Neural Networks hw1

Dylan Satow

October 2025

1 Written Problems

1.1 Hard-Coding Neural Networks

The goal is to create a neural network that outputs 1 if the input sequence (x_1, x_2, x_3, x_4) is a superincreasing sequence, and 0 otherwise. A sequence is superincreasing if every element is greater than the sum of its predecessors. For a 4-element sequence, this translates to the following three conditions being true simultaneously:

1. $x_2 > x_1$
2. $x_3 > x_1 + x_2$
3. $x_4 > x_1 + x_2 + x_3$

We can design the network to check each of these conditions in one of the three hidden units, and then use the output unit to perform a logical AND on the results from the hidden units.

Activation Functions For both the hidden layer (ϕ_1) and the output layer (ϕ_2), we will use the hard threshold activation function, defined in the assignment as:

$$\phi(z) = I(z \geq 0) = \begin{cases} 1 & \text{if } z \geq 0 \\ 0 & \text{if } z < 0 \end{cases}$$

Hidden Layer Each hidden unit h_k will compute $h_k = \phi_1(\mathbf{w}_k^{(1)} \cdot \mathbf{x} + b_k^{(1)})$. We want $h_k = 1$ if the k^{th} condition is met, and $h_k = 0$ otherwise.

- For h_1 (checks $x_2 > x_1 \iff x_2 - x_1 > 0$): We need the input to the activation, z_1, to be ≥ 0 if and only if this condition holds. We can set $z_1 = x_2 - x_1$. This gives the weights $\mathbf{w}_1^{(1)} = [-1, 1, 0, 0]$ and bias $b_1^{(1)} = 0$.
- For h_2 (checks $x_3 > x_1 + x_2 \iff x_3 - x_1 - x_2 > 0$): We set $z_2 = x_3 - x_1 - x_2$. This gives the weights $\mathbf{w}_2^{(1)} = [-1, -1, 1, 0]$ and bias $b_2^{(1)} = 0$.

- For h_3 (checks $x_4 > x_1 + x_2 + x_3 \iff x_4 - x_1 - x_2 - x_3 > 0$): We set $z_3 = x_4 - x_1 - x_2 - x_3$. This gives the weights $\mathbf{w}_3^{(1)} = [-1, -1, -1, 1]$ and bias $b_3^{(1)} = 0$.

Output Layer The output unit y computes $y = \phi_2(\mathbf{w}^{(2)} \cdot \mathbf{h} + b^{(2)})$. We need $y = 1$ if and only if $h_1 = 1$, $h_2 = 1$, and $h_3 = 1$. This is a logical AND operation. The input to the output activation is $z_{out} = w_1^{(2)} h_1 + w_2^{(2)} h_2 + w_3^{(2)} h_3 + b^{(2)}$. Let's set the weights $\mathbf{w}^{(2)} = [1, 1, 1]$. Then $z_{out} = h_1 + h_2 + h_3 + b^{(2)}$.

- If all hidden units are 1, their sum is 3. We need $z_{out} = 3 + b^{(2)} \geq 0$.
- If any hidden unit is 0, the maximum sum is 2. We need $z_{out} = 2 + b^{(2)} < 0$.

From these two conditions, we need $b^{(2)} \geq -3$ and $b^{(2)} < -2$. A value such as $b^{(2)} = -2.5$ would work. For an integer solution, if we choose $b^{(2)} = -3$, the first condition becomes $3 - 3 = 0 \geq 0$ (output 1) and the second becomes $2 - 3 = -1 < 0$ (output 0). This works perfectly.

Final Parameters

- **Activation Functions**: $\phi_1(z) = \phi_2(z) = I(z \geq 0)$.
- **Hidden Layer Weights**:

$$W^{(1)} = \begin{bmatrix} -1 & 1 & 0 & 0 \\ -1 & -1 & 1 & 0 \\ -1 & -1 & -1 & 1 \end{bmatrix}$$

- **Hidden Layer Biases**:

$$\mathbf{b}^{(1)} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

- **Output Layer Weights**:

$$\mathbf{w}^{(2)} = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}$$

- **Output Layer Bias**:

$$b^{(2)} = -3$$

1.2 Backpropagation

Computation Graph

The computation graph describes the flow of data from inputs to the final loss. The nodes in the graph are the variables involved in the computation, and the directed edges represent the functions that transform one variable into another.

- **Inputs**: x (features), t (true labels), and all weight matrices and biases $(W^{(1)}, b^{(1)}, W^{(2)}, b^{(2)}, W^{(g)}, b^{(g)}, W^{(3)}, W^{(4)})$.
- **Graph Structure**:
  1. $x, W^{(1)}, b^{(1)} \to a_1 = W^{(1)}x + b^{(1)}$
  2. $x, W^{(2)}, b^{(2)} \to a_2 = W^{(2)}x + b^{(2)}$
  3. $a_1 \to c_1 = \sigma(a_1)$ (Sigmoid activation, where $\sigma(a) = 1/(1 + e^{-a})$)
  4. $a_1, a_2 \to u = a_1 + a_2$
  5. $u \to c_2 = \text{ReLU}(u)$
  6. $c_1, c_2, W^{(g)}, b^{(g)} \to g = W^{(g)}[c_1; c_2] + b^{(g)}$ (Concatenation)
  7. $g, x, W^{(3)}, W^{(4)} \to y = W^{(3)}g + W^{(4)}x$
  8. $y \to y' = \text{softmax}(y)$
  9. $y', t \to S = \sum_k I(t_k = 1) \log(y'_k)$
  10. $S \to J = -S$

The graph shows multiple paths from the input x to the loss J: one through a_1, one through a_2, and a direct one to y. The total gradient $\frac{\partial J}{\partial x}$ will be the sum of the gradients from these three paths.

Backward Pass

We derive the backpropagation equations by applying the chain rule, starting from the output J and moving backward through the graph. We compute the gradient of the cost J with respect to each variable. Let $\delta_v = \frac{\partial J}{\partial v}$ denote the gradient of the loss with respect to a variable v.

1. **Loss**: $J = -S$
$$\frac{\partial J}{\partial S} = -1$$
2. **Score**: $S = \sum_k t_k \log(y'_k)$ (assuming t is one-hot, so $I(t_k = 1)$ is t_k) The gradient of the cross-entropy loss with respect to the softmax input y is a standard result:
$$\delta_y = \frac{\partial J}{\partial y} = y' - t$$
3. **Output Logits**: $y = W^{(3)}g + W^{(4)}x$. From here, we can compute gradients for $g, x, W^{(3)}, W^{(4)}$.
$$\delta_g = \frac{\partial J}{\partial g} = (W^{(3)})^T \delta_y$$
$$\delta_x^{(\text{from y})} = (W^{(4)})^T \delta_y$$
The gradients for the weights are:
$$\delta_{W^{(3)}} = \delta_y g^T$$
$$\delta_{W^{(4)}} = \delta_y x^T$$

4. **Hidden Layer g**: $g = W^{(g)}[c_1; c_2] + b^{(g)}$. Let $c_{concat} = [c_1; c_2]$.

$$\delta_{c_{concat}} = \frac{\partial J}{\partial c_{concat}} = (W^{(g)})^T \delta_g$$

This gradient is then split for c_1 and c_2:

$$\delta_{c_1} = (\delta_{c_{concat}})_{1:d_1} \quad \text{and} \quad \delta_{c_2} = (\delta_{c_{concat}})_{d_1+1:end}$$

(where d_1 is the dimension of c_1). The gradients for the weights are:

$$\delta_{W^{(g)}} = \delta_g c_{concat}^T$$

$$\delta_{b^{(g)}} = \delta_g$$

5. **ReLU Activation**: $c_2 = \text{ReLU}(u)$

$$\delta_u = \frac{\partial J}{\partial u} = \delta_{c_2} \odot I(u > 0) \quad (\odot \text{ is element-wise product})$$

6. **Sum**: $u = a_1 + a_2$

$$\delta_{a_1}^{(\text{from u})} = \delta_u \quad \text{and} \quad \delta_{a_2} = \delta_u$$

7. **Sigmoid Activation**: $c_1 = \sigma(a_1)$

$$\delta_{a_1}^{(\text{from c1})} = \delta_{c_1} \odot (\sigma(a_1)(1 - \sigma(a_1))) = \delta_{c_1} \odot (c_1(1 - c_1))$$

8. **Total gradients for** a_1, a_2:

$$\delta_{a_1} = \delta_{a_1}^{(\text{from c1})} + \delta_{a_1}^{(\text{from u})} = (\delta_{c_1} \odot (c_1(1 - c_1))) + \delta_u$$

$$\delta_{a_2} = \delta_u$$

9. **Linear Layers** a_1, a_2: For $a_1 = W^{(1)}x + b^{(1)}$:

$$\delta_x^{(\text{from a1})} = (W^{(1)})^T \delta_{a_1}$$

$$\delta_{W^{(1)}} = \delta_{a_1} x^T$$

$$\delta_{b^{(1)}} = \delta_{a_1}$$

For $a_2 = W^{(2)}x + b^{(2)}$:

$$\delta_x^{(\text{from a2})} = (W^{(2)})^T \delta_{a_2}$$

$$\delta_{W^{(2)}} = \delta_{a_2} x^T$$

$$\delta_{b^{(2)}} = \delta_{a_2}$$

10. **Total Gradient for x**: The final gradient for x is the sum from all paths:

$$\delta_x = \frac{\partial J}{\partial x} = \delta_x^{(\text{from y})} + \delta_x^{(\text{from a1})} + \delta_x^{(\text{from a2})}$$

$$\frac{\partial J}{\partial x} = (W^{(4)})^T (y' - t) + (W^{(1)})^T \delta_{a_1} + (W^{(2)})^T \delta_{a_2}$$