Chapter 7: Gradients and initialisation

Problem 7.1

A two-layer network with two hidden units in each layer can be defined as:

$$\begin{aligned} y = \phi_0 &+ \phi_1 \mathrm{a}[\psi_{01} + \psi_{11}\mathrm{a}[\theta_{01} + \theta_{11}x] + \psi_{21}\mathrm{a}[\theta_{02} + \theta_{12}x]] \\ &+ \phi_2 \mathrm{a}[\psi_{02} + \psi_{12}\mathrm{a}[\theta_{01} + \theta_{11}x] + \psi_{22}\mathrm{a}[\theta_{02} + \theta_{12}x]] \end{aligned}$$

where the functions $a[\bullet]$ are ReLU functions. Compute the derivatives of the output y with respect to each of the 13 parameters $\boldsymbol{\phi}_\bullet, \boldsymbol{\theta}_{\bullet\bullet}, \boldsymbol{\psi}_{\bullet\bullet}$ directly (i.e., not using the backpropagation algorithm). The derivative of the ReLU function with respect to its input $\partial a[z]/\partial z$ is the indicator function $\mathbb{I}[z > 0]$, which returns one if the argument is greater than zero and zero otherwise.

Derivatives with respect to the $\boldsymbol{\phi}_\bullet$ parameters:

$$\begin{aligned} \frac{\partial y}{\partial \phi_0} &= 1 \\ \frac{\partial y}{\partial \phi_1} &= \mathrm{a}[\psi_{01} + \psi_{11}\mathrm{a}[\theta_{01} + \theta_{11}x] + \psi_{21}\mathrm{a}[\theta_{02} + \theta_{12}x]] \\ \frac{\partial y}{\partial \phi_2} &= \mathrm{a}[\psi_{02} + \psi_{12}\mathrm{a}[\theta_{01} + \theta_{11}x] + \psi_{22}\mathrm{a}[\theta_{02} + \theta_{12}x]] \end{aligned}$$

with respect to the $\boldsymbol{\theta}_{\bullet\bullet}$ parameters:

$$\begin{aligned} \frac{\partial y}{\partial \theta_{01}} &= \phi_1\psi_{11}\mathbb{I}[\theta_{01} + \theta_{11}x > 0] + \phi_2\psi_{12}\mathbb{I}[\theta_{01} + \theta_{11}x > 0] \\ \frac{\partial y}{\partial \theta_{11}} &= x(\phi_1\psi_{11}\mathbb{I}[\theta_{01} + \theta_{11}x > 0] + \phi_2\psi_{12}\mathbb{I}[\theta_{01} + \theta_{11}x > 0]) \\ \frac{\partial y}{\partial \theta_{02}} &= \phi_1\psi_{21}\mathbb{I}[\theta_{02} + \theta_{12}x > 0] + \phi_2\psi_{22}\mathbb{I}[\theta_{02} + \theta_{12}x > 0] \\ \frac{\partial y}{\partial \theta_{12}} &= x(\phi_1\psi_{21}\mathbb{I}[\theta_{02} + \theta_{12}x > 0] + \phi_2\psi_{22}\mathbb{I}[\theta_{02} + \theta_{12}x > 0]) \end{aligned}$$

and with respect to the $\boldsymbol{\psi}_{\bullet\bullet}$ parameters:

$$
\begin{aligned}
\frac{\partial y}{\partial \psi_{01}} &= \phi_1 \mathbb{I}[\psi_{01} + \psi_{11}\text{a}[\theta_{01} + \theta_{11}x] + \psi_{21}\text{a}[\theta_{02} + \theta_{12}x] > 0] \\
\frac{\partial y}{\partial \psi_{11}} &= \phi_1 \text{a}[\theta_{01} + \theta_{11}x]\mathbb{I}[\psi_{01} + \psi_{11}\text{a}[\theta_{01} + \theta_{11}x] + \psi_{21}\text{a}[\theta_{02} + \theta_{12}x] > 0] \\
\frac{\partial y}{\partial \psi_{21}} &= \phi_1 \text{a}[\theta_{02} + \theta_{12}x]\mathbb{I}[\psi_{01} + \psi_{11}\text{a}[\theta_{01} + \theta_{11}x] + \psi_{21}\text{a}[\theta_{02} + \theta_{12}x] > 0] \\
\frac{\partial y}{\partial \psi_{02}} &= \phi_2 \mathbb{I}[\psi_{02} + \psi_{12}\text{a}[\theta_{01} + \theta_{11}x] + \psi_{22}\text{a}[\theta_{02} + \theta_{12}x] > 0]
\end{aligned}
$$

Problem 7.2

Find an expression for the final term in each of the five chains of derivates in equation 7.12:

$$
\begin{aligned}
\frac{\partial l_i}{\partial f_2} &= \frac{\partial h_3}{\partial f_2}\left(\frac{\partial f_3}{\partial h_3}\frac{\partial l_i}{\partial f_3}\right) \\
\frac{\partial l_i}{\partial h_2} &= \frac{\partial f_2}{\partial h_2}\left(\frac{\partial h_3}{\partial f_2}\frac{\partial f_3}{\partial h_3}\frac{\partial l_i}{\partial f_3}\right) \\
\frac{\partial l_i}{\partial f_1} &= \frac{\partial h_2}{\partial f_1}\left(\frac{\partial f_2}{\partial h_2}\frac{\partial h_3}{\partial f_2}\frac{\partial f_3}{\partial h_3}\frac{\partial l_i}{\partial f_3}\right) \\
\frac{\partial l_i}{\partial h_1} &= \frac{\partial f_1}{\partial h_1}\left(\frac{\partial h_2}{\partial f_1}\frac{\partial f_2}{\partial h_2}\frac{\partial h_3}{\partial f_2}\frac{\partial f_3}{\partial h_3}\frac{\partial l_i}{\partial f_3}\right) \\
\frac{\partial l_i}{\partial f_0} &= \frac{\partial h_1}{\partial f_0}\left(\frac{\partial f_1}{\partial h_1}\frac{\partial h_2}{\partial f_1}\frac{\partial f_2}{\partial h_2}\frac{\partial h_3}{\partial f_2}\frac{\partial f_3}{\partial h_3}\frac{\partial l_i}{\partial f_3}\right)
\end{aligned}
\tag{7.12}
$$

Given:

$$
\begin{aligned}
f_0 &= \beta_0 + \omega_0 \cdot x_i \\
h_1 &= \sin[f_0] \\
f_1 &= \beta_1 + \omega_1 \cdot h_1 \\
h_2 &= \exp[f_1] \\
f_2 &= \beta_2 + \omega_2 \cdot h_2 \\
h_3 &= \cos[f_2] \\
f_3 &= \beta_3 + \omega_3 \cdot h_3 \\
l_i &= (y_i - f_3)^2
\end{aligned}
$$

then:

$$
\begin{aligned}
\frac{\partial h_3}{\partial f_2} &= -\text{sin}[f_2] \\
\frac{\partial f_3}{\partial h_3} &= \omega_3 \\
\frac{\partial l_i}{\partial f_3} &= -2(y_i - f_3) \\
\frac{\partial l_i}{\partial f_2} &= -\text{sin}[f_2](\omega_3 \cdot -2(y_i - f_3)) = 2\omega_3\text{sin}[f_2](y_i - f_3)
\end{aligned}
$$

For the second term:

$$
\begin{aligned}
\frac{\partial f_2}{\partial h_2} &= \omega_2 \\
\frac{\partial l_i}{\partial h_2} &= 2\omega_2\omega_3\text{sin}[f_2](y_i - f_3)
\end{aligned}
$$

For the third term:

$$
\begin{aligned}
\frac{\partial h_2}{\partial f_1} &= \text{exp}[f_1] \\
\frac{\partial l_i}{\partial f_1} &= 2\omega_2\omega_3\text{sin}[f_2](y_i - f_3)\text{exp}[f_1]
\end{aligned}
$$

For the fourth term:

$$
\begin{aligned}
\frac{\partial f_1}{\partial h_1} &= \omega_1 \\
\frac{\partial l_i}{\partial h_1} &= 2\omega_1\omega_2\omega_3\text{sin}[f_2](y_i - f_3)\text{exp}[f_1]
\end{aligned}
$$

For the final term:

$$
\begin{aligned}
\frac{\partial h_1}{\partial f_0} &= \text{cos}[f_0] \\
\frac{\partial l_i}{\partial f_0} &= 2\omega_0\omega_1\omega_2\omega_3\text{sin}[f_2](y_i - f_3)\text{exp}[f_1]\text{cos}[f_0]
\end{aligned}
$$

Problem 7.3

What are the sizes of each term in equation 7.19?

$$\frac{\partial l_i}{\partial \mathbf{f}_0} = \frac{\partial \mathbf{h}_1}{\partial \mathbf{f}_0}\frac{\partial \mathbf{f}_1}{\partial \mathbf{h}_1}\left(\frac{\partial \mathbf{h}_2}{\partial \mathbf{f}_1}\frac{\partial \mathbf{f}_2}{\partial \mathbf{h}_2}\frac{\partial \mathbf{h}_3}{\partial \mathbf{f}_2}\frac{\partial \mathbf{f}_3}{\partial \mathbf{h}_3}\frac{\partial l_i}{\partial \mathbf{f}_3}\right) \tag{7.19}$$

- $\frac{\partial l_i}{\partial \mathbf{f}_0}$: $D_1 \times 1$
- $\frac{\partial \mathbf{h}_1}{\partial \mathbf{f}_0}$: $D_1 \times D_1$ (number of hidden units in the first layer)
- $\frac{\partial \mathbf{f}_1}{\partial \mathbf{h}_1}$: $D_1 \times D_2$
- $\frac{\partial \mathbf{h}_2}{\partial \mathbf{f}_1}$:$D_2 \times D_2$ (number of hidden units in the second layer)
- $\frac{\partial \mathbf{f}_2}{\partial \mathbf{h}_2}$:$D_2 \times D_3$
- $\frac{\partial \mathbf{h}_3}{\partial \mathbf{f}_2}$: $D_3 \times D_3$ (number of hidden units in the third layer)
- $\frac{\partial \mathbf{f}_3}{\partial \mathbf{h}_3}$: $D_3 \times D_f$
- $\frac{\partial l_i}{\partial \mathbf{f}_3}$: $D_f \times 1$ where D_f is the dimensionality of the model output $\mathbf{f}_3$

Problem 7.4

Calculate the derivate of $\partial l_i / \partial \mathbf{f}[\mathbf{x}_i, \boldsymbol{\phi}]$ for the least squares loss function:

$$l_i = (y_i - \mathbf{f}[\mathbf{x}_i, \boldsymbol{\phi}])^2$$

$$\partial l_i / \partial \mathbf{f}[\mathbf{x}_i, \boldsymbol{\phi}] = -2(y_i - \mathbf{f}[\mathbf{x}_i, \boldsymbol{\phi}])$$

Problem 7.5

Calculate the derivate of $\partial l_i / \partial \mathbf{f}[\mathbf{x}_i, \boldsymbol{\phi}]$ for the binary classification loss function:

$$l_i = -(1 - y_i)\log(1 - \text{sig}[\mathbf{f}[\mathbf{x}_i, \boldsymbol{\phi}]]) - y_i \log(\text{sig}[\mathbf{f}[\mathbf{x}_i, \boldsymbol{\phi}]])$$

where $\text{sig}[z] = 1/(1 + \exp(-z))$ is the sigmoid function.

The derivative of the sigmoid function is given by:

$$\frac{\partial \text{sig}[z]}{\partial z} = \text{sig}[z](1 - \text{sig}[z])$$

Therefore, differentiating $-y_i \log(\text{sig}[\mathbf{f}[\mathbf{x}_i, \boldsymbol{\phi}]])$:

$$-y_i \frac{1}{\text{sig}[\mathbf{f}[\mathbf{x}_i, \boldsymbol{\phi}]]} \cdot \text{sig}[\mathbf{f}[\mathbf{x}_i, \boldsymbol{\phi}]](1 - \text{sig}[\mathbf{f}[\mathbf{x}_i, \boldsymbol{\phi}]]) = -y_i(1 - \text{sig}[\mathbf{f}[\mathbf{x}_i, \boldsymbol{\phi}]])$$

differentiating $-(1 - y_i)\log(1 - \text{sig}[\mathbf{f}[\mathbf{x}_i, \boldsymbol{\phi}]])$:

$$-(1 - y_i)\frac{-1}{1 - \text{sig}[\mathbf{f}[\mathbf{x}_i, \boldsymbol{\phi}]]} \cdot \text{sig}[\mathbf{f}[\mathbf{x}_i, \boldsymbol{\phi}]](1 - \text{sig}[\mathbf{f}[\mathbf{x}_i, \boldsymbol{\phi}]]) = (1 - y_i)\text{sig}[\mathbf{f}[\mathbf{x}_i, \boldsymbol{\phi}]]$$

Combining the two terms we get:

$$\frac{\partial l_i}{\partial \mathbf{f}[\mathbf{x}_i, \boldsymbol{\phi}]} = (1 - y_i)\text{sig}[\mathbf{f}[\mathbf{x}_i, \boldsymbol{\phi}]] - y_i(1 - \text{sig}[\mathbf{f}[\mathbf{x}_i, \boldsymbol{\phi}]])$$

Which simplifies to:

$$\frac{\partial l_i}{\partial \mathbf{f}[\mathbf{x}_i, \boldsymbol{\phi}]} = \text{sig}[\mathbf{f}[\mathbf{x}_i, \boldsymbol{\phi}]] - y_i$$

Problem 7.6

Show that for $\mathbf{z} = \boldsymbol{\beta} + \boldsymbol{\Omega}\mathbf{h}$:

$$\frac{\partial \mathbf{z}}{\partial \mathbf{h}} = \boldsymbol{\Omega}^T$$

where $\partial\mathbf{z}/\partial\mathbf{h}$ is a matrix containing the terms $\partial z_i/\partial h_j$ in its i-th row and j-th column. To do this, first find an expression for the constituent elements $\partial z_i/\partial h_j$, and then consider the form that the matrix $\partial\mathbf{z}/\partial\mathbf{h}$ must take.

Each z_i is given by:

$$z_i = \beta_i + \sum_j \omega_{ij} h_j$$

The derivative of z_i with respect to h_j is:

$$\frac{\partial z_i}{\partial h_j} = \omega_{ij}$$

This will be at the i-th row and j-th column of the matrix $\partial \mathbf{z}/\partial \mathbf{h}$, which is the transpose of $\mathbf{\Omega}$.

Problem 7.7

Consider the case where we use the logistic sigmoid as an activation function, so $h = \text{sig}[f]$. Compute the derivate $\partial h/\partial f$ for this activation function. What happens to the derivate when the input takes (i) a large positive value and (ii) a large negative value?

The derivative of the sigmoid function is given by:

$$\frac{\partial \text{sig}[f]}{\partial z} = \text{sig}[f](1 - \text{sig}[f])$$

As f approaches a large positive value, the sigmoid function approaches 1, and the derivative approaches 0. As f approaches a large negative value, the sigmoid function approaches 0, and the derivative also approaches 0.

Problem 7.8

Consider using (i) the Heaviside function and (ii) the rectangular function as activation functions:

$$\text{Heaviside}[z] = \begin{cases} 0 & z < 0 \\ 1 & z \geq 0 \end{cases}$$

and

$$\text{rect}[z] = \begin{cases} 0 & z < 0 \\ 1 & 0 \leq z \leq 1 \\ 0 & z > 1 \end{cases}$$

Discuss why these functions are problematic for neural network training with gradient-based optimization methods.

For the Heaviside function, the derivative is zero everywhere except at zero, which means that during backpropagation, the gradient signal used to update weights for inputs that do not exactly equal 0 will be 0, leading to no weight updates. This is known as the vanishing gradient problem. At $z = 0$, the function is discontinuous, and its derivative is undefined. This makes it difficult to apply standard gradient-based optimization methods, as these rely on the ability to compute gradients everywhere in the input space.

For the rectangular function, the rectangular function has a derivative of 0 almost everywhere, except at the discontinuities (at $z = 0$ and $z = 1$), where it is undefined. This leads to the vanishing gradient problem, where the gradient signal is lost, and weights are not updated effectively during training.

Problem 7.9

Consider a loss function $l[\mathbf{f}]$ where $\mathbf{f} = \boldsymbol{\beta} + \boldsymbol{\Omega}\mathbf{h}$. We want to find how the loss l changes when we change $\boldsymbol{\Omega}$, which we'll express with a matrix that contains the derivate $\partial l/\partial \Omega_{ij}$ at the i-th row and j-th column. Find an expression for $\partial f_{ij}/\partial \Omega_{ij}$, and, using the chain rule, show that:

$$\frac{\partial l}{\partial \boldsymbol{\Omega}} = \frac{\partial l}{\partial \mathbf{f}}\mathbf{h}^T$$

For any given component f_{ij}, we have:

$$f_{ij} = \beta_i + \sum_k \Omega_{ik} h_k$$

Therefore, the derivate of f_{ij} with respect to a specific weight Ω_{ij} is:

$$\frac{\partial f_{ij}}{\partial \Omega_{ij}} = h_j$$

Using the chain rule, we have:

$$\frac{\partial l_i}{\partial \Omega_{ij}} = \frac{\partial l}{\partial f_i}\frac{\partial f_i}{\partial \Omega_{ij}} = \frac{\partial l}{\partial f_{ij}} h_j$$

This can be written in vector form as:

$$\frac{\partial l}{\partial \boldsymbol{\Omega}} = \frac{\partial l}{\partial \mathbf{f}}\mathbf{h}^T$$

Problem 7.10

Derive the equations for the backward pass of the backpropagation algorithm for a network that uses leaky ReLU activations, which are defined as:

$$\mathrm{a}[z] = \mathrm{ReLU}[z] = \begin{cases} z & z \geq 0 \\ \alpha \cdot z & z < 0 \end{cases}$$

where α is a small positive constant.

The Leaky ReLU has a gradient of $+1$ when the input is greater than zero and α when it is less. The backpropagation equations (equation 7.23) will hence be the same, except for the update:

$$\frac{\partial l_i}{\partial \mathbf{f}_{k-1}} = \mathbb{I}[\mathbf{f}_{k-1} > 0] \odot \left(\boldsymbol{\Omega}_k^T \frac{\partial l_i}{\partial \mathbf{f}_k}\right) + \mathbb{I}[\mathbf{f}_{k-1} < 0] \odot \alpha \left(\boldsymbol{\Omega}_k^T \frac{\partial l_i}{\partial \mathbf{f}_k}\right)$$

Problem 7.11

Consider training a network with fifty layers, where we only have enough memory to store the pre-activations at every tenth hidden layer during the forward pass. Explain how to compute the derivatives in this situation using gradient checkpointing.

Gradient checkpointing is a technique used to reduce memory usage during the training of deep neural networks by selectively storing only a subset of intermediate states during the forward pass. Then, during the backward pass, the algorithm recomputes the non-stored states as needed by re-executing the forward computations. This trade-off between computational overhead and memory usage is particularly useful when training very deep networks on hardware with limited memory.

The advantage of gradient checkpointing is that it significantly lowers the memory required to store intermediate activations, making it feasible to train deeper networks on limited hardware. However, recomputing activations during the backward pass increases the overall computation time, and it requires modifications to the standard training loop to manage checkpoints and recompute activations as needed.

Problem 7.12

This problem explores computing derivates on general acyclic computational graphs. Consider the function:

$$y = \exp[\exp[x] + \exp[x]^2] + \sin[\exp[x] + \exp[x]^2] \tag{7.42}$$

We can break this down into a series of intermediate computations so that:

$$\begin{aligned}
f_1 &= \exp[x]\\
f_2 &= f_1^2\\
f_3 &= f_1 + f_2\\
f_4 &= \exp[f_3]\\
f_5 &= \sin[f_3]\\
y &= f_4 + f_5
\end{aligned}$$

Compute the derivate $\partial y/\partial x$ by *reverse-mode differentiation* (i.e., backpropagation) using the chain rule. In other words, compute in order:

$$\frac{\partial y}{\partial f_5}, \frac{\partial y}{\partial f_4}, \frac{\partial y}{\partial f_3}, \frac{\partial y}{\partial f_2}, \frac{\partial y}{\partial f_1}, \text{and} \frac{\partial y}{\partial x}$$

using the chain rule in each case to make use of the derivatives already computed.

The derivatives are:

$$\begin{aligned}
\frac{\partial y}{\partial f_5} &= 1\\
\frac{\partial y}{\partial f_4} &= 1\\
\frac{\partial y}{\partial f_3} &= \frac{\partial y}{\partial f_4}\frac{\partial f_4}{\partial f_3} + \frac{\partial y}{\partial f_5}\frac{\partial f_5}{\partial f_3} = \exp[f_3] + \cos[f_3]\\
\frac{\partial y}{\partial f_2} &= \frac{\partial y}{\partial f_3}\frac{\partial f_3}{\partial f_2} = \exp[f_3] + \cos[f_3]\\
\frac{\partial y}{\partial f_1} &= \frac{\partial y}{\partial f_3}\frac{\partial f_3}{\partial f_1} + \frac{\partial y}{\partial f_2}\frac{\partial f_2}{\partial f_1} = \exp[f_3] + \cos[f_3] + 2f_1(\exp[f_3] + \cos[f_3])\\
\frac{\partial y}{\partial x} &= \frac{\partial y}{\partial f_1}\frac{\partial f_1}{\partial x} = \exp[f_3] + \cos[f_3] + 2f_1(\exp[f_3] + \cos[f_3])\exp[f_1]
\end{aligned}$$

Problem 7.13

For the same function in problem 7.42, compute the derivative $\partial y/\partial x$ by *forward-mode differentiation* using the chain rule. In other words, compute in order:

$$\frac{\partial f_1}{\partial x}, \frac{\partial f_2}{\partial x}, \frac{\partial f_3}{\partial x}, \frac{\partial f_4}{\partial x}, \frac{\partial f_5}{\partial x}, \text{and} \frac{\partial y}{\partial x}$$

using the chain rule in each case to make use of the derivatives already computed. Why do we not use forward-mode differentiation when we calculate the parameter gradients for deep networks?

The derivatives are:

$$\begin{aligned}
\frac{\partial f_1}{\partial x} &= \exp[x] \\
\frac{\partial f_2}{\partial x} &= \frac{\partial f_2}{\partial f_1}\frac{\partial f_1}{\partial x} = 2f_1\exp[x] \\
\frac{\partial f_3}{\partial x} &= \frac{\partial f_3}{\partial f_1}\frac{\partial f_1}{\partial x} + \frac{\partial f_3}{\partial f_2}\frac{\partial f_2}{\partial x} = \exp[x](1+2f_1) \\
\frac{\partial f_4}{\partial x} &= \frac{\partial f_4}{\partial f_3}\frac{\partial f_3}{\partial x} = \exp[f_3]\exp[x](1+2f_1) \\
\frac{\partial f_5}{\partial x} &= \frac{\partial f_5}{\partial f_3}\frac{\partial f_3}{\partial x} = \cos[f_3]\exp[x](1+2f_1) \\
\frac{\partial y}{\partial x} &= \frac{\partial y}{\partial f_4}\frac{\partial f_4}{\partial x} + \frac{\partial y}{\partial f_5}\frac{\partial f_5}{\partial x} = \exp[f_3]\exp[x](1+2f_1) + \cos[f_3]\exp[x](1+2f_1)
\end{aligned}$$

Forward-mode differentiation would require a separate forward pass for each parameter to compute its gradient, leading to a significant computational and memory burden in deep networks. Reverse-mode differentiation, on the other hand, allows for the efficient computation of all parameter gradients in a single backward pass following a forward pass, leveraging the chain rule in a manner that's particularly well-suited to the architectures of deep networks.

Problem 7.14

Consider a random variable a with variance σ^2 and a symmetrical distribution around the mean 0. Prove that if we pass this variable through the ReLU function, then the second moment of the transformed variable is $\mathbb{E}[b^2] = \sigma^2/2$.

If $b = \text{ReLU}[a] = \max(0, a)$, then the second moment of b is given by:

$$\mathbb{E}[b^2] = \mathbb{E}[(\max(0,a))^2] = \mathbb{E}[a^2 \cdot \mathbb{I}(a > 0)]$$

Since the distribution of a is symmetrical around 0 and the variance is σ^2, the expected value of a^2, which is the second moment of a, equals the variance (because the mean is 0). Considering that the probability of a being positive is 0.5, we have:

$$\mathbb{E}[b^2] = \mathbb{E}[a^2 \cdot \mathbb{I}(a > 0)] = \frac{1}{2}\mathbb{E}[a^2] = \frac{1}{2}\sigma^2$$

as required.

Problem 7.15

What would you expect to happen if we initialized all of the weights and biases in the network to zero?

When weights and biases are initialized to zero, the activation functions in the network (especially if they are symmetric around the origin, like the sigmoid or tanh functions) will output zeros (or constant values for biases) in the forward pass. During the backward pass, the derivative of the loss with respect to the weights will also be zero, given the derivative of the activation function and the gradient of the loss. This results in no updates being made to the weights and biases during training, effectively halting the learning process. Additionally, starting with zero weights means the gradients start small and diminish even further as they are propagated back through the network, making learning exceedingly slow or non-existent for deep networks (vanishing gradients).

Problem 7.16

Implement the code in figure 7.8 in PyTorch and plot the training loss as a function of the number of epochs.

Coding questions not included.

Problem 7.17

Change the code in figure 7.8 to tackle a binary classification problem. You will need to (i) change the targets y so they are binary, (ii) change the network to predict numbers between zero and one (iii) change the loss function appropriately.

Coding questions not included.