MLE Sheet 4

May 2025

Exercise 1: Backpropagation for a Classification Task

Task 1

Softmax function:

$$\hat{y}_i = \frac{e^{\tilde{z}_i^{(L)}}}{\sum_{m=1}^{C} e^{\tilde{z}_m^{(L)}}} = \sigma(\tilde{\boldsymbol{z}}^{(L)})_i$$

Cross-entropy loss:

$$\mathcal{L}(\hat{\boldsymbol{y}}, \boldsymbol{y}) = -\sum_{i=1}^{C} y_i \log(\hat{y}_i)$$

Here, $\boldsymbol{y} \in \{0,1\}^C$ is a one-hot encoded vector of true class labels.
We want to compute:

$$\frac{\partial \mathcal{L}}{\partial \tilde{z}_j^{(L)}} = \sum_{i=1}^{C} \frac{\partial \mathcal{L}}{\partial \hat{y}_i} \cdot \frac{\partial \hat{y}_i}{\partial \tilde{z}_j^{(L)}}$$

We take derivative of the Loss w.r.t Softmax Output

$$\frac{\partial \mathcal{L}}{\partial \hat{y}_i} = -\frac{y_i}{\hat{y}_i}$$

We take derivative of Softmax w.r.t Preactivation

$$\frac{\partial \hat{y}_i}{\partial \tilde{z}_j^{(L)}} = \hat{y}_i(\delta_{ij} - \hat{y}_j)$$

where δ_{ij} is the Kronecker delta.
Using Chain Rule

$$\frac{\partial \mathcal{L}}{\partial \tilde{z}_j^{(L)}} = \sum_{i=1}^{C} \left(-\frac{y_i}{\hat{y}_i} \right) \cdot \hat{y}_i (\delta_{ij} - \hat{y}_j)$$
$$= -\sum_{i=1}^{C} y_i (\delta_{ij} - \hat{y}_j)$$

Since $\boldsymbol{y}$ is one-hot encoded, only the true class index k has $y_k = 1$, and all other $y_i = 0$. Therefore, the sum reduces to:

$$-(\delta_{kj} - \hat{y}_j) = \hat{y}_j - y_j$$

Therefore,

$$\delta^{(L)} := \nabla_{\tilde{\boldsymbol{z}}^{(L)}} \mathcal{L} = \hat{\boldsymbol{y}} - \boldsymbol{y}$$

This concludes the derivation of the initial step in the backpropagation recursion.

Task 2

From Task 1, we know that the gradient of the loss with respect to the preactivation of the output layer is:

$$\delta^{(L)} := \nabla_{\tilde{\boldsymbol{z}}^{(L)}} \mathcal{L} = \hat{\boldsymbol{y}} - \boldsymbol{y}$$

We now have to compute the gradients of the loss with respect to the parameters of the last hidden layer, i.e., with respect to $W^{(L-1)}$ and $\boldsymbol{b}^{(L-1)}$.

(a) Gradient with Respect to the Weights $W^{(L-1)}$

Using the backpropagation rule:

$$\nabla_{W^{(l)}} \mathcal{L} = \delta^{(l+1)} \cdot (\boldsymbol{z}^{(l)})^\top$$

for $l = L - 1$, we get:

$$\nabla_{W^{(L-1)}} \mathcal{L} = \delta^{(L)} \cdot (\boldsymbol{z}^{(L-1)})^\top$$

Dimensions

- $\delta^{(L)} \in \mathbb{R}^C$
- $\boldsymbol{z}^{(L-1)} \in \mathbb{R}^{d_{L-1}}$
- So, $\nabla_{W^{(L-1)}} \mathcal{L} \in \mathbb{R}^{C \times d_{L-1}}$

This matches the dimensions of the weight matrix $W^{(L-1)} \in \mathbb{R}^{C \times d_{L-1}}$.

(b) Gradient with Respect to the Biases $\boldsymbol{b}^{(L-1)}$

Using the backpropagation rule:

$$\nabla_{\boldsymbol{b}^{(l)}} \mathcal{L} = \delta^{(l+1)}$$

for $l = L-1$, we obtain:

$$\nabla_{\boldsymbol{b}^{(L-1)}} \mathcal{L} = \delta^{(L)}$$

Dimensions

- $\delta^{(L)} \in \mathbb{R}^C \Rightarrow \nabla_{\boldsymbol{b}^{(L-1)}} \mathcal{L} \in \mathbb{R}^C$

  This matches the dimensions of the bias vector $\boldsymbol{b}^{(L-1)} \in \mathbb{R}^C$.

Task 3

To compute the error signal for the last hidden layer (layer $L-1$), we use the backpropagation recursion formula:

$$\delta^{(L-1)} = \left(W^{(L-1)\top} \delta^{(L)}\right) \odot \phi' \left(\tilde{\boldsymbol{z}}^{(L-1)}\right)$$

From Task 1, we know:

$$\delta^{(L)} = \hat{\boldsymbol{y}} - \boldsymbol{y}$$

Substituting this into the formula:

$$\delta^{(L-1)} = \left(W^{(L-1)\top} (\hat{\boldsymbol{y}} - \boldsymbol{y})\right) \odot \phi' \left(\tilde{\boldsymbol{z}}^{(L-1)}\right)$$

ReLU Activation and Its Derivative

We now assume the activation function ϕ is the ReLU function:

$$\phi(x) = \max(0, x)$$

with derivative:

$$\phi'(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{if } x \leq 0 \end{cases}$$

Effect on Gradient Flow

Consider the j-th component $\delta_j^{(L-1)}$. If the preactivation $\tilde{z}_j^{(L-1)} < 0$, then:

$$\phi' \left(\tilde{z}_j^{(L-1)}\right) = 0 \quad \Rightarrow \quad \delta_j^{(L-1)} = 0$$

This means that the neuron j in layer $L-1$ does not contribute to the gradient update, effectively blocking the gradient from flowing backward through it.

Even at the boundary point $\tilde{z}_j^{(L-1)} = 0$, we adopt the common convention $\phi'(0) = 0$, so $\delta_j^{(L-1)} = 0$ holds there as well.

Conclusion

The use of the ReLU activation introduces sparsity in the backward pass: only those neurons that were *active* (i.e., had positive preactivations) during the forward pass can propagate error gradients backward. This has two important consequences:

- It can improve computational efficiency and reduce interference between gradients.
- However, it also risks the "dying ReLU" problem: if a neuron consistently receives negative inputs, it never activates, and its weights are never updated.

This insight explains why ReLU networks may require careful initialization and learning rate tuning in practice.