Homework 4 Neural Networks

Instructions

Answer the questions and upload your answers to courseville. Answers can be in Thai or English. Answers can be either typed or handwritten and scanned. the assignment is divided into several small tasks. Each task is weighted equally (marked with T). For this assignment, each task is awarded equally. There are also optional tasks (marked with OT) counts for half of the required task.

The Basics

In this section, we will review some of the basic materials taught in class. These are simple tasks and integral to the understanding of deep neural networks, but many students seem to misunderstand.

T1

Compute the forward and backward pass of the following computation. Note that this is a simplified residual connection.

$$\begin{aligned} x_1 &= ReLU(x_0 * w_0 + b_0) \\ y_1 &= x_1 * w_1 + b_1 \\ z &= ReLU(y_1 + x_0) \end{aligned}$$

Let $x_0 = 1.0, w_0 = 0.3, w_1 = -0.2, b_0 = 0.1, b_1 = -0.3.$ Find the gradient of z with respect to w_0, w_1, b_0, and b_1

Answer

$$\begin{aligned} u_0 &= x_0 * w_0 \\ y_0 &= u_0 + b_0 \\ x_1 &= ReLU(y_0) \\ \\ u_1 &= x_1 * w_1 \\ y_1 &= u_1 + b_1 \\ v_1 &= y_1 + x + 0 \\ z &= ReLU(v_1) \end{aligned}$$

Forward pass

$$\begin{aligned}
u_0 &= 1.0 * 0.3 = 0.3 \\
y_0 &= 0.3 + 0.1 = 0.4 \\
x_1 &= ReLU(0.4) = 0.4 \\
\\
u_1 &= 0.4 * -0.2 = -0.8 \\
y_1 &= -0.08 + -0.3 = -0.38 \\
v_1 &= -0.38 + 1.0 = 0.62 \\
z &= ReLU(0.62) = 0
\end{aligned}$$

Backward pass

$$\begin{aligned}
\frac{\partial z}{\partial w_0} &= \frac{\partial ReLU(v_1)}{\partial v_1}\frac{\partial v_1}{\partial y_1}\frac{\partial y_1}{\partial u_1}\frac{\partial u_1}{\partial x_1}\frac{\partial x_1}{\partial y_0}\frac{\partial y_0}{\partial u_0}\frac{\partial u_0}{\partial w_0} \\
&= 1(v_1 > 0) \cdot (1) \cdot (1) \cdot (w_1) \cdot 1(y_0 > 0) \cdot (1) \cdot (x_0) \\
&= -0.2
\end{aligned}$$

$$\begin{aligned}
\frac{\partial z}{\partial w_1} &= \frac{\partial ReLU(v_1)}{\partial v_1}\frac{\partial v_1}{\partial y_1}\frac{\partial y_1}{\partial u_1}\frac{\partial u_1}{\partial w_1} \\
&= 1(v_1 > 0) \cdot (1) \cdot (1) \cdot (x_1) \\
&= 0.4
\end{aligned}$$

$$\begin{aligned}
\frac{\partial z}{\partial b_0} &= \frac{\partial ReLU(v_1)}{\partial v_1}\frac{\partial v_1}{\partial y_1}\frac{\partial y_1}{\partial u_1}\frac{\partial u_1}{\partial x_1}\frac{\partial x_1}{\partial y_0}\frac{\partial y_0}{\partial b_0} \\
&= 1(v_1 > 0) \cdot (1) \cdot (1) \cdot (w_1) \cdot 1(y_0 > 0) \cdot (1) \\
&= -0.2
\end{aligned}$$

$$\begin{aligned}
\frac{\partial z}{\partial b_1} &= \frac{\partial ReLU(v_1)}{\partial v_1}\frac{\partial v_1}{\partial y_1}\frac{\partial y_1}{\partial b_1} \\
&= 1(v_1 > 0) \cdot (1) \cdot (1) \\
&= 1
\end{aligned}$$

T2

Given the following network architecture specifications, determine the size of the output A, B, and C.

Answer

Output size A: 1024 x 32

Output size B: 512 x 32

Output size C: 1 x 32

T3

What is the total number of learnable parameters in this network? (Don't forget the bias term)

Answer

First hidden layer:

Learnable parameters = #node + #weight = 1024 + (30 x 1024) = 31744

Second hidden layer:

Learnable parameters = #node + #weight = 512 + (512 x 1024) = 524800

Output:

Learnable parameters = #node + #weight = 1 + (1 x 512) = 513

Total = 557057

Deep Learning from (almost) scratch

In this section we will code simple a neural network model from scratch (numpy). However, before we go into coding let's start with some loose ends, namely the gradient of the softmax layer

Recall in class we define the softmax layer as:

$$P(y = j) = \frac{exp(h_j)}{\Sigma_k exp(h_k)}$$

where h_j is the output of the previous layer for class index j The cross entropy loss is defined as:

$$L = -\Sigma_j y_j log P(y = j)$$

where y_j is 1 if y is class j, and 0 otherwise.

T4.

Prove that the derivative of the loss with respect to h_i is $P(y = i) - y_i$. In other words, find $\frac{\partial L}{\partial hi}$ for $i \in 0, \ldots, N - 1$ where N is the number of classes.

Hint: first find $\frac{\partial P(y=j)}{\partial h_i}$ for the case where $j = i$, and the case where $j \neq i$. Then, use the results with chain rule to find the derivative of the loss.

Next, we will code a simple neural network using numpy. Use the starter code **hw4.zip** on github. There are 8 tasks you need to complete in the starter code.

Answer

If $i = j$

let $o_j = \frac{exp(h_j)}{\Sigma_k exp(h_k)}$

$$\begin{aligned}\frac{\partial o_j}{\partial h_i} &= \frac{\partial \frac{exp(h_i)}{\Sigma_k exp(h_k)}}{\partial h_i} \\ &= o_i(1-o_i)\end{aligned}$$

If $i \neq j$

$$\begin{aligned}\frac{\partial o_j}{\partial h_i} &= \frac{\partial \frac{exp(h_j)}{\Sigma_k exp(h_k)}}{\partial h_i} \\ &= -o_i o_j\end{aligned}$$

And

$$\begin{aligned}\frac{\partial L}{\partial o_j} &= \frac{-\Sigma_j y_j log(o_j)}{\partial o_j} \\ &= -\Sigma_j y_j \frac{log(o_j)}{\partial o_j} \\ &= -\Sigma_j \frac{y_j}{o_j}\end{aligned}$$

So if $i = j$

$$\begin{aligned}\frac{\partial L}{\partial h_i} &= \frac{-\Sigma_i y_j log(o_i)}{\partial o_i} \frac{o_i}{\partial h_i} \\ &= -\Sigma_i \frac{y_i}{o_i} o_i(1-o_i) \\ &= y_i o_i - y_i\end{aligned}$$

And if $i \neq j$

$$\begin{aligned}\frac{\partial L}{\partial h_i} &= \frac{-\Sigma_j y_j log(o_j)}{\partial o_j} \frac{o_j}{\partial h_i} \\ &= -\Sigma_j y_j \frac{1}{o_j} \frac{o_j}{\partial h_i} \\ &= -y_i(1-o_i) - \Sigma_{j \neq i} y_j \frac{1}{o_j}(-o_j o_i) \\ &= -y_i + y_i p_i + \Sigma_{j \neq i} y_j o_i \\ &= -y_i + p_i \Sigma_j y_j \\ &= -y_i + p_i\end{aligned}$$

P.S. $\Sigma_j y_j = 1$ since it's one-hot output