end{aligned}$$

where $\sigma(\cdot)$ is the sigmoid (logistic) function, applied element wise. The specially designed loss function L has both $\boldsymbol{h}$ and $\hat{\boldsymbol{y}}$:

$$\begin{aligned}L &= S + R\\ S &= \frac{1}{2}\left\|\hat{\boldsymbol{y}} - \boldsymbol{s}\right\|_2^2\\ R &= \boldsymbol{r}^T\boldsymbol{h}\end{aligned}$$

where $\boldsymbol{r}$ and $\boldsymbol{s}$ are given vectors.

(a) Draw the computation graph relating $\boldsymbol{x}$, $\boldsymbol{z}$, $\boldsymbol{h}$, $\boldsymbol{y}$, R, S and L.

(b) Derive the backpropagation equations for computing $\bar{\boldsymbol{x}} = \frac{\partial \mathcal{L}}{\partial \boldsymbol{x}}$. You may use σ' to denote the derivation of the sigmoid function, so you do not have to write it out explicitly.

4. **[7 marks] Coding FNN for MNIST**
Put all your plots and answers in the `hw2_writeup.pdf` file.

In class, we had talked about a two-layer feed-forward neural network for classifying MNIST handwritten digits. You are now given a starter code here [1] . Run the code as it is. You will notice that after training is done, three figures will appear.

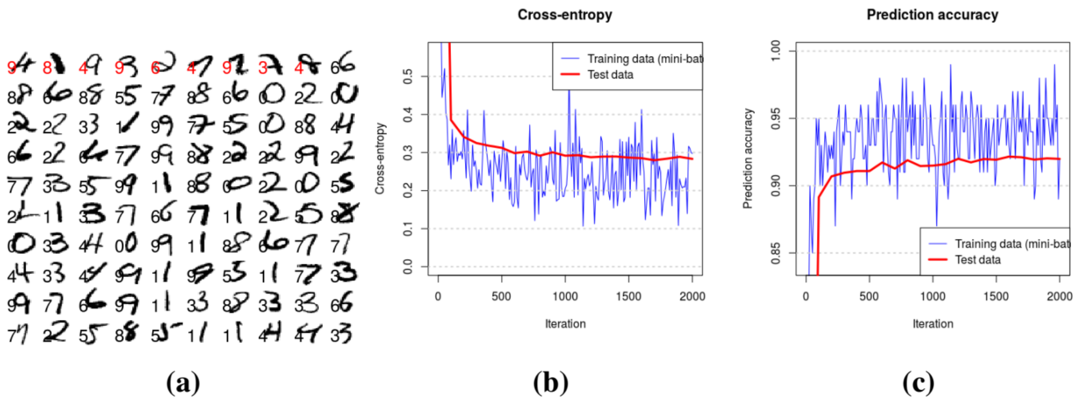


Figure 1: Three figures generated by the code: (a) Prediction performance on 100 randomly selected test points, (b) Cost function (cross-entropy) on test/training data, (c) Prediction accuracy on test/training data

100 randomly selected test images (out of the total 10 000 test images) are displayed together with their predicted label. For those images that have been incorrectly classified, the label is colored red. We use a cross-entropy loss function

(i) **[0.25 mark]** Why is the training accuracy so “noisy”?

(ii) **[0.25 mark]** What classification accuracy do you get on the test data?

(iii) **[0.5 mark]** How many parameters does this single-layer neural network model have?

(iv) **[1 mark]** Recall *epoch* vs *iteration* from the lecture notes. How many iterations does the code take until it has seen all training data points, i.e., how many iterations are included in each epoch? How many epochs do you think is being completed in the given code?

Next, add a hidden layer to make a **two-layer neural network**. Choose 200 hidden neurons. Keep softmax activation on the last layer, but use the sigmoid (or logistic) activation function for the hidden neurons. You can use `torch.sigmoid` command.

[1] Please go through Practical 7 for an introduction to PyTorch

Save the new file as `mnist_twolayers.ipynb`. Try using the following weights and biases

```
U = 200 # number of hidden units
self.W1 = nn.Parameter(0.1 * torch.randn(784, U))
self.b1 = nn.Parameter(torch.zeros(U))
self.W2 = nn.Parameter(0.1 * torch.randn(U, 10))
self.b2 = nn.Parameter(torch.zeros(10))
```

The forward model will have:

```
Q = torch.sigmoid(X.mm(self.W1) + self.b1)
G = F.softmax(Q.mm(self.W2) + self.b2, dim=1)
```

(v) [**0.5 mark**] What classification accuracy do you get on the test data now? Does the accuracy improve over a single-layer neural net? Plot the prediction accuracy for training and test data.

(vi) [**1 mark**] Try some other numbers of hidden neurons (ranging from 10 to 750). How does it affect the performance?

(vii) [**0.5 mark**] What happens if you initialize the weights with zeros as we did for the single layer neural network?

Continue to create even deeper neural nets. Save `mnist_twolayers.ipynb` as a new file `mnist_fivelayers.ipynb`. Add in total of five layers with 200, 100, 60 and 30 hidden neurons between each layer.

(viii) [**0.5 mark**] What classification accuracy do you get on test data? Does the accuracy improve over the two-layers neural net? Plot the prediction accuracy for training and test data.

(ix) [**0.5 mark**] Instead of using sigmoid activation function, use *Rectified Linear Unit* (ReLU) activation. To use ReLU, simply replace all `torch.sigmoid` in the code with `F.relu`. What classification accuracy on the test data do you get? Plot the prediction accuracy for training and test data.

(x) [**0.5 mark**] What happens when the weights and biases are all initialized to zero values?

(xi) [**0.5 mark**] Did you find any `NaN` values cropping up while training? If so, what do you think might be the reason? Instead of using `F.softmax` in the forward model, return only the logit

```
Z = Q4.mm(self.W5) + self.b5
return Z
```

and replace `crossentropy` with `F.cross_entropy`. Report if `NaN` values go away.

(xii) [**0.25 mark**] How many parameters does this five-layer neural network have?

(xiii) [**0.5 mark**] Use the Adam optimizer. Replace the `optim.SGD` with `optim.Adam`. Change the learning rate from 0.5 to 0.003. Use ReLU activations and random initialization of weights and small-positive (= 0.1) initialization of biases. What classification accuracy on the test data do you get? Is there any improvement in the classification accuracy on the test data? Justify why is there an improvement or deterioration. Plot the prediction accuracy for training and test data.

(xiv) **[0.75 mark]** Train for 10000 iterations. Plot the prediction accuracy for training and test data. Do you think you can get better classification accuracy on test data if you train longer?

5. **[5 marks] Coding CNN for MNIST**
Put all your plots and answers in the `hw2_writeup.pdf` file.

The previous question considered an FNN model where $28 \times 28 \times 1$[2] pixels in each image was *flattened* into a long vector with 784 elements. This destroys spatial information present in the images. CNN can exploit spatial information better compared to FNN. In this question, you will implement a CNN.

The CNN you create will have three convolutional layers and two final dense layers. The settings for the three convolutional layers are given in Table 1

Table 1: Architechture of the three convolutional layers

	Layer 1	**Layer 2**	**Layer 3**
Number of kernels/filters	4	8	12
Filter size	5×5	5×5	4×4
Stride (equal in both directions)	1	2	2
Zero Padding (equal in both directions)	2	2	1

You have to use PyTorch for constructing the CNN model. PyTorch requires weight tensors of the size: (`output channels` $\times$ `input channels` $\times$ `filter rows` $\times$ `filter columns`). Hence the weight tensor and offset vector for the first convolutional layer.

```
U1 = 4
self.W1 = nn.Parameter(0.1 * torch.randn(U1, 1, 5, 5))
self.b1 = nn.Parameter(torch.ones(U1)/10)
```

The corresponding model implementation for that convolutional layer is

```
Q1 = F.relu(F.conv2d(X, self.W1, bias=self.b1, stride=1,
   padding=2))
```

After the three convolutional layers, we use two dense layers. Before we can apply the first dense layer, all hidden units in the third convolutional layer needs to be flattened into a long vector. This can be done with the command

```
Q3flat = Q3.view(-1, U3flat)
```

(i) **[0.5 mark]** How many hidden units are there in total in the third convolutional layer, i.e., what is `U3flat` supposed to be in the code above?

[2]Note a grayscale image has depth of one, therefore the number of input channel is 1.

(ii) **[1.5 marks]** Save `mnist_fivelayers.ipynb` as a new file `mnist_CNN.ipynb`. Replace the first three dense layers in the previous code with three convolutional layers using the settings in Table 1 for each of these three layers. Add the reshaping command according to what is stated above. Update the fourth and fifth layer according to the instructions above. Train for at least 4000 iterations.

What prediction accuracy do you achieve on test data? Plot the prediction accuracy of test and train data.

(iii) **[1.5 marks]** Towards the end of training phase, the learning rate $\eta = 0.003$ may be really too large. The prediction accuracy and/or the cross-entropy on test data would oscillate and we may not get down to the best minimum of the cost function. You would therefore prefer having a lot smaller η, but with a very small η from the very beginning of the training will increase the training time. One solution is to start learning fast (to get approximately close to the minimum) and then slow down.

Reduce the learning rate `lr` slowly. Adjust the learning rate as the following function

$$\eta^{(t)} = \eta_{\min} + (\eta_{\max} - \eta_{\min})\, e^{-t/2000}$$

with $\eta_{\min} = 0.0001$ and $\eta_{\max} = 0.003$. Incorporate this adjustment in the code, and train with atleast 6000 iterations. What prediction accuracy on the test data did you achieve? Plot the prediction accuracy of test and train data.

(iv) **[0.5 mark]** At this point, the network can start overfitting on training data. How can you see that?

(v) **[1 mark]** If you manage to get 99% prediction accuracy on test data, plot the prediction accuracy of test and train data.