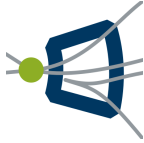


Data Analysis Block Course: Statistics and Machine learning



Neural Networks by Hand - Solutions

Prof. Dr. Mikhail Mikhasenko

Assistants: Hendric Jonas, Christoph Tönnis



29 September 2026

[Indico schedule](#)

[RUB Moodle](#)

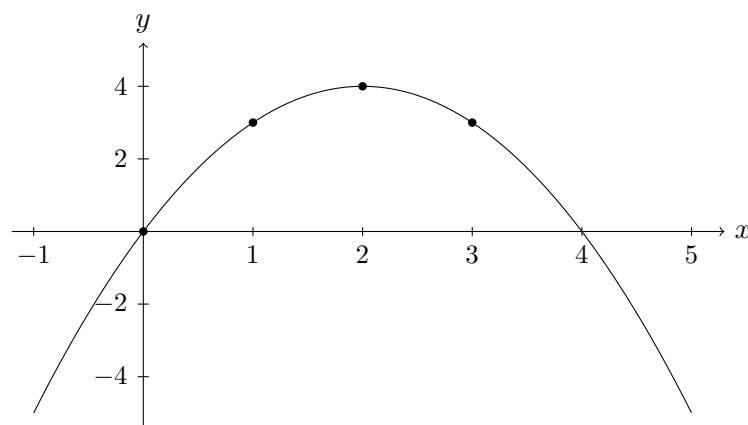
Part I - A small neural network

Solution 1.1 - Generate the training data

For $f(x) = -x^2 + 4x = 4 - (x - 2)^2$, the training data are:

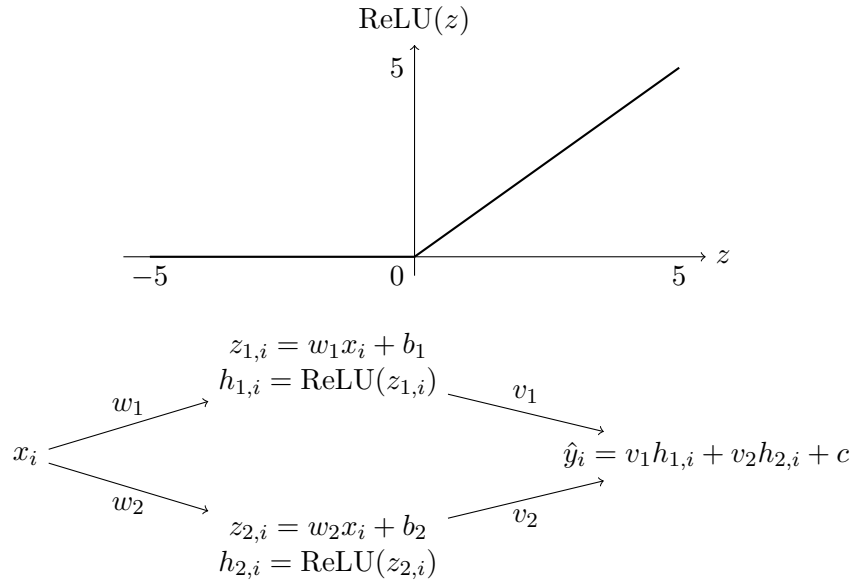
x_i	y_i
0	0
1	3
2	4
3	3

The parabola has vertex $(2, 4)$ and zeros at 0 and 4. At both ends of the requested interval, $f(-1) = f(5) = -5$.



Solution 1.2 - Our neural network architecture

There are **seven trainable parameters**: four weights w_1, w_2, v_1, v_2 and three biases b_1, b_2, c . The hidden neurons use ReLU; the output is linear. ReLU is zero for $z \leq 0$ and equals z for $z \geq 0$.



Solution 1.3 - A complete forward pass

With $w_1 = 1$, $b_1 = -0.5$, $w_2 = -1$, $b_2 = 2.5$, $v_1 = v_2 = 1$ and $c = 0$, the hidden-layer values are:

x_i	y_i	$z_{1,i}$	$h_{1,i}$	$z_{2,i}$	$h_{2,i}$
0	0	-0.5	0	2.5	2.5
1	3	0.5	0.5	1.5	1.5
2	4	1.5	1.5	0.5	0.5
3	3	2.5	2.5	-0.5	0

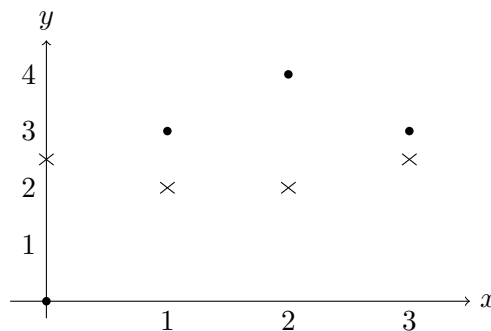
Neuron 1 is active for $x > 0.5$, including training inputs 1, 2 and 3. Neuron 2 is active for $x < 2.5$, including inputs 0, 1 and 2. At each threshold, the corresponding activation is zero.

The output is $\hat{y}_i = h_{1,i} + h_{2,i}$:

x_i	y_i	$\hat{y}_i$	Residual $\hat{y}_i - y_i$
0	0	2.5	2.5
1	3	2	-1
2	4	2	-2
3	3	2.5	-0.5

$$L = \frac{2.5^2 + (-1)^2 + (-2)^2 + (-0.5)^2}{4} = 2.875$$

Dots show the targets; crosses show the initial predictions.



Solution 1.4 - Backpropagation by hand

1. Start at the loss. Since the loss is the mean over four observations, $\delta_i = \frac{1}{2}(\hat{y}_i - y_i)$:

x_i	y_i	$\hat{y}_i$	$\hat{y}_i - y_i$	δ_i
0	0	2.5	2.5	1.25
1	3	2	-1	-0.5
2	4	2	-2	-1
3	3	2.5	-0.5	-0.25

2. Output-layer gradients. Apply the chain rule and sum each parameter's contributions across the observations:

$$\frac{\partial L}{\partial v_1} = \sum_i \delta_i h_{1,i} = 1.25(0) - 0.5(0.5) - 1(1.5) - 0.25(2.5) = -2.375$$

$$\frac{\partial L}{\partial v_2} = \sum_i \delta_i h_{2,i} = 1.25(2.5) - 0.5(1.5) - 1(0.5) - 0.25(0) = 1.875$$

$$\frac{\partial L}{\partial c} = \sum_i \delta_i = 1.25 - 0.5 - 1 - 0.25 = -0.5$$

3. Backpropagate through ReLU. An inactive ReLU has derivative zero, so this observation sends no gradient through that neuron. No pre-activation is zero in this initial forward pass.

x_i	$z_{1,i}$	$\text{ReLU}'(z_{1,i})$	$z_{2,i}$	$\text{ReLU}'(z_{2,i})$
0	-0.5	0	2.5	1
1	0.5	1	1.5	1
2	1.5	1	0.5	1
3	2.5	1	-0.5	0

Using $\partial L / \partial z_{j,i} = \delta_i v_j \text{ReLU}'(z_{j,i})$ with $v_1 = v_2 = 1$:

x_i	$\partial L / \partial z_{1,i}$	$\partial L / \partial z_{2,i}$
0	0	1.25
1	-0.5	-0.5
2	-1	-1
3	-0.25	0

4. First-layer gradients. Each weight receives the pre-activation gradient multiplied by x_i ; each bias receives the pre-activation gradient itself:

$$\frac{\partial L}{\partial w_1} = 0(0) - 0.5(1) - 1(2) - 0.25(3) = -3.25$$

$$\frac{\partial L}{\partial b_1} = 0 - 0.5 - 1 - 0.25 = -1.75$$

$$\frac{\partial L}{\partial w_2} = 1.25(0) - 0.5(1) - 1(2) + 0(3) = -2.5$$

$$\frac{\partial L}{\partial b_2} = 1.25 - 0.5 - 1 + 0 = -0.25$$

The table in Solution 1.5 collects all seven gradients and applies the update.

Solution 1.5 - One gradient-descent step

Update all parameters simultaneously using $\theta_{\text{new}} = \theta - 0.1 \partial L / \partial \theta$. All gradients refer to the old parameters.

Parameter	Old value	Gradient	New value
w_1	1	-3.25	1.325
b_1	-0.5	-1.75	-0.325
w_2	-1	-2.5	-0.750
b_2	2.5	-0.25	2.525
v_1	1	-2.375	1.2375
v_2	1	1.875	0.8125
c	0	-0.5	0.050

The new forward pass gives:

x_i	y_i	$h_{1,i}$	$h_{2,i}$	$\hat{y}_{i,\text{new}}$
0	0	0	2.525	2.1015625
1	3	1	1.775	2.7296875
2	4	2.325	1.025	3.7600000
3	3	3.65	0.275	4.7903125

Using unrounded predictions:

$$\begin{aligned} L_{\text{new}} &= \frac{2.1015625^2 + (-0.2703125)^2 + (-0.24)^2 + 1.7903125^2}{4} \\ &= 1.9381131591796875 < 2.875 = L_{\text{old}} \end{aligned}$$

The total loss falls, although the error at $x = 3$ grows from -0.5 to 1.7903125 . The reduction in the other squared errors outweighs this increase. Gradient descent optimizes the combined loss; it need not improve every observation in one step.

Part II - Investigating a fitted neural network

Solution 2.1 - Identify the network parameters

The supplied network $\hat{f}(x) = 3 \text{ReLU}(x) - 4 \text{ReLU}(x - 1.5)$ has parameters

$$\begin{aligned} w_1 &= 1 & b_1 &= 0 \\ w_2 &= 1 & b_2 &= -1.5 \\ v_1 &= 3 & v_2 &= -4 & c &= 0 \end{aligned}$$

Solution 2.2 - Check the training points

x_i	True $f(x_i)$	Network $\hat{f}(x_i)$
0	0	0
1	3	3
2	4	4
3	3	3

All four predictions are exact, so the training MSE is zero. This establishes agreement at these four inputs only.

Solution 2.3 - Has the network learned the true function?

x	True $f(x)$	Network $\hat{f}(x)$	Difference $\hat{f} - f$
0.5	1.75	1.50	-0.25
1.5	3.75	4.50	0.75
2.5	3.75	3.50	-0.25
4.0	0.00	2.00	2.00

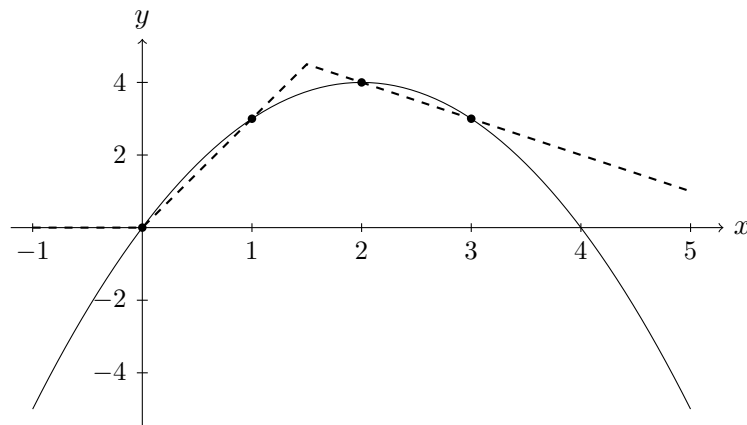
1. The network has not learned the exact quadratic function, despite having zero training error.
2. Unseen inputs test whether the learned relationship generalizes beyond the observations used to choose the parameters. Finitely many training values do not determine the entire function.

Solution 2.4 - What function has the ReLU network learned?

For $x < 0$, both activations vanish. For $0 \leq x < 1.5$, only the first contributes. For $x \geq 1.5$, substitution gives $3x - 4(x - 1.5) = 6 - x$:

$$\hat{f}(x) = \begin{cases} 0 & x < 0 \\ 3x & 0 \leq x < 1.5 \\ 6 - x & x \geq 1.5 \end{cases}$$

The target is a smooth quadratic curve; the network is continuous and piecewise linear, with changes in slope at 0 and 1.5. The solid curve below is the target; the dashed line is the network.

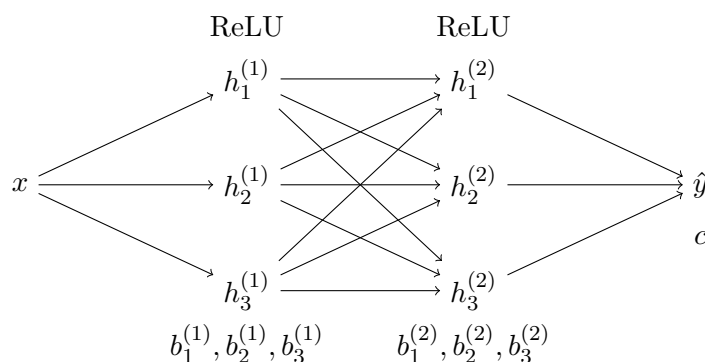


Part III - A deeper neural network

Solution 3.1 - Architecture and number of parameters

Connection	Weights	Biases	Total
Input to hidden layer 1	$1 \cdot 3 = 3$	3	6
Hidden layer 1 to layer 2	$3 \cdot 3 = 9$	3	12
Hidden layer 2 to output	$3 \cdot 1 = 3$	1	4
Total	15	7	22

The fully connected $1 \rightarrow 3 \rightarrow 3 \rightarrow 1$ architecture has 22 trainable parameters, compared with seven in the shallow network. Zero-valued weights in the supplied example still count as parameters of the architecture. Every hidden neuron and the output have a bias.



Solution 3.2 - First hidden layer

The first-layer activations are $\text{ReLU}(x)$, $\text{ReLU}(x - 1)$ and $\text{ReLU}(x - 2)$:

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	0	0	-1	0	-2	0
1	1	1	0	0	-1	0
2	2	2	1	1	0	0
3	3	3	2	2	1	1

The active regions are $x > 0$, $x > 1$ and $x > 2$. Among the training inputs, the active sets are $\{1, 2, 3\}$, $\{2, 3\}$ and $\{3\}$. At a threshold the corresponding activation is still zero.

There are now three thresholds instead of two. The fitted shallow network has thresholds 0 and 1.5; the initial network in Part I has thresholds 0.5 and 2.5, with its second neuron active to the left of 2.5.

Solution 3.3 - Second hidden layer

The second layer acts on the features from the first layer:

$$\begin{aligned}
 h_1^{(2)} &= \text{ReLU}(h_1^{(1)}) \\
 h_2^{(2)} &= \text{ReLU}(h_1^{(1)} - h_2^{(1)}) \\
 h_3^{(2)} &= \text{ReLU}(h_2^{(1)} - h_3^{(1)})
 \end{aligned}$$

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

These neurons combine already transformed features rather than receiving the original input directly.

Solution 3.4 - Output of the deep network

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

x_i	True $f(x_i)$	Output calculation	Prediction
0	0	$-0 + 4(0) + 2(0)$	0
1	3	$-1 + 4(1) + 2(0)$	3
2	4	$-2 + 4(1) + 2(1)$	4
3	3	$-3 + 4(1) + 2(1)$	3

The training MSE is zero for both networks. Agreement on the four training observations does not imply that they represent the same function elsewhere.

Solution 3.5 - Test the network between the training points

The first hidden layer at the new inputs is:

x	$z_1^{(1)}$	$h_1^{(1)}$	$z_2^{(1)}$	$h_2^{(1)}$	$z_3^{(1)}$	$h_3^{(1)}$
0.5	0.5	0.5	-0.5	0	-1.5	0
1.25	1.25	1.25	0.25	0.25	-0.75	0
1.75	1.75	1.75	0.75	0.75	-0.25	0
2.5	2.5	2.5	1.5	1.5	0.5	0.5

All second-layer pre-activations are nonnegative, so each equals its activation:

x	$z_1^{(2)} = h_1^{(2)}$	$z_2^{(2)} = h_2^{(2)}$	$z_3^{(2)} = h_3^{(2)}$
0.5	0.5	0.5	0
1.25	1.25	1	0.25
1.75	1.75	1	0.75
2.5	2.5	1	1

The resulting comparison is:

x	True $f(x)$	Shallow network	Deep network
0.5	1.7500	1.5000	1.5000
1.25	3.4375	3.7500	3.2500
1.75	3.9375	4.2500	3.7500
2.5	3.7500	3.5000	3.5000

At 0.5 and 2.5 the errors are equal. At 1.25 and 1.75, the deep network's absolute error is 0.1875, compared with 0.3125 for the shallow network. The models interpolate differently despite their identical training error.

Solution 3.6 - What function has the deeper network learned?

Evaluate the features region by region:

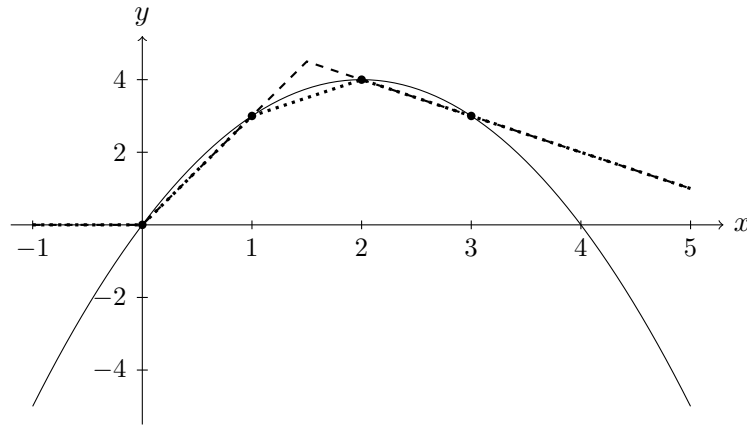
Region	$(h_1^{(1)}, h_2^{(1)}, h_3^{(1)})$	$(h_1^{(2)}, h_2^{(2)}, h_3^{(2)})$
$x < 0$	$(0, 0, 0)$	$(0, 0, 0)$
$0 \leq x < 1$	$(x, 0, 0)$	$(x, x, 0)$
$1 \leq x < 2$	$(x, x - 1, 0)$	$(x, 1, x - 1)$
$x \geq 2$	$(x, x - 1, x - 2)$	$(x, 1, 1)$

Substitution in $-h_1^{(2)} + 4h_2^{(2)} + 2h_3^{(2)}$ gives

$$\hat{f}_{\text{deep}}(x) = \begin{cases} 0 & x < 0 \\ -x + 4x = 3x & 0 \leq x < 1 \\ -x + 4 + 2(x - 1) = x + 2 & 1 \leq x < 2 \\ -x + 4 + 2 = 6 - x & x \geq 2 \end{cases}$$

The shallow network has slopes 0, 3, -1 and changes slope at 0 and 1.5. The deep network has slopes 0, 3, 1, -1 and changes slope at 0, 1 and 2. It adds a linear region and changes the interpolation. More capacity does not by itself guarantee better generalization.

The solid curve is the target, the dashed line the shallow network, and the dotted line the deep network. Dots mark the training observations.



Solutions - Conceptual questions

1. ReLU is piecewise linear. Composing it with affine transformations preserves a piecewise-affine form, conventionally called piecewise linear here.
2. Potential slope changes occur where ReLU pre-activations cross zero. In one hidden layer these positions satisfy $w x + b = 0$, hence $x = -b/w$ for $w \neq 0$. Weights and biases determine the boundaries. In deeper layers the boundaries depend on previously constructed features. A zero output weight or cancellation can hide a hidden neuron's slope change in the final output.

3. Output weights scale the hidden activations. Their magnitudes and signs determine how these contributions affect the value and slope of the prediction.
4. Two ReLU neurons can introduce two distinct thresholds and combine several linear regions, giving a function more flexible than a single straight line.
5. Additional neurons can add activation boundaries and learned features. This permits a finer approximation but also increases capacity; better performance on unseen data is not guaranteed.
6. Zero training error means agreement only at the observed inputs. Different functions can agree there and disagree elsewhere, so generalization must be assessed separately.

Key concepts

Concept	Meaning
Weight / bias	Multiplicative connection / additive offset
Pre-activation	Affine input to an activation function
ReLU	$\max(0, z)$
Forward pass	Compute activations and predictions
MSE	Average squared prediction error
Backpropagation	Apply the chain rule backwards to compute gradients
Gradient descent	Update parameters against the loss gradient
Piecewise-linear network	Affine behavior within each activation region
Generalization	Prediction quality on unseen inputs