Syed Muhammad Hamza Shah

CSC 546

Homework 4

Professor Liang

1. Backpropagation in A Neural Network

$\partial L/\partial w2$:

$\partial L/\partial w2 = \partial L/\partial h3 * \partial h3/\partial h2 * \partial h2/\partial w2 = (\partial L/\partial h3) * f3' * w4 * f2' * x$

$\partial L/\partial w3$:

$\partial L/\partial w3 = \partial L/\partial h3 * \partial h3/\partial w3 = (\partial L/\partial h3) * f3' * h1$

$\partial L/\partial w4$:

$\partial L/\partial w4 = \partial L/\partial h3 * \partial h3/\partial w4 = (\partial L/\partial h3) * f3' * h2$

$\partial L/\partial b1$:

$\partial L/\partial b1 = \partial L/\partial h3 * \partial h3/\partial h1 * \partial h1/\partial b1 = (\partial L/\partial h3) * f3' * w3 * f1'$

$\partial L/\partial b2$:

$\partial L/\partial b2 = \partial L/\partial h3 * \partial h3/\partial h2 * \partial h2/\partial b2 = (\partial L/\partial h3) * f3' * w4 * f2'$

$\partial L/\partial b3$:

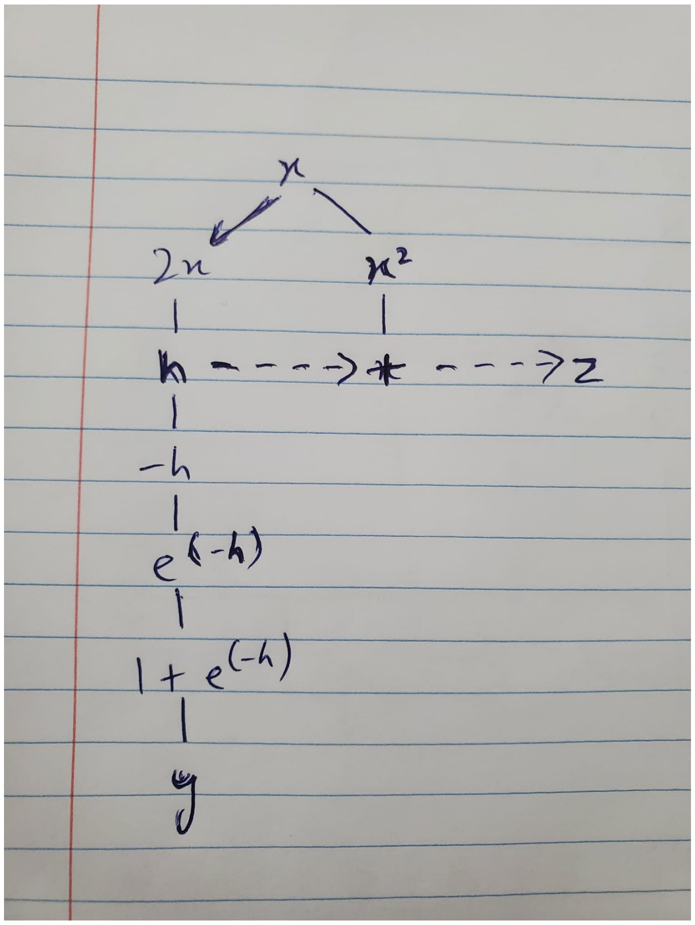
$\partial L/\partial b3 = \partial L/\partial h3 * \partial h3/\partial b3 = (\partial L/\partial h3) * f3'$

$\partial L/\partial x$:

$\partial L/\partial x$ = $\partial L/\partial h3$ * ($\partial h3/\partial h1$ * $\partial h1/\partial x$ + $\partial h3/\partial h2$ * $\partial h2/\partial x$)

= ($\partial L/\partial h3$) * ($f3'$ * $w3$ * $f1'$ * $w1$ + $f3'$ * $w4$ * $f2'$ * $w2$)

2. Computational Graph (20 points)



We have two paths from x to y:

1. x -> 2x -> h -> -h -> e^(-h) -> 1 + e^(-h) -> y
2. x -> x^2 -> z

To compute ∂y/∂x, we will consider both paths:

Path 1: ∂y/∂x = ∂y/∂h * ∂h/∂x

∂y/∂h = (-1) * e^(-h) / (1 + e^(-h))^2

$\partial h/\partial$x = 2

∂y/∂x (Path 1) = ∂y/∂h * $\partial h/\partial$x = (-1) * e^(-h) / (1 + e^(-h))^2 * 2 = -2 * e^(-h) / (1 + e^(-h))^2

Path 2 does not contribute to ∂y/∂x since there is no direct connection between z and y in the graph.

Therefore, ∂y/∂x = -2 * e^(-h) / (1 + e^(-h))^2

3. Target (output) Normalization for a Neural Network

Output normalization can be beneficial for this task, especially when the scales of the different output components vary significantly. In this case, the monthly income ranges from 0 to 10,000, while the age ranges from 0 to 100. The difference in scale may cause the training algorithm to focus more on minimizing the error in the larger-scale component (monthly income) while neglecting the error in the smaller-scale component (age). This can lead to suboptimal performance in the model.

Normalizing the outputs will help the training algorithm to focus on both components of the output vector equally, potentially improving the model's overall performance.

A possible normalization technique that can be applied is min-max scaling, which scales the outputs to a range of 0 to 1:

normalized_output = (original_output - min_output) / (max_output - min_output)

For y(1) (monthly income):

normalized_y(1) = (y(1) - 0) / (10000 - 0)

For y(2) (age):

normalized_y(2) = (y(2) - 0) / (100 - 0)

After normalization:

original_output = normalized_output * (max_output - min_output) + min_output

4. Activation Functions for Regression

(1) To ensure that the final output y is nonnegative ($y \geq 0$), we can use the Rectified Linear Unit (ReLU) activation function:

$y = f(z) = \max(0, z)$

(2) To ensure that the final output y is nonpositive ($y \leq 0$), we can use a modified version of the Rectified Linear Unit (ReLU) activation function:

$y = f(z) = \min(0, z)$

(3) To ensure that the final output y is within the range $a \le y \le b$, we can use a scaled and shifted version of the Sigmoid activation function. First, we normalize the Sigmoid function to fit the desired range:

normalized_sigmoid(z) = (b - a) * (1 / (1 + e^(-z))) + a

Then, the activation function $y = f(z)$ is:

$y = f(z)$ = normalized_sigmoid(z)

5. Normalization Inside a Neural Network

(1) Batch Normalization (BN) becomes unstable with small batch sizes because it relies on batch-wise mean and variance estimates, which become less accurate and more noisy with smaller batches.

(2) Layer Normalization (LN) is independent of batch size as it normalizes activations within each individual sample, making it stable and less sensitive to batch size.

6. Skip Connections in a Deep Neural Network

Skip connections help build deep networks by mitigating the vanishing gradient problem, easing optimization, improving information flow between layers, and enhancing feature reuse. These factors collectively contribute to better training and performance of deep networks.

7. Randomness of a Neural Network

The randomness in training the same network multiple times arises from factors like weight initialization, mini-batch selection, regularization techniques, and data augmentation. When reporting in a paper, it's preferable to report the average performance of multiple runs or use an ensemble of models to provide a fair representation of the model's performance.