Mathematical Foundations of a Two-Layer Neural Network

Lecture Notes

1 Objective

This lecture covers the mathematical operations underlying a **two-layer fully connected neural network** for supervised learning. We explain:

- Forward pass computations,
- Loss function,
- Backpropagation (gradients),
- Weight updates using gradient descent.

2 Network Architecture and Notation

We process inputs $\mathbf{X} \in \mathbb{R}^{N \times D_{in}}$ to produce outputs $\hat{\mathbf{Y}} \in \mathbb{R}^{N \times D_{out}}$.

- N: Number of examples (batch size)
- D_{in}: Number of input features per example
- H: Number of neurons in the hidden layer
- D_{out}: Number of output dimensions

Parameter Matrices

- First layer weights: $\mathbf{W}_1 \in \mathbb{R}^{D_{in} \times H}$
- Second layer weights: $\mathbf{W}_2 \in \mathbb{R}^{H \times D_{out}}$

3 Forward Pass

3.1 Step 1: Input to Hidden Layer Transformation

$$\mathbf{H} = \mathbf{X} \cdot \mathbf{W}_1$$

- $\mathbf{X} \in \mathbb{R}^{N \times D_{in}}$
- $\mathbf{W}_1 \in \mathbb{R}^{D_{in} \times H}$
- Result: $\mathbf{H} \in \mathbb{R}^{N \times H}$

3.2 Step 2: Activation Function (ReLU)

ReLU applied element-wise:

$$\mathbf{H}_{relu} = \max(0, \mathbf{H})$$

$$\mathbf{H}_{relu} \in \mathbb{R}^{N \times H}$$

3.3 Step 3: Hidden Layer to Output Layer Transformation

$$\hat{\mathbf{Y}} = \mathbf{H}_{relu} \cdot \mathbf{W}_2$$

- $\mathbf{H}_{relu} \in \mathbb{R}^{N \times H}$
- $\mathbf{W}_2 \in \mathbb{R}^{H \times D_{out}}$
- Result: $\hat{\mathbf{Y}} \in \mathbb{R}^{N \times D_{out}}$

4 Loss Function

We measure the difference between the predicted outputs $\hat{\mathbf{Y}}$ and the true outputs $\mathbf{Y}$.

4.1 Sum of Squared Errors (SSE)

$$\mathcal{L} = \sum_{i=1}^{N} \sum_{j=1}^{D_{out}} \left(\hat{Y}_{ij} - Y_{ij} \right)^2$$

Alternatively, in matrix form:

$$\mathcal{L} = \|\hat{\mathbf{Y}} - \mathbf{Y}\|_F^2$$

where $\| \cdot \|_F$ is the Frobenius norm.

5 Backpropagation

5.1 Gradient of Loss with Respect to Predictions

$$\frac{\partial \mathcal{L}}{\partial \hat{\mathbf{Y}}} = 2(\hat{\mathbf{Y}} - \mathbf{Y})$$

$$\text{Shape: } \mathbb{R}^{N \times D_{out}}$$

5.2 Gradient with Respect to $\mathbf{W}_2$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}_2} = \mathbf{H}_{relu}^{\top} \cdot \frac{\partial \mathcal{L}}{\partial \hat{\mathbf{Y}}}$$

$$\text{Shape: } \mathbb{R}^{H \times D_{out}}$$

5.3 Gradient with Respect to Hidden Layer Activations

$$\frac{\partial \mathcal{L}}{\partial \mathbf{H}_{relu}} = \frac{\partial \mathcal{L}}{\partial \hat{\mathbf{Y}}} \cdot \mathbf{W}_2^{\top}$$

$$\text{Shape: } \mathbb{R}^{N \times H}$$

5.4 Apply ReLU Derivative

$$\frac{\partial \mathcal{L}}{\partial \mathbf{H}} = \begin{cases} \frac{\partial \mathcal{L}}{\partial \mathbf{H}_{relu}}, & \text{if } \mathbf{H} > 0 \\ 0, & \text{otherwise} \end{cases}$$

5.5 Gradient with Respect to $\mathbf{W}_1$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}_1} = \mathbf{X}^{\top} \cdot \frac{\partial \mathcal{L}}{\partial \mathbf{H}}$$

$$\text{Shape: } \mathbb{R}^{D_{in} \times H}$$

6 Weight Updates (Gradient Descent)

With learning rate η:

$$\mathbf{W}_1 \leftarrow \mathbf{W}_1 - \eta \cdot \frac{\partial \mathcal{L}}{\partial \mathbf{W}_1}$$

$$\mathbf{W}_2 \leftarrow \mathbf{W}_2 - \eta \cdot \frac{\partial \mathcal{L}}{\partial \mathbf{W}_2}$$

Symbol	Description	Shape
$\mathbf{X}$	Input data	$N \times D_{in}$
$\mathbf{Y}$	Target output	$N \times D_{out}$
$\mathbf{W}_1$	Input $\rightarrow$ Hidden	$D_{in} \times H$
$\mathbf{W}_2$	Hidden $\rightarrow$ Output	$H \times D_{out}$
$\mathbf{H}$	Pre-activation hidden layer	$N \times H$
$\mathbf{H}_{relu}$	Activated hidden layer	$N \times H$
$\hat{\mathbf{Y}}$	Predicted output	$N \times D_{out}$
$\mathcal{L}$	Loss (scalar)	1×1

Table 1: Summary of Shapes in a Two-Layer Neural Network

7 Summary of Dimensions

8 Mathematical Flow Diagram

Forward Pass

$$\mathbf{X} \in \mathbb{R}^{N \times D_{in}} \xrightarrow{\mathbf{W}_1} \mathbf{H} \in \mathbb{R}^{N \times H} \xrightarrow{\text{ReLU}} \mathbf{H}_{relu} \in \mathbb{R}^{N \times H} \xrightarrow{\mathbf{W}_2} \hat{\mathbf{Y}} \in \mathbb{R}^{N \times D_{out}}$$

Backward Pass

$$\hat{\mathbf{Y}} \rightarrow \frac{\partial \mathcal{L}}{\partial \hat{\mathbf{Y}}} \rightarrow \mathbf{W}_2 \rightarrow \mathbf{H}_{relu} \rightarrow \text{ReLU mask} \rightarrow \mathbf{H} \rightarrow \mathbf{W}_1 \rightarrow \mathbf{X}$$

9 Intuition Behind the Math

- Matrix multiplication computes weighted sums across the layers for multiple examples.
- ReLU introduces non-linearity, enabling the network to model complex relationships.
- Gradients are computed using matrix calculus, where transposes align matrix dimensions for the chain rule.
- Gradient descent updates the weights in the direction that reduces the loss.

10 Conclusion

This two-layer network demonstrates the fundamental mathematical principles behind feed-forward neural networks. Mastery of these concepts is essential before advancing to deeper networks, convolutional neural networks, and other architectures.