A diagram of a machine

Description automatically generated

Solve for forward and backward propagation at unit 1. Assume teacher forcing.

Given:

H = [1, 4, -3, 2]

O = [0.3, -0.1, 0.9, 0.2]

Assume W1 and W2 are weight matrices.

W3 \* softmax([1,0,0,0]\*W1 + [0.3, -0.1, 0.9, 0.2] \*W2) = [1, 4, -3, 2]

W3 = ?

## Answer key:

### Forward pass

W1 ≈ | 2.2473 36.5192 -10.0749 13.5214 |

To solve for the weight matrix W3 in the equation:

W3 \* softmax([1, 0, 0, 0] \* W1 + [0.3, -0.1, 0.9, 0.2] \* W2) = [1, 4, -3, 2]

Given that W1 and W2 are identity matrices, we can simplify the equation as follows:

W3 \* softmax([1, 0, 0, 0] + [0.3, -0.1, 0.9, 0.2]) = [1, 4, -3, 2]

Now, let's calculate the softmax of the vector [1.3, -0.1, 0.9, 0.2]:

softmax([1.3, -0.1, 0.9, 0.2]) = [e^1.3, e^(-0.1), e^0.9, e^0.2] / (e^1.3 + e^(-0.1) + e^0.9 + e^0.2)

Let's compute the values for this softmax:

softmax([1.3, -0.1, 0.9, 0.2]) ≈ [e^1.3, e^(-0.1), e^0.9, e^0.2] / (e^1.3 + e^(-0.1) + e^0.9 + e^0.2) ≈ [3.669, 0.9048, 2.4596, 1.2214] / (8.2548)

Now, we can rewrite the equation as:

W3 \* [3.669/8.2548, 0.9048/8.2548, 2.4596/8.2548, 1.2214/8.2548] = [1, 4, -3, 2]

Now, we can solve for W3 by multiplying both sides of the equation by the inverse of the softmax result:

W3 = [1, 4, -3, 2] \* [8.2548/3.669, 8.2548/0.9048, 8.2548/2.4596, 8.2548/1.2214]

Now, calculate the product:

W3 ≈ [1, 4, -3, 2] \* [2.2473, 9.1298, 3.3583, 6.7607]

Perform element-wise multiplication:

W3 ≈ [2.2473, 36.5192, -10.0749, 13.5214]

### Backward pass

To update the weight matrices W1, W2, and W3 using backpropagation with a mean squared error (MSE) loss, you can follow these steps. We'll assume a learning rate of 1.

Given:

- W3 = [2.2473, 36.5192, -10.0749, 13.5214]

- W1 and W2 are identity matrices of size 4.

- Gold output = [0.3, 0.13, 0, 0.8]

- Computed output = [1, 4, -3, 2]

\*\*Step 1: Compute the error (loss):\*\*

We'll compute the squared error loss (MSE) between the gold output and the computed output.

Loss = 1/2 \* Σ(gold - computed)^2

Loss = 1/2 \* [(0.3 - 1)^2 + (0.13 - 4)^2 + (0 - (-3))^2 + (0.8 - 2)^2]

Loss ≈ 1/2 \* [0.49 + 12.09 + 9 + 0.04] ≈ 10.81

\*\*Step 2: Backpropagation for W3:\*\*

The update for W3 is based on the gradient of the loss with respect to W3. We'll use the chain rule to compute this gradient.

dLoss/dW3 = (dLoss/dOutput) \* (dOutput/dW3)

The gradient dLoss/dOutput can be computed as (computed - gold).

dLoss/dOutput = [0.7, -3.87, -3, -1.2]

Now, let's compute dOutput/dW3:

dOutput/dW3 = (dOutput/dh) \* (dh/dW3)

Where dh is the hidden state, and dh/dW3 is the hidden-to-output weight.

We'll assume that the hidden state dh is 1 for this calculation.

dOutput/dW3 = dh \* (dOutput/dW3) = [1, 1, 1, 1] \* [2.2473, 36.5192, -10.0749, 13.5214]

dLoss/dW3 = [0.7, -3.87, -3, -1.2] \* [2.2473, 36.5192, -10.0749, 13.5214]

dLoss/dW3 ≈ [1.72411, -141.63704, 30.2247, 16.22568]

Now, we can update W3 using the gradient descent update rule:

W3\_new = W3 - (learning\_rate) \* dLoss/dW3

W3\_new = [2.2473, 36.5192, -10.0749, 13.5214] - 1 \* [1.72411, -141.63704, 30.2247, 16.22568]

W3\_new ≈ [0.52319, 178.15624, -40.2996, -2.70468]

\*\*Step 3: Backpropagation for W1 and W2:\*\*

Since W1 and W2 are identity matrices, their gradients are straightforward.

dLoss/dW1 = (dLoss/dOutput) \* (dOutput/dh) \* (dh/dW1)

dLoss/dW1 = [0.7, -3.87, -3, -1.2] \* [1, 1, 1, 1] \* W1 = [0.7, -3.87, -3, -1.2]

dLoss/dW2 = [0.7, -3.87, -3, -1.2] \* [1, 1, 1, 1] \* W2 = [0.7, -3.87, -3, -1.2]

The updated W1 and W2 will be the same as their original values since they are identity matrices.

W1\_new = W1 - 1 \* [0.7, -3.87, -3, -1.2] = W1 - [0.7, -3.87, -3, -1.2]

W2\_new = W2 - 1 \* [0.7, -3.87, -3, -1.2] = W2 - [0.7, -3.87, -3, -1.2]

So, W1\_new and W2\_new are the same as W1 and W2, respectively.