

Homework 5: Neural Networks and Backpropagation

Due: May 1st, 2026 at 11:59pm ET

Instructions: Your answers to the questions below, including plots and mathematical work, should be submitted as a single PDF file. It's preferred that you write your answers using software that typesets mathematics (e.g. L^AT_EX or MathJax in iPython), though if you need to you may scan handwritten work. For the coding problems, the text **Submission:** indicates all you need to submit in your PDF submission.

Note: Starter code for Problem 3 can be found in `backprop.py`.

Problem 1: The XOR Problem (20 points)

The XOR function is a boolean function which takes in two binary inputs and returns true if the inputs are different, and false if they are the same. In 1969, the book *Perceptrons* by Marvin Minsky and Seymour Papert showed that certain classes of models were incapable of representing the XOR function. This result was widely mis-cited and was used to argue that neural networks were incapable of learning simple functions.

In this problem you will reason about whether simple neural network architectures can represent the XOR function. Consider the four input points $x = (x_1, x_2) \in \{0, 1\}^2$ with labels $y \in \{-1, +1\}$ given by XOR:

$$y = \begin{cases} +1 & \text{if } (x_1, x_2) \in \{(0, 1), (1, 0)\} \\ -1 & \text{if } (x_1, x_2) \in \{(0, 0), (1, 1)\}. \end{cases}$$

Throughout, we will say that a model achieves *zero loss* if it classifies all four points correctly (i.e., zero classification error on this dataset).

Problem 1(a) (5 points): Linear classifier

Consider a simple linear classifier with weights w_1, w_2 and bias b , using the classification rule

$$\hat{y} = \text{sign}(w_1 x_1 + w_2 x_2 + b).$$

Either:

- (a) Find values of w_1, w_2, b that achieve zero loss on the XOR dataset, *or*
- (b) Prove that it is impossible for any choice of w_1, w_2, b to achieve zero loss.

Hint: you may find it helpful to write out inequalities for each of the four points and see if you can find a contradiction.

Problem 1(b): Simple neural network with one hidden layer

Now consider a neural network with a single hidden layer with two hidden units and a ReLU activation function:

$$h_1 = \text{ReLU}(a_1x_1 + a_2x_2 + c_1), \quad h_2 = \text{ReLU}(d_1x_1 + d_2x_2 + c_2),$$

and an output layer with classification rule

$$\hat{y} = \text{sign}(v_1h_1 + v_2h_2 + b).$$

Either:

- (a) Find values of the parameters $(a_1, a_2, c_1, d_1, d_2, c_2, v_1, v_2, b)$ that achieve zero loss on XOR, *or*
- (b) Prove that it is impossible even with the hidden layer.

Problem 2: Backprop by Hand (30 points)

In this problem you will do a forward pass and a backward pass by hand for a tiny network with just **three** learnable parameters and a ReLU activation function. Throughout, you will apply the chain rule to compute intermediate derivatives. Given the derivatives you compute, you will be asked to update the parameters of the model using gradient descent.

Problem 2

Fix a single data point (x, y) with $x = 3$ and $y = 2$. Consider the network (a single hidden ReLU unit, followed by a linear output)

$$z = wx + b, \quad h = \text{ReLU}(z), \quad \hat{y} = vh,$$

with squared loss

$$L = \frac{1}{2}(\hat{y} - y)^2,$$

where the learnable parameters are w , b , and v . For this problem, we will initialize these parameters as $w = 1$, $b = -1$, and $v = 2$. We will use a learning rate $\eta = 0.025$ and perform one step of gradient descent:

$$w \leftarrow w - \eta \frac{\partial L}{\partial w}, \quad b \leftarrow b - \eta \frac{\partial L}{\partial b}, \quad v \leftarrow v - \eta \frac{\partial L}{\partial v}.$$

- (a) Compute the forward pass: z , h , \hat{y} , and L .
- (b) Compute $\frac{\partial L}{\partial \hat{y}}$.
- (c) Use the chain rule to compute $\frac{\partial L}{\partial v}$.
- (d) Use the chain rule to compute $\frac{\partial L}{\partial w}$.
- (e) Use the chain rule to compute $\frac{\partial L}{\partial b}$.
- (f) Apply one gradient descent update step to obtain updated parameters w' , b' , v' .
- (g) Using w' , b' , v' , compute a new forward pass on the same example and compute an updated loss L' . Show numerically that $L' < L$.

Hint: your updated loss should be very close to zero, i.e., $L' < 0.01$. If it's not, you may have made a computation error!

Problem 3: Implementing Backpropagation (50 points)

In this problem you will implement backpropagation *from scratch* to train a one-hidden-layer neural network with 32 hidden units. You will then use this code to train a model to fit a degree-3 polynomial. **Code file:** `backprop.py`. The only packages you will need are `numpy` and `matplotlib`.

Problem 3 (50 points): Implementing Backpropagation

The provided script (`backprop.py`) includes working scaffold code for:

- generating synthetic training data $x \in \mathbb{R}$ and targets $y = ax^3 + bx^2 + cx + d$,
- defining a 1-hidden-layer neural network with ReLU activation
- plotting the function learned by the neural network and training loss

Here are the variables you will need to use in your implementation:

- `x`: the input data
- `y`: the target data
- `W1`, `b1`, `W2`, `b2`: the weights and biases for the first and second layers
- `lr`: the learning rate

Your job is to fill in the missing pieces of the forward and backward passes and the gradient descent update rule. These places are marked with `### YOUR CODE HERE ###` blocks. You may wish to use `np.matmul`, `np.sum`, and `np.maximum`, and `np.mean` in your solution.

Then, you should be able to run the script and see a plot of your learned function, both on the training domain $[-2, 2]$ and on the out-of-domain domain $[-10, 10]$. You will also be able to see a plot of your training loss. As a check, your initial loss should be around 26.95, and your final loss should be around 0.05 after 2000 training steps.

Once you have the code working, answer the following questions:

- (a) How does model performance change if you decrease the hidden dimension? (e.g., `hidden_dim = 4`)
- (b) Why does the model fit well on $[-2, 2]$ but not on $[-10, 10]$? How does this relate to the statement of the universal approximation theorem from lecture?

Submission: Include the generated plots and your answers to the above questions in your PDF submission. You do not need to submit your code.