END-Semester Exam (100 points)
May 4, 2023 **Course** : Natural Language Processing

Name: ______________________
Student Number: ______________________

This exam has 18 printed page/s. **Write your roll number on every sheet**. Write your answer clearly within the spaces provided. **If you need more space than is provided, you probably made a mistake in interpreting the question.** Start with rough work at the end of the question paper. Use the marks alongside each question for time management. **Illogical or incoherent answers are worse than wrong answers or even no answers and may fetch negative credit.** You may not use any computing or communication device during the exam.**The exam will be closed notes. One page (both sides) of hand-written notes is allowed**.

---

1. True/False question and also write your explanation:
   (a) (3 points) Suppose we have a loss function $f(x, y; \theta)$, defined in terms of parameters θ, inputs x, and labels y. Suppose that, for some special input and label pair (x_0, y_0), the loss equals zero. Then the gradient of the loss with respect to θ is equal to zero (True, False).

   (b) (3 points) During backpropagation, when the gradient flows backwards through the sigmoid or tanh non-linearities, it cannot change sign (True, False).

   (c) (3 points) Suppose we are training a neural network with stochastic gradient descent on mini-batches. True or false: summing the cost across the minibatch is equivalent to averaging the cost across the minibatch, if in the first case we divide the learning rate by the minibatch size (True, False).

2. Suppose we want to classify movie review text as (1) either positive or negative sentiment and (2) either action, comedy, or romance movie genre. To perform these two related classification tasks, we use

a neural network that shares the first layer but branches into two separate layers to compute the two classifications. The loss is a weighted sum of the two cross-entropy(CE) losses.

$$\mathbf{h} = \text{ReLU}(W_0\mathbf{x} + b_0)$$
$$\hat{y}_1 = \text{softmax}(W_1\mathbf{h} + b_1)$$
$$\hat{y}_2 = \text{softmax}(W_2\mathbf{h} + b_2)$$
$$J = \alpha\text{CE}(y_1,\hat{y}_1) + \beta\text{CE}(y_2,\hat{y}_2)$$

Here input $x \in \mathbb{R}^{10}$ is some vector encoding of the input text, label $\hat{y}_1 \in \mathbb{R}^2$ is a one-hot vector encoding the true sentiment, label $\hat{y}_2 \in \mathbb{R}^3$ is a one-hot vector encoding the true movie genre,$h \in \mathbb{R}^{10}$ is a hidden layer, $W_0 \in \mathbb{R}^{10*10}$,$W_1 \in \mathbb{R}^{2*10}$,$W_2 \in \mathbb{R}^{3*10}$ are weight matrices, and α and β are scalars that balance the two losses.

(a) (4 points) To compute backpropagation for this network, we can use the multivariable chain rule. Assuming we already know:

$$\frac{\partial \hat{\mathbf{y}}_1}{\partial \mathbf{x}} = \mathbf{\Delta}_1, \qquad \frac{\partial \hat{\mathbf{y}}_2}{\partial \mathbf{x}} = \mathbf{\Delta}_2, \qquad \frac{\partial J}{\partial \hat{\mathbf{y}}_1} = \delta_3^T, \qquad \frac{\partial J}{\partial \hat{\mathbf{y}}_2} = \delta_4^T$$

(b) (4 points) When we train this model, we find that it underfits the training data. Why might underfitting happen in this case? Provide at least one suggestion to reduce underfitting.

(c) (4 points) In class, we learned how the ReLU activation function ReLU(z) = max(0, z) could "die" or become saturated/inactive when the input is negative. A friend of yours suggests the use of another activation function, f(z) = max(0.2z, 2z), which he claims will remove the saturation problem. Would this address the problem? Why or why not?

(d) (4 points) The same friend also proposes another activation function, $g(z) = 1.5z$. Would this be a good activation function? Why or why not?

(e) (4 points) There are several tradeoffs when choosing the batch size for training. Name one advantage of having very large batch sizes during training. Also, name one advantage of having very small batch sizes during training.

(f) (4 points) Suppose we are training a feed-forward neural network with several hidden layers (all with ReLU non-linearities) and a softmax output. A friend suggests that you should initialize all the weights (including biases) to zeros. Is this a good idea or not? Explain briefly in 1-2 sentences.

3. The most widely used activation function in deep learning now is ReLU: Rectified Linear Unit:

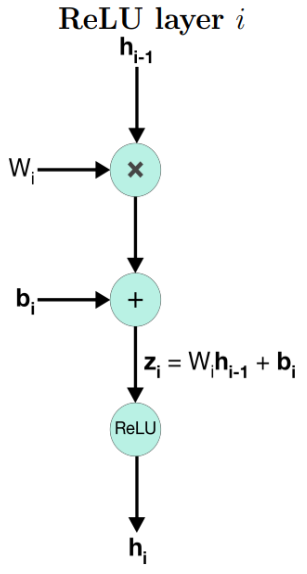
$$\text{ReLU(z) = max(0,z)}$$

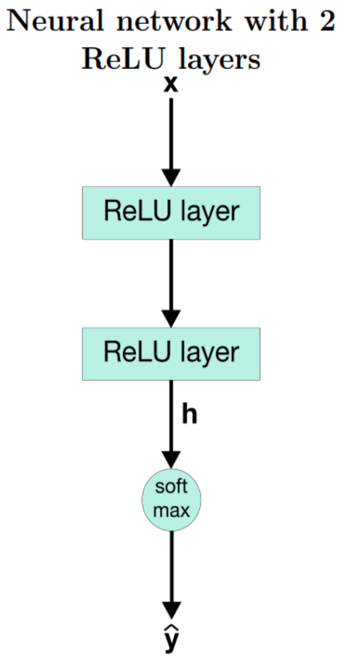
In this problem, we'll explore using the ReLU function in a neural network. Throughout this problem, you are allowed (and highly encouraged) to use variables to represent intermediate gradients (i.e., δ_1, δ_2, etc.).

(a) (3 points) Compute the gradient of the ReLU function, i.e., derive $\frac{\delta}{\delta 2} ReLU$.

$$1\{x > 0\} = \begin{cases} 1 & \text{if x} > 0 \\ 0 & \text{if x} \leq 0 \end{cases}$$

(b) (10 points) Now, we'll use the ReLU function to construct a multi-layer neural network. Below, the figure on the left shows a computation graph for a single ReLU hidden layer at layer i. The second figure shows the computation graph for the full neural network we'll use in this problem. The network is specified by the equations below:





$$\mathbf{z_1} = \mathbf{W}_1\mathbf{x} + \mathbf{b_1}$$
$$\mathbf{h_1} = \text{ReLU}(\mathbf{z_1})$$
$$\mathbf{z_2} = \mathbf{W}_2\mathbf{h_1} + \mathbf{b_2}$$
$$\mathbf{h_2} = \text{ReLU}(\mathbf{z_2})$$
$$\hat{\mathbf{y}} = \text{softmax}(\mathbf{h_2})$$
$$J = \text{CE}(\mathbf{y}, \hat{\mathbf{y}})$$
$$\text{CE}(\mathbf{y}, \hat{\mathbf{y}}) = -\sum_i y_i \log(\hat{y}_i)$$

The dimensions of our parameters and variables are $x \in \mathbb{R}^{D_x*1}$, $W_1 \in \mathbb{R}^{H*D_x}$,$b_1 \in \mathbb{R}^H$, $W_2 \in \mathbb{R}^{D_y*H}$,$b_2 \in \mathbb{R}^{D_y}$, $\hat{y} \in \mathbb{R}^{D_y*1}$ Note: x is a single-column vector.

1. (5 points) Compute the gradients $\frac{\delta J}{\delta W_2}$ and $\frac{\delta J}{\delta b_2}$.

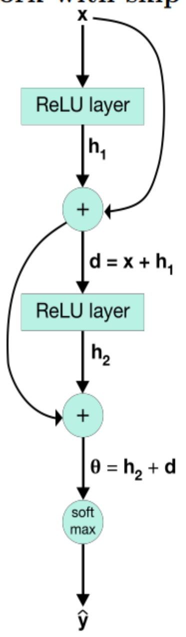
   Hint: Recall from PA1 that $\frac{\delta J}{\delta \theta}$ =$\hat{y}$ - y, where θ is the input of softmax.

2. (5 points) Compute the gradients $\frac{\delta J}{\delta W_1}$ and $\frac{\delta J}{\delta b_1}$. You may use gradients you have already derived in the previous part.

(c) (8 points) When neural networks become very deep (i.e., have many layers), they become difficult to train due to the vanishing gradient problem – as the gradient is back-propagated through many layers, repeated multiplication can make the gradient extremely small, so that performance plateaus or even degrades.
An effective approach, particularly in computer vision applications, is ResNet. The core idea of ResNet is **skip connections** that skip one or more layers. See the computation graph below:

For this part, the dimensions from part B still apply, but assume $D_y = D_x = H$. Compute the gradient $\frac{\delta J}{\delta x}$. Again, you are allowed (and highly encouraged) to use variables to represent intermediate gradients (i.e., $\delta_1, \delta_2,$, etc.)

Neural network with skip connections



$$\begin{aligned}
\mathbf{z_1} &= \mathbf{W}_1\mathbf{x} + \mathbf{b_1}\\
\mathbf{h_1} &= \mathrm{ReLU}(\mathbf{z_1})\\
\mathbf{d} &= \mathbf{h_1} + \mathbf{x}\\
\mathbf{z_2} &= \mathbf{W}_2\mathbf{d} + \mathbf{b_2}\\
\mathbf{h_2} &= \mathrm{ReLU}(\mathbf{z_2})\\
\theta &= \mathbf{h_2} + \mathbf{d}\\
\hat{\mathbf{y}} &= \mathrm{softmax}(\theta)\\
J &= \mathrm{CE}(\mathbf{y}, \hat{\mathbf{y}})
\end{aligned}$$

4. RNNs are versatile! and this family of neural networks has many important advantages and can be used in a variety of tasks. They are commonly used in many state-of-the-art architectures for NLP.

   (a) (12 points) For each of the following tasks, state how you would run an RNN to do that task. In particular, specify how the RNN would be used at **test** time (not training time), and specify :

   1. how many outputs, i.e., number of times the softmax $\hat{y^{(t)}}$ is called from your RNN. If the number of outputs is not fixed, state it as arbitrary.
   2. what each $\hat{y^{(t)}}$ is a probability distribution over (e.g., distributed over all species of cats)
   3. which inputs are fed at each time step to produce each output

      (a) (4 points) Named-Entity Recognition: For each word in a sentence, classify that word as either a person, organization, location, or none. Inputs: A sentence containing n words.

      (b) (4 points) Sentiment Analysis: Classify the sentiment of a sentence ranging from negative to positive (integer values from 0 to 4). Inputs: A sentence containing n words.

(c) (4 points) Language models: generating text from a chatbot that was trained to speak like you by predicting the next word in the sequence. Input: A single start word or token that is fed into the first time step of the RNN.

(b) (22 points) You build a sentiment analysis system that feeds a sentence into an RNN and then computes the sentiment class between 0 (very negative) and 4 (very positive) based only on the **final** hidden state of the RNN.

1. (3 points) What is one advantage that an RNN would have over a neural window-based model for this task?

2. (3 points)You observe that your model predicts very positive sentiment for the following passage: `Yesterday turned out to be a terrible day. I overslept my alarm clock, and to make matters worse, my dog ate my homework. At least my dog seems happy...` . Why might the model misclassify the appropriate sentiment for this sentence?

3. (9 points) Your friend suggests using an LSTM instead. Recall the units of an LSTM cell are defined as:

$$\begin{gathered}
i_t = \sigma(W^{(i)}x_t + U^{(i)}h_{t-1}) \\
f_t = \sigma(W^{(f)}x_t + U^{(f)}h_{t-1}) \\
o_t = \sigma(W^{(o)}x_t + U^{(o)}h_{t-1}) \\
\tilde{c_t} = \tanh(W^{(c)}x_t + U^{(c)}h_{t-1}) \\
c_t = f_t \cdot c_{t-1} + i_t \cdot \tilde{c_t} \\
h_t = o_t \cdot \tanh(c_t)
\end{gathered}$$

Where the final output of the last stem cell is defined by $\hat{y_t}$ = softmax($h_tW + b$). The final cost function, J, uses the cross-entropy loss. Consider an LSTM for two time steps, t and t-1. Derive the gradient $\frac{\delta J}{\delta U^{(c)}}$ in terms of the following gradients: $\frac{\delta h_t}{\delta h_{t-1}}$, $\frac{\delta h_{t-1}}{\delta U^{(c)}}$, $\frac{\delta J}{\delta h_t}$, $\frac{\delta c_t}{\delta U^{(c)}}$, $\frac{\delta c_{t-1}}{\delta U^{(c)}}$, $\frac{\delta c_t}{\delta c_{t-1}}$, $\frac{\delta h_t}{\delta c_t}$ and $\frac{\delta h_t}{\delta o_t}$ Not all of the gradients may be used. You can leave the answer in the form of a chain rule and do not have to calculate any individual gradients in your final resul

4. (4 points) Which part of the gradient $\frac{\delta J}{\delta U^{(c)}}$ allows LSTMs to mitigate the effect of the vanishing gradient problem? Explain in two sentences or less how this would help classify the correct sentiment for the sentence in part b).

5. (3 points) Rather than using the last hidden state to output the sentiment of a sentence, what could be a better solution to improve the performance of the sentiment analysis task?

5. (10 points) The field of neural computation started with perceptrons which are neurons that compute in the following way. If W.X $>$ t, then Y=1, else 0. W and X are weight and input vectors respectively, and t is a scalar called threshold. Your task is to give a neural network with ONE hidden layer only and NO MORE THAN two hidden neurons to compute N bit odd parity. The network should output 1 when the number of 1s in the input is an odd number.

   Solution:

6. (2 points) What have you learned from this course?

Rough page:

Rough page:

Rough page:

Rough page:

Rough page:

Rough page: