CS640 Homework 3: Neural Network

In this assignment, you will

1. derive both forward and backward propagation,
2. implement a neural network from scratch, and
3. run experiments with your model.

Collaboration

You are allowed to work in a team of at most **three** on the coding part(**Q2**), but you must run the experiments and answer written questions independently.

Instructions

General Instructions

In an ipython notebook, to run code in a cell or to render Markdown+LaTeX press `Ctrl+Enter` or `[>|]` (like "play") button above. To edit any code or text cell (double) click on its content. To change cell type, choose "Markdown" or "Code" in the drop-down menu above.

Most of the written questions are followed up a cell for you enter your answers. Please enter your answers in a new line below the **Answer** mark. If you do not see such cell, please insert one by yourself. Your answers and the questions should **not** be in the same cell.

Instructions on Math

Some questions require you to enter math expressions. To enter your solutions, put down your derivations into the corresponding cells below using LaTeX. Show all steps when proving statements. If you are not familiar with LaTeX, you should look at some tutorials and at the examples listed below between $..$. The OEIS website can also be helpful.

Alternatively, you can scan your work from paper and insert the image(s) in a text cell.

Submission

Once you are ready, save the note book as PDF file (File -> Print -> Save as PDF) and submit via Gradescope. In case some contents cannot be displayed properly in the PDF file, check out this tool: Colab2PDF.

Q0: Name(s)

Please write your name in the next cell. If you are collaborating with someone, please list their names as well.

Answer

Q1: Written Problems

Consider a simple neural network with three layers: an input layer, a hidden layer, and an output layer.

Let $w^{(1)}$ and $w^{(2)}$ be the layers' weight matrices and let $b^{(1)}$ and $b^{(2)}$ be their biases. For convention, suppose that w_{ij} is the weight between the ith node in the previous layer and the jth node in the current one.

Additionally, the activation function for both layers is the sigmoid function $\sigma(x) = \frac{1}{1+e^{-x}}$. Let $z^{(1)}$ and $z^{(2)}$ be the outputs of the two layers before activation, and let $a^{(1)} = \sigma(z^{(1)})$ and $a^{(2)} = \sigma(z^{(2)})$.

Lastly, we choose the L2 loss $L(y_{\text{true}}, y_{\text{predict}}) = \frac{1}{2}(y_{\text{true}} - y_{\text{predict}})^2$ as the loss function.

Q1.1: Forward Pass

Suppose that

$w^{(1)} = \begin{bmatrix} 0.4 & 0.6 & 0.2 \\ 0.3 & 0.9 & 0.5 \end{bmatrix}, b^{(1)} = [1, 1, 1]$; and

$w^{(2)} = \begin{bmatrix} 0.2 \\ 0.2 \\ 0.8 \end{bmatrix}, b^{(2)} = [0.5]$.

If the input is $a^{(0)} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$, what is the network output? Show your calculation steps and round your **final** answer to 2 digits after decimal.

Note: You should NOT round any intermediate results.

[Answer]

$$
\begin{aligned}
z^{(1)} &= {w^{(1)}}^T a^{(0)} + {b^{(1)}}^T \\
a^{(1)} &= \sigma(z^{(1)}) \\
z^{(2)} &= {w^{(2)}}^T a^{(1)} + {b^{(2)}}^T \\
a^{(2)} &= \sigma(z^{(2)})
\end{aligned}
$$

After plugging in numbers and rounding, the final answer is 0.82.

Show code

Q1.2: Backward Propagation

Derive the expressions of the following gradients:

1. $\frac{\partial L}{\partial w^{(2)}}$ and $\frac{\partial L}{\partial b^{(2)}}$
2. $\frac{\partial L}{\partial w^{(1)}}$ and $\frac{\partial L}{\partial b^{(1)}}$

For each gradient, start by deriving the element-level expression using chain rule, and then construct the final answer in matrix form. You can use a self-defined variable(e.g., η) to shorten a long expression (especially for the first-layer gradients).

An example of your answer should look like the following.

(Element level)

$$
\begin{aligned}
\frac{\partial L}{\partial w_i^{(2)}} &= \frac{\partial L}{\partial a^{(2)}} \cdot \frac{\partial a^{(2)}}{\partial z^{(2)}} \cdot \frac{\partial z^{(2)}}{\partial w_i^{(2)}} \ (\text{chain rule}) \\
&= L'(y_{\text{true}}, a^{(2)}) \cdot f_2'(z^{(2)}) \cdot a_i^{(1)} \ (\text{substitution}) \\
&= \dots \ (\text{further substitution and/or simplification if needed})
\end{aligned}
$$

(Matrix form)

$$
\frac{\partial L}{\partial w^{(2)}} = \dots \ (\text{only the final answer is needed})
$$

Note: The derivative of $\sigma(x)$ is $\sigma(x)(1-\sigma(x))$ if x is a scalar, or $\sigma(x) \odot (1-\sigma(x))$ if x is a vector.

[Answer]

Second layer:

$$
\begin{aligned}
\frac{\partial L}{\partial w_i^{(2)}} &= \frac{\partial L}{\partial a^{(2)}} \cdot \frac{\partial a^{(2)}}{\partial z^{(2)}} \cdot \frac{\partial z^{(2)}}{\partial w_i^{(2)}} \\
&= L'(y_{\text{true}}, a^{(2)}) \cdot f_2'(z^{(2)}) \cdot a_i^{(1)} \\
&= (a^{(2)} - y_{\text{true}}) \sigma(z^{(2)})(1 - \sigma(z^{(2)})) a_i^{(1)} \\
\frac{\partial L}{\partial b^{(2)}} &= \frac{\partial L}{\partial a^{(2)}} \cdot \frac{\partial a^{(2)}}{\partial z^{(2)}} \cdot \frac{\partial z^{(2)}}{\partial b^{(2)}} \\
&= L'(y_{\text{true}}, a^{(2)}) \cdot f_2'(z^{(2)}) \cdot 1 \\
&= (a^{(2)} - y_{\text{true}}) \sigma(z^{(2)})(1 - \sigma(z^{(2)}))
\end{aligned}
$$

Then, in matrix form

$$
\begin{aligned}
\frac{\partial L}{\partial w^{(2)}} &= a^{(1)} \cdot ((a^{(2)} - y_{\text{true}}) \odot (\sigma(z^{(2)})(1 - \sigma(z^{(2)}))))^T \\
\frac{\partial L}{\partial b^{(2)}} &= ((a^{(2)} - y_{\text{true}}) \odot (\sigma(z^{(2)})(1 - \sigma(z^{(2)}))))^T
\end{aligned}
$$

Let

$$
\eta = (a^{(2)} - y_{\text{true}}) \odot (\sigma(z^{(2)})(1 - \sigma(z^{(2)})))
$$

First layer:

$$
\begin{aligned}
\frac{\partial L}{\partial w_{ij}^{(1)}} &= \frac{\partial L}{\partial a^{(2)}} \cdot \frac{\partial a^{(2)}}{\partial z^{(2)}} \cdot \frac{\partial z^{(2)}}{\partial a_j^{(1)}} \cdot \frac{\partial a_j^{(1)}}{\partial z_j^{(1)}} \cdot \frac{\partial z_j^{(1)}}{\partial w_{ij}^{(1)}} \\
&= L'(y_{\text{true}}, a^{(2)}) \cdot f_2'(z^{(2)}) \cdot w_j^{(2)} \cdot f_1'(z^{(1)}) \cdot a_i^{(0)} \\
\frac{\partial L}{\partial w_{ij}^{(1)}} &= \frac{\partial L}{\partial a^{(2)}} \cdot \frac{\partial a^{(2)}}{\partial z^{(2)}} \cdot \frac{\partial z^{(2)}}{\partial a_j^{(1)}} \cdot \frac{\partial a_j^{(1)}}{\partial z_j^{(1)}} \cdot \frac{\partial z_j^{(1)}}{\partial w_{ij}^{(1)}} \\
&= L'(y_{\text{true}}, a^{(2)}) \cdot f_2'(z^{(2)}) \cdot w_j^{(2)} \cdot f_1'(z^{(1)})
\end{aligned}
$$

Then, in matrix form

$$
\begin{aligned}
\frac{\partial L}{\partial w^{(1)}} &= a^{(0)} \cdot (w^{(2)} \cdot \eta \odot f_1'(z^{(1)}))^T \\
&= a^{(0)} \cdot (w^{(2)} \cdot \eta \odot (\sigma(z^{(1)}) \odot (1 - \sigma(z^{(1)}))))^T \\
\frac{\partial L}{\partial b^{(1)}} &= (w^{(2)} \cdot \eta \odot f_1'(z^{(1)}))^T \\
&= (w^{(2)} \cdot \eta \odot (\sigma(z^{(1)}) \odot (1 - \sigma(z^{(1)}))))^T
\end{aligned}
$$

› Q2: Implementation

In this part, you need to construct a neural network model and run a test experiment. We provide a skeleton script of for the model and full script for the test experiment.

[ ] ↳ *10 cells hidden*

Q3: Real Data Experiments

In this part, you need to try out different model parameter values and observe how they affect the results.

For each of the questions below, implement experiments and insert performance scores to the designated dictionary. The performance scores can be computed using the imported functions (for F1 score, you need to specify `average = "macro"` when calling the function). You can refer to Q2.4 as an example of implementing experiments.

Note: Remember to initialize a new instance of your model for each different choice of hyper-parameter.

[ ] ↳ *10 cells hidden*

Q4: Follow-up Questions

For each question below, provide a short answer. You can cite your code if needed.

Q4.1: Briefly describe the workflow of how your model classifies the data.

↳ *1 cell hidden*

Q4.2: In your own words, explain how the forward propagation in your model works.

↳ *1 cell hidden*

Q4.3: In your own words, explain how the backward propagation in your model works.

↳ *1 cell hidden*

Q4.4: In theory, how do the total number of epochs, the learning rate, and the regularization parameter impact the performance of model? Does any of the theoretical impact actually happen in your result? If so, point them out.

[Answer]

Epochs When the number is too small, the model doesn't learn enough and hence it cannot predict the test data well. When the number is too large, the model tends to overfit the training data, also resulting in poor performance on the test data.

Learning Rate Learning rate is the step size of the gradient descent. Therefore, if the value is too small, it may take the model longer to fit the data. On the other hand, if the value is too large, the model may "overstep" and hence cannot reach the optimal point.

Regularization Parameter The role of regularization is to prevent overfitting. If the parameter is too small, then it cannot contribute enough to the weight update and hence may not be able to prevent overfitting. On the other hand, if the value is too large, it may contribute too much to the update, which prevents the model from learning.