



Data Analysis Block Course: Statistics and Machine learning

Neural Networks by Hand

Prof. Dr. Mikhail Mikhasenko

Assistants: Hendric Jonas, Christoph Tönnis



29 September 2026

[Indico schedule](#)

[RUB Moodle](#)

Learning goals

After this exercise, you should be able to

- evaluate a simple neural network by hand,
- explain the role of weights, biases, and activation functions,
- perform a complete forward pass through a small neural network,
- calculate the mean squared error for a small training dataset,
- use backpropagation to compute gradients of the loss with respect to all trainable parameters,
- explain how the ReLU activation affects gradient propagation,
- perform one gradient-descent update,
- verify that a gradient step can reduce the training loss,
- distinguish between fitting the training points and learning the underlying function,
- explain why a ReLU network produces a piecewise-linear function.

The calculations in this sheet are designed to be done by hand.

Part I - A small neural network

We want to approximate the function

$$f(x) = -x^2 + 4x \quad (1)$$

We use only the four training inputs

$$x_i = 0, 1, 2, 3$$

Exercise 1.1 - Generate the training data

Evaluate the function at the four training inputs and complete the table.

x_i	$y_i = f(x_i)$
0	
1	
2	
3	

Plot the four training points and sketch the true function $f(x) = -x^2 + 4x$ in the range $-1 \leq x \leq 5$.

Exercise 1.2 - Our neural network architecture

We use the architecture

$$1 \rightarrow 2 \rightarrow 1$$

There is

- one input
- one hidden layer with two neurons
- one output neuron

The hidden layer uses the ReLU activation

$$\text{ReLU}(z) = \max(0, z) \quad (2)$$

The hidden neurons and linear output are defined by

$$\begin{aligned} z_{1,i} &= w_1 x_i + b_1 & h_{1,i} &= \text{ReLU}(z_{1,i}) \\ z_{2,i} &= w_2 x_i + b_2 & h_{2,i} &= \text{ReLU}(z_{2,i}) \\ \hat{y}_i &= v_1 h_{1,i} + v_2 h_{2,i} + c \end{aligned} \quad (3)$$

Sketch the following

- the ReLU function for input values $z \in [-5, 5]$
- the neural network

For this, count the trainable parameters first. Your drawing should contain

- the input x_i
- the two hidden neurons
- the output $\hat{y}_i$
- all weights
- all biases
- the ReLU activation in the hidden layer

Exercise 1.3 - A complete forward pass

To build the neural network, the parameters need to be fit. We start with a list of initial guesses for which the output of the neural network is calculated. This will be compared to the training data using the mean squared error. Training repeats forward passes, backpropagation and parameter updates, with validation performance guiding the stopping point. Here we perform one training step by hand.

For the first training step, use the following initial parameters:

$$\begin{aligned} w_1 &= 1 & b_1 &= -0.5 \\ w_2 &= -1 & b_2 &= 2.5 \\ v_1 &= 1 & v_2 &= 1 & c &= 0 \end{aligned}$$

This exercise is divided into multiple steps:

Step 1 - Hidden-layer activations

For each of the four training inputs, calculate

$$\begin{aligned} z_{1,i} &= w_1 x_i + b_1 \\ h_{1,i} &= \text{ReLU}(z_{1,i}) \end{aligned} \tag{4}$$

and analogously for $z_{2,i}$ and $h_{2,i}$. Complete the table below:

x_i	y_i	$z_{1,i}$	$h_{1,i}$	$z_{2,i}$	$h_{2,i}$
0	0				
1	3				
2	4				
3	3				

For each hidden neuron, identify for which input values the ReLU is active.

Step 2 - Output predictions

Using

$$\hat{y} = v_1 h_1 + v_2 h_2 + c$$

calculate the network prediction for every training point:

x	y	$\hat{y}$	residual $\hat{y} - y$
0	0		
1	3		
2	4		
3	3		

Plot the four predictions together with the four training points.

Step 3 - Calculate the loss

We use the mean squared error

$$L = \frac{1}{N} \sum_{i=1}^N (\hat{y}_i - y_i)^2 \tag{5}$$

with $N = 4$.

Calculate the MSE for the current network. This is the loss before training.

Exercise 1.4 - Backpropagation by hand

We now calculate how the loss changes with respect to every trainable parameter.

The important idea is:

Start at the loss and repeatedly apply the chain rule while moving backwards through the network.

We perform this calculation only once by hand. A neural-network library repeats exactly the same type of calculation automatically.

Step 1 - Start at the loss

For each sample, define the squared error L_i and its contribution to the mean loss. Differentiating the mean gives

$$\begin{aligned} L_i &= (\hat{y}_i - y_i)^2 & L &= \frac{1}{4} \sum_i L_i \\ \delta_i &= \frac{\partial L}{\partial \hat{y}_i} = \frac{2}{4} (\hat{y}_i - y_i) \end{aligned} \tag{6}$$

Calculate δ_i for every training point.

x_i	y_i	$\hat{y}_i$	$\hat{y}_i - y_i$	$\delta = \partial L / \partial \hat{y}$
0	0			
1	3			
2	4			
3	3			

Step 2 - Gradients of the output layer

The output $\hat{y}_i = v_1 h_{1,i} + v_2 h_{2,i} + c$ has derivatives $h_{1,i}$, $h_{2,i}$ and 1 with respect to v_1 , v_2 and c . Apply the chain rule to compute the three parameter gradients:

$$\begin{aligned} \frac{\partial L}{\partial v_1} &= \sum_i \frac{\partial L}{\partial \hat{y}_i} \frac{\partial \hat{y}_i}{\partial v_1} = \sum_i \delta_i h_{1,i} \\ \frac{\partial L}{\partial v_2} &= \sum_i \delta_i h_{2,i} \\ \frac{\partial L}{\partial c} &= \sum_i \delta_i \end{aligned} \tag{7}$$

Record your results:

parameter	gradient
v_1	
v_2	
c	

Step 3 - Backpropagate through ReLU

The derivative of ReLU is

$$\text{ReLU}'(z) = \begin{cases} 0 & z < 0 \\ 1 & z > 0 \end{cases}$$

For the values encountered in this exercise, none of the z values are exactly zero.

Using your values from the forward pass, complete the table:

x_i	$z_{1,i}$	$\text{ReLU}'(z_{1,i})$	$z_{2,i}$	$\text{ReLU}'(z_{2,i})$
0				
1				
2				
3				

For the first hidden neuron, apply the chain rule through the output and activation. Since $\partial \hat{y}_i / \partial h_{1,i} = v_1$, the derivative simplifies as follows. The second neuron is analogous:

$$\begin{aligned}
\frac{\partial L}{\partial z_{1,i}} &= \frac{\partial L}{\partial \hat{y}_i} \frac{\partial \hat{y}_i}{\partial h_{1,i}} \frac{\partial h_{1,i}}{\partial z_{1,i}} \\
&= \delta_i v_1 \text{ReLU}'(z_{1,i}) \\
\frac{\partial L}{\partial z_{2,i}} &= \delta_i v_2 \text{ReLU}'(z_{2,i})
\end{aligned} \tag{8}$$

Calculate these values for both neurons at all four training points.

Complete:

x_i	$\partial L / \partial z_{1,i}$	$\partial L / \partial z_{2,i}$
0		
1		
2		
3		

What happens to the gradient when a ReLU neuron is inactive?

Step 4 - Gradients of the first layer

For the first neuron,

$$z_{1,i} = w_1 x_i + b_1$$

Therefore,

$$\begin{aligned}
\frac{\partial z_{1,i}}{\partial w_1} &= x_i \\
\frac{\partial z_{1,i}}{\partial b_1} &= 1
\end{aligned}$$

Calculate the gradients $\partial L / \partial w_1$ and $\partial L / \partial b_1$.

Repeat the calculation for the second hidden neuron and collect all seven gradients:

parameter	gradient
w_1	
b_1	
w_2	
b_2	
v_1	
v_2	
c	

Exercise 1.5 - One gradient-descent step

We now update every parameter using gradient descent. For a parameter θ

$$\theta_{\text{new}} = \theta_{\text{old}} - \eta \frac{\partial L}{\partial \theta} \quad (9)$$

Use the learning rate

$$\eta = 0.1$$

Step 1 - Update the parameters

Complete the table.

parameter	old value	gradient	new value
w_1	1.0		
b_1	-0.5		
w_2	-1.0		
b_2	2.5		
v_1	1.0		
v_2	1.0		
c	0.0		

Step 2 - Did training improve the model?

Using the updated parameters, perform one more forward pass (see Exercise 1.3). (You do not need to perform another backward pass)

Calculate the new predictions and the new MSE.

x	y	new prediction $\hat{y}_{\text{new}}$
0	0	
1	3	
2	4	
3	3	

Compare L_{old} and L_{new} .

Did this gradient-descent step reduce the loss? Does every single training point necessarily improve after one gradient step?

Explain why the optimizer can still reduce the total loss even if one individual prediction becomes worse.

Part II - Investigating a fitted neural network

Repeating forward pass, backpropagation, and parameter updates many times is exactly what neural-network training does.

Instead of repeating these calculations by hand, we now inspect a network whose parameters are already chosen.

Consider

$$\begin{aligned}h_1(x) &= \text{ReLU}(x) \\h_2(x) &= \text{ReLU}(x - 1.5) \\ \hat{f}(x) &= 3h_1(x) - 4h_2(x)\end{aligned}$$

This is again a network with architecture

$$1 \rightarrow 2 \rightarrow 1$$

Exercise 2.1 - Identify the network parameters

Write the network in the standard form

$$\begin{aligned}z_1 &= w_1x + b_1 & h_1 &= \text{ReLU}(z_1) \\z_2 &= w_2x + b_2 & h_2 &= \text{ReLU}(z_2) \\ \hat{y} &= v_1h_1 + v_2h_2 + c\end{aligned}$$

Determine the parameters

$$w_1, b_1, w_2, b_2, v_1, v_2, c$$

Exercise 2.2 - Check the training points

Evaluate the fitted network at

$$x = 0, 1, 2, 3$$

Complete the table.

x	true $f(x)$	network $\hat{f}(x)$
0	0	
1	3	
2	4	
3	3	

Calculate the training MSE.

What is special about this result?

Exercise 2.3 - Has the network learned the true function?

Now evaluate both the true function and the neural network at new input values:

$$x = 0.5, 1.5, 2.5, 4$$

Complete the table.

x	true $f(x)$	network $\hat{f}(x)$	difference
0.5			
1.5			
2.5			
4.0			

Discuss:

1. The network has zero training error. Has it learned the exact function?
2. Why is performance on unseen inputs important?

Exercise 2.4 - What function has the ReLU network learned?

The fitted network is

$$\hat{f}(x) = 3 \text{ReLU}(x) - 4 \text{ReLU}(x - 1.5)$$

Because ReLU changes its behavior at zero, the expression changes at $x = 0$ and $x = 1.5$.

Consider the three regions separately and determine $\hat{f}(x)$. Write the final result as

$$\hat{f}(x) = \begin{cases} \dots & x < 0 \\ \dots & 0 \leq x < 1.5 \\ \dots & x \geq 1.5 \end{cases}$$

Plot the function. What is the most important visual difference between the quadratic target function and the ReLU network?

Part III - A deeper neural network

So far, we have considered a neural network with one hidden layer. We now investigate a deeper network with the architecture

$$1 \rightarrow 3 \rightarrow 3 \rightarrow 1$$

There is

- one input
- a first hidden layer with three neurons
- a second hidden layer with three neurons
- one output neuron

Both hidden layers use the ReLU activation function.

In this part, the parameters are already given. You do not need to perform backpropagation for this larger network. The goal is to understand how information is propagated through multiple hidden layers and how additional neurons can change the shape of the learned function.

Exercise 3.1 - Architecture and number of parameters

For a fully connected neural network, every neuron in one layer is connected to every neuron in the next layer.

1. Draw the architecture

$$1 \rightarrow 3 \rightarrow 3 \rightarrow 1$$

1. Count the number of weights and biases between the input and the first hidden layer.
2. Count the number of weights and biases between the first and second hidden layers.
3. Count the number of weights and biases between the second hidden layer and the output.
4. Determine the total number of trainable parameters.

Compare this number with the seven trainable parameters of the network from Parts I and II.

Exercise 3.2 - First hidden layer

The first hidden layer is defined by

$$\begin{aligned} z_{1,i}^{(1)} &= x_i & h_{1,i}^{(1)} &= \text{ReLU}(z_{1,i}^{(1)}) \\ z_{2,i}^{(1)} &= x_i - 1 & h_{2,i}^{(1)} &= \text{ReLU}(z_{2,i}^{(1)}) \\ z_{3,i}^{(1)} &= x_i - 2 & h_{3,i}^{(1)} &= \text{ReLU}(z_{3,i}^{(1)}) \end{aligned}$$

Calculate the activations for the four training inputs.

x_i	$z_{1,i}^{(1)}$	$h_{1,i}^{(1)}$	$z_{2,i}^{(1)}$	$h_{2,i}^{(1)}$	$z_{3,i}^{(1)}$	$h_{3,i}^{(1)}$
0						
1						
2						
3						

At which input value does each of the three neurons become active?

How does this differ from the two-neuron hidden layer considered previously?

Exercise 3.3 - Second hidden layer

The second hidden layer receives the three outputs of the first hidden layer as its inputs.

It is defined by

$$\begin{aligned} z_{1,i}^{(2)} &= h_{1,i}^{(1)} & h_{1,i}^{(2)} &= \text{ReLU}(z_{1,i}^{(2)}) \\ z_{2,i}^{(2)} &= h_{1,i}^{(1)} - h_{2,i}^{(1)} & h_{2,i}^{(2)} &= \text{ReLU}(z_{2,i}^{(2)}) \\ z_{3,i}^{(2)} &= h_{2,i}^{(1)} - h_{3,i}^{(1)} & h_{3,i}^{(2)} &= \text{ReLU}(z_{3,i}^{(2)}) \end{aligned}$$

Use the results from Exercise 3.2 and complete the table.

x	$z_{1,i}^{(2)}$	$h_{1,i}^{(2)}$	$z_{2,i}^{(2)}$	$h_{2,i}^{(2)}$	$z_{3,i}^{(2)}$	$h_{3,i}^{(2)}$
0						
1						
2						
3						

Notice that the second hidden layer no longer acts directly on x . It transforms features that were already created by the first hidden layer.

Exercise 3.4 - Output of the deep network

The output layer is

$$\hat{f}_{\text{deep}}(x) = -h_1^{(2)} + 4h_2^{(2)} + 2h_3^{(2)}$$

Calculate the prediction for every training point.

x_i	true $f(x_i)$	deep network $\hat{f}_{\text{deep}}(x_i)$
0	0	
1	3	
2	4	
3	3	

Calculate the training MSE.

Compare the result with the fitted $1 \rightarrow 2 \rightarrow 1$ network from Part II.

- Do both networks achieve the same training error?
- Does this imply that both networks represent the same function?

Exercise 3.5 - Test the network between the training points

The two networks both reproduce the four training points exactly. We now compare what they predict between the training points.

Evaluate

$$f(x) = -x^2 + 4x \tag{10}$$

at

$$x = 0.5, 1.25, 1.75, 2.5$$

and use your result from the shallow network of Part II. For the deep network, perform the forward pass through both hidden layers and complete the table.

x_i	true $f(x_i)$	shallow network	deep network
0.5			
1.25			
1.75			
2.5			

Which network is closer to the true function at these points? Why can two networks with zero training error behave differently between the training observations?

Exercise 3.6 - What function has the deeper network learned?

Determine the output of the deeper network in the following regions:

1. $x < 0$
2. $0 \leq x < 1$
3. $1 \leq x < 2$
4. $x \geq 2$

Write the result as a piecewise function

$$\hat{f}_{\text{deep}}(x) = \begin{cases} \dots & x < 0 \\ \dots & 0 \leq x < 1 \\ \dots & 1 \leq x < 2 \\ \dots & x \geq 2 \end{cases}$$

Plot in the same coordinate system

- the true function $f(x) = -x^2 + 4x$
- the four training points
- the fitted $1 \rightarrow 2 \rightarrow 1$ network from Part II
- the fitted $1 \rightarrow 3 \rightarrow 3 \rightarrow 1$ network

What has changed when moving to the larger network?

In particular, compare the number and positions of the changes in slope.

Short conceptual questions

1. Why is the output of this ReLU network piecewise linear?
2. What determines the positions at which the slope changes?
3. What role do the output weights v_1 and v_2 play?
4. Why can a network with only two hidden neurons already represent a function that is more complicated than a single straight line?
5. What would additional ReLU neurons allow the network to do?
6. Why does zero training error not guarantee good generalization?