Introduction to Artificial Intelligence (INFO8006)

Exercise session 6

Multilayer perceptron

For an input vector $\mathbf{x} \in \mathbb{R}^d$ and activation function $\sigma(.)$, a MLP with L layers can be written as

$$\begin{aligned}\mathbf{h}_0 &= \mathbf{x} \\ \mathbf{h}_1 &= \sigma(\mathbf{W}_1^T \mathbf{h}_0 + \mathbf{b}_1) \\ &\vdots \\ \mathbf{h}_L &= \sigma(\mathbf{W}_L^T \mathbf{h}_{L-1} + \mathbf{b}_L) \\ \mathbf{y} &= \mathbf{h}_L\end{aligned}$$

with $\mathbf{h}_l \in \mathbb{R}^{q_l}$ defined as the the hidden vector, $\mathbf{W}_l \in \mathbb{R}^{q_{l-1} \times q_l}$ the weight matrix and $\mathbf{b}_l \in \mathbb{R}^{q_l}$ the bias vector of the l-th layer.

Training loop

For a given input-output training pair (x, y), a neural network $\Phi_\theta(.)$ parameterized by θ, a loss function $\mathcal{L}(.)$ and a learning rate λ:

1. Compute all intermediate values in the network for the given input x up to the output prediction $\hat{y} = \Phi_\theta(x)$. This is known as the forward pass.
2. Compute the symbolic gradient of the loss w.r.t. every parameters $g_i(.) = \frac{\partial \mathcal{L}}{\partial \theta_i}$ using the chain rule.
3. Evaluate the loss for your training point as $\mathcal{L}(y, \hat{y})$.
4. Backpropagate the loss and the activations through the network to compute the gradient values g_i. This is known as the backward pass.
5. Update every parameter using their gradient as

$$\theta_i \leftarrow \theta_i - \lambda g_i.$$

Universal approximation theorem

The universal approximation theorem states that a single-hidden-layer network with a finite number of hidden units can approximate any continuous function on a compact subset of $\mathbb{R}^d$ to arbitrary accuracy.

In session exercices: Ex. 1, Ex. 2

Exercise 1 Escape game (January 2022)

A new virtual escape game came out, and you decide to play it. You arrive in a 5×5 grid world where each cell (x, y) is a room with doors leading to the adjacent rooms. The game's goal is to reach the exit room as fast as possible, but its position is unknown. Furthermore, some regions of the world are full of riddles, and crossing rooms in these regions takes longer. Fortunately, a leaderboard with the players' best times is provided, starting from a few different rooms. Due to rounding errors, you assume that the best times reported in the leaderboard are measurements affected by additive Gaussian noise $\mathcal{N}(0, 1)$.

i	Starting room	Measured best time
1	$(4, 5)$	2.0
2	$(5, 3)$	3.5
3	$(3, 3)$	4.5
4	$(4, 1)$	7.0
5	$(1, 2)$	8.5

From the leaderboard, you wish to learn a heuristic approximating the best time to get to the exit, starting from room (x, y). You decide to use a small neural network as approximator, described by the following parametric function,

$$h(x, y; \phi) = \mathrm{ReLU}(xw_1 + yw_2 + w_3) + \mathrm{ReLU}(xw_4 + yw_5 + w_6)$$
$$\mathrm{ReLU}(x) = \max(x, 0),$$

where $\phi = (w_1, w_2, w_3, w_4, w_5, w_6)$ is the set of parameters/weights of the neural network.

1. Among the following sets of parameters (A, B or C), which one would you use? Justify your answer.

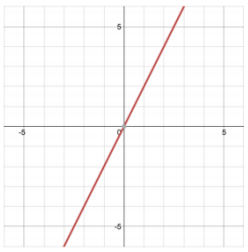
Set	w_1	w_2	w_3	w_4	w_5	w_6
ϕ_A	-1.5	1	4	1	-1.5	6
ϕ_B	-1	1.5	3	0	-1	4
ϕ_C	-2	0.5	4.5	1.5	0	5

2. You now assume a Gaussian prior $\mathcal{N}(0, 1)$ on each parameter. Which set of parameters in the table above would you now choose? Justify your answer.

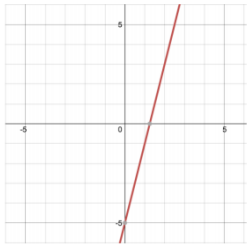
3. Discuss the procedure you would implement on a computer to find the optimal set of parameters, had the table above not been provided.

Exercise 2 Neural network representations (CS188, Spring 2024)

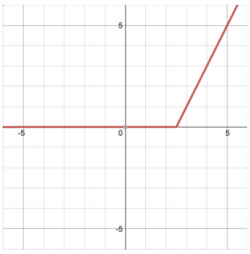
You are given a number of functions (a-h) of a single variable, x, which are graphed below. For each network structure proposed afterwards, indicate which functions they are able to represent. When possible, indicate appropriate values for the parameters.



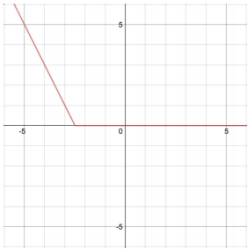
(a) $2x$



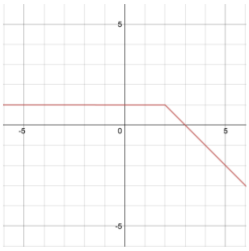
(b) $4x - 5$



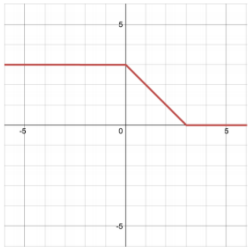
(c) $\begin{cases} 2x-5 & x \geq 2.5 \\ 0 & x < 2.5 \end{cases}$



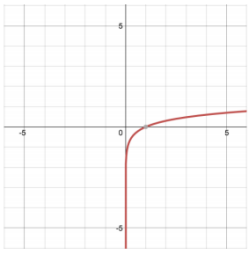
(d) $\begin{cases} -2x-5 & x \leq -2.5 \\ 0 & x > -2.5 \end{cases}$



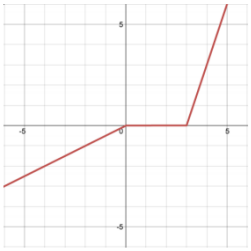
(e) $\begin{cases} -x+3 & x \geq 2 \\ 1 & x < 2 \end{cases}$



(f) $\begin{cases} 3 & x \leq 0 \\ 3-x & 0 < x \leq 3 \\ 0 & x > 3 \end{cases}$

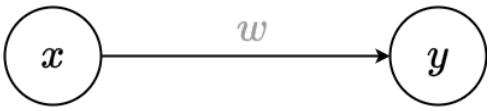


(g) $\log(x)$

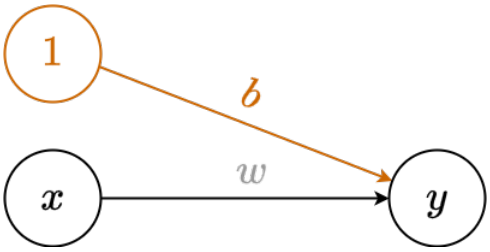


(h) $\begin{cases} 0.5x & x \leq 0 \\ 0 & 0 < x \leq 3 \\ 3x-9 & x > 3 \end{cases}$

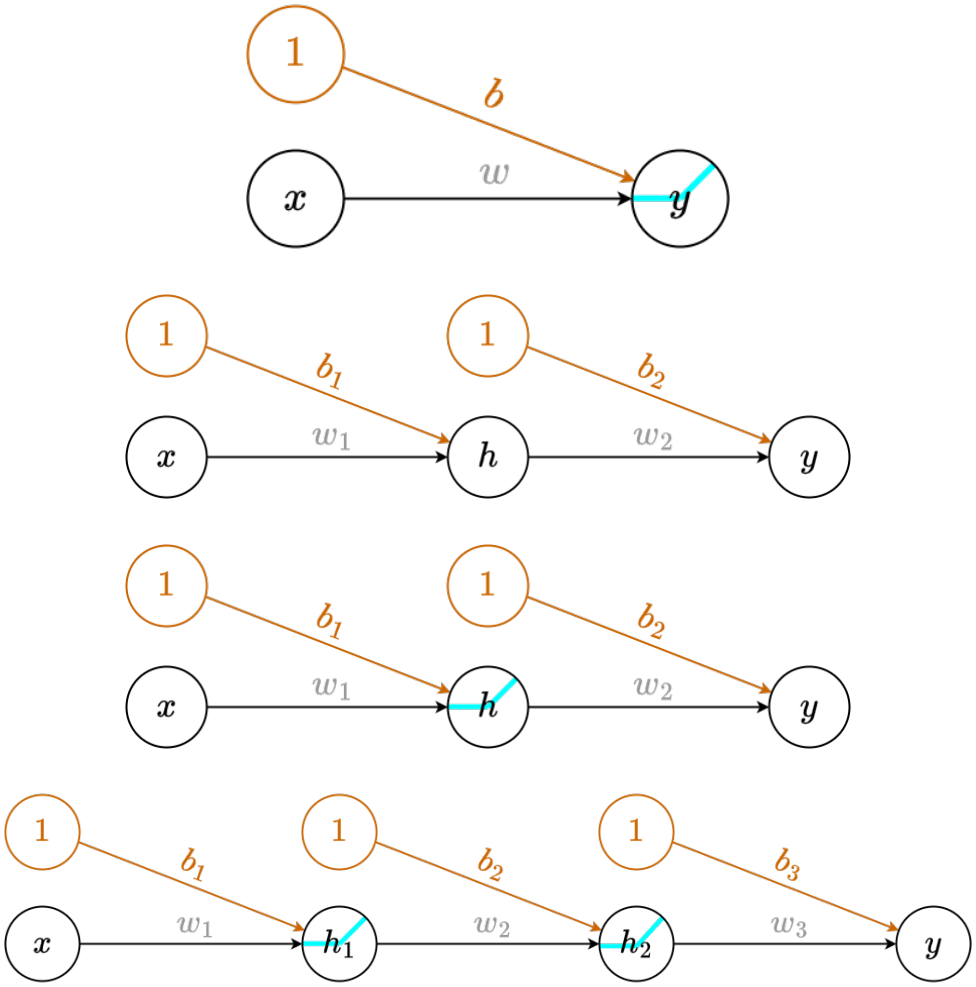
1. We start with a linear transformation of the scalar input x, weight w, and output y, such that $y = wx$.



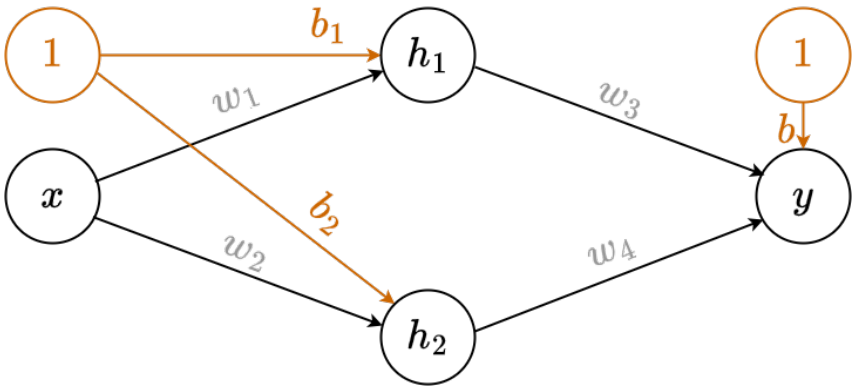
2. We then introduce a bias term b, such that $y = wx + b$.



3. We now introduce a non-linearity $\sigma(.)$ into the network. We use the ReLU non-linearity, which has the form ReLU(x) = max(0, x).
4. Now we consider neural networks with multiple affine transformations, as depicted below.
5. We now add a ReLU non-linearity to the network between the affine transformations.
6. Now we add another hidden layer to the network, as depicted below. You do not have to guess parameters values anymore.

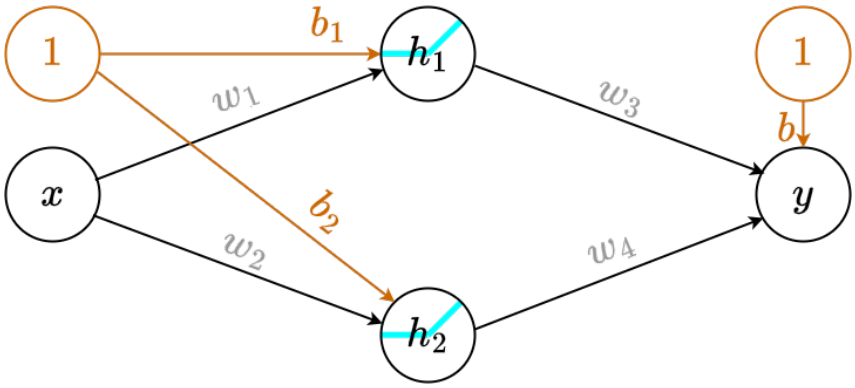


7. We'd like to consider using a neural net with just one hidden layer, but larger. Let's first consider using just two affine functions, with no nonlinearity in between. You do not have to guess parameters values anymore.



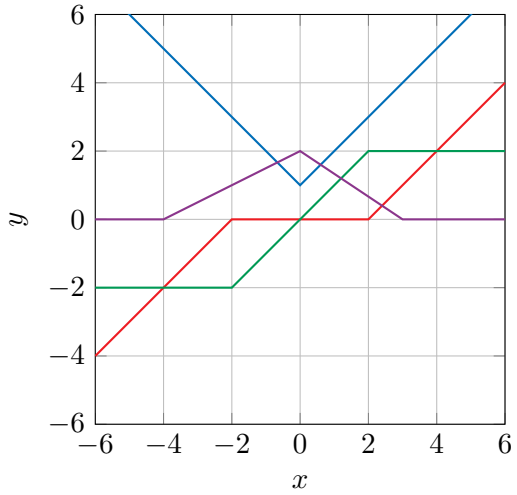
8. Now we'll add a non-linearity between the two affine layers, to produce the neural network below with a hidden layer of size 2. You do not have to guess parameters values anymore.

9. Are there functions that can't be represented by all proposed networks? If yes, explain why and what you would need to model them.



Exercise 3 ReLU

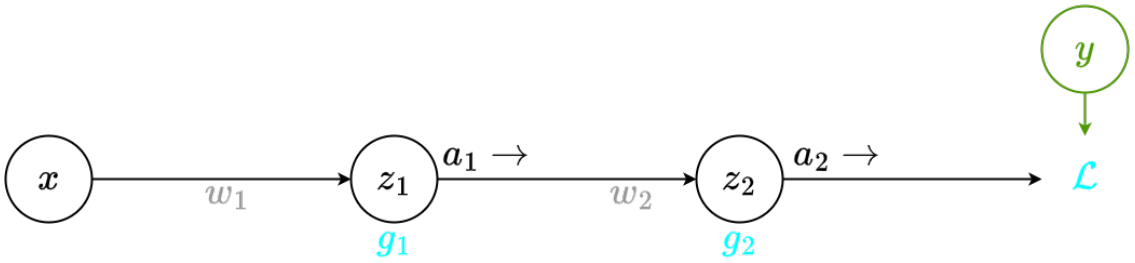
For each of the piecewise-linear functions below, write a function $y = f(x)$ as a composition of sums $(+, -)$, ReLU ($\text{ReLU}(x) = \max(x, 0)$) non-linearities, and real parameters (weights and biases) that exactly matches the function over $\mathbb{R}$.



For example, $y = \text{ReLU}(x + 2) - \text{ReLU}(-2x)$ is a valid function.

Exercise 4 Classification (CS188, Spring 2024)

Consider the following simple neural network for binary classification. Here x is a single real-valued input feature with an associated class y (0 or 1). There are two weight parameters w_1 and w_2, and non-linearity functions g_1 and g_2. The network will output a value a_2 between 0 and 1, representing the probability of being in class 1. We will be using a loss function $\mathcal{L}$ to compare the prediction a_2 with the true class y.



1. Write down the forward pass on this network, writing the output values for each node z_1, a_1, z_2 and a_2 in terms of the node's input values.
2. Derive the arguments of the loss $\mathcal{L}(a_2, y)$ in terms of the input x, weights w_i, and activation functions g_i.
3. Using the chain rule, differentiate $\mathcal{L}$ w.r.t. w_2. Write your expression as a product of partial derivatives at each node: i.e. the partial derivative of the node's output with respect to its inputs. (Hint: the series of expressions you wrote in part 1 will be helpful; you may use any of those variables.)
4. Motivate a choice for the output activation function g_2 given the nature of the problem. Motivate also a loss function. Give the expression of the gradients of both functions w.r.t. to their inputs.
5. We set the loss function to the binary cross-entropy

$$\mathcal{L}(a_2, y) = -y \log a_2 - (1 - y) \log(1 - a_2),$$

   and g_1 and g_2 are both sigmoid functions $\sigma(z) = \frac{1}{1+e^{-z}}$. If you made this choice in the previous question, use your results to get the expression of $\frac{\partial \mathcal{L}}{\partial w_2}$. Otherwise, start by expressing $\frac{\partial \sigma(z)}{\partial z}$ and $\frac{\partial \mathcal{L}}{\partial a_2}$.
6. Now use the chain rule to express $\frac{\partial \mathcal{L}}{\partial w_1}$ as a product of partial derivatives at each node of interest.
7. Finally, write $\dfrac{\partial \mathcal{L}}{\partial w_1}$ in terms of x, y, w_i, a_i, z_i.
8. What is the gradient descent update for w_1 with step-size λ in terms of the values computed above?

Exercise 5 Training loop (INFO8006, January 2024)

Let us consider a neural network f with one hidden layer taking as input a scalar $x \in \mathbb{R}$ and producing as output a scalar $y = f(x; \theta) \in \mathbb{R}$. The neural network is defined as

$$f(x; \theta) = w_1 \text{ReLU}(w_2 x + w_3) + w_4 \text{ReLU}(w_5 x + w_6) + w_7 \text{ReLU}(w_8 x + w_9),$$

where $\theta = (w_1, w_2, w_3, w_4, w_5, w_6, w_7, w_8, w_9)$ is the set of parameters and $\text{ReLU}(x) = \max(x, 0)$ is the rectified linear unit function.

1. Draw the computation graph representing the neural network and the flow of information from inputs to outputs. Your diagram should be a directed graph that follows the following conventions:
   - circled nodes correspond to variables (input, output, parameters or intermediate variables),
   - squared nodes correspond to primitive operations (addition, multiplication, ReLU) and produce an intermediate variable as output,
   - directed edges correspond to the flow of information, from inputs to outputs.
2. For $\theta = (-1, \frac{1}{2}, 0, -4, 1, -2, 4, 1, -5)$, draw the function $y = f(x; \theta)$ for $x \in [-10, 10]$.

3. Using the data point $(x, y) = (10, -15)$ and the value of θ given above, we want to fine-tune the parameter w_8 of the neural network such that $f(x; \theta)$ produces a more accurate prediction of y.

   (a) Evaluate the squared error loss at the data point $(x, y) = (10, -15)$ and the current value of θ.

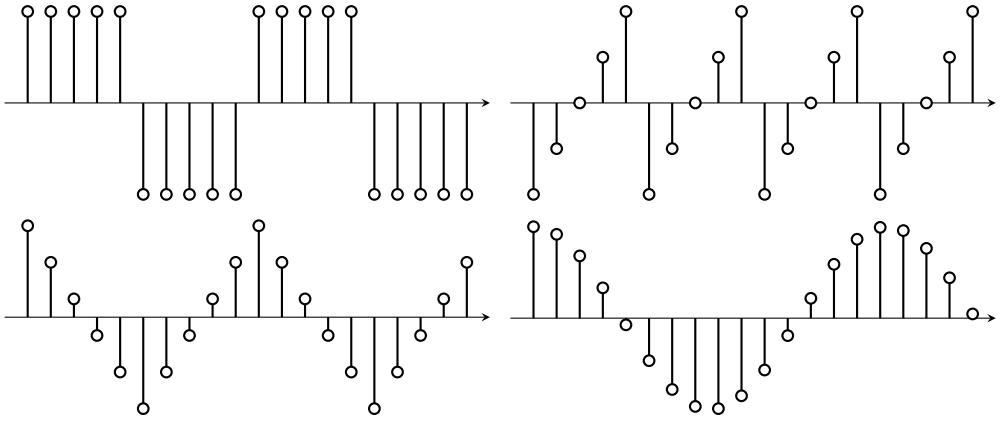
   (b) Derive an expression for the derivative of the squared error loss with respect to w_8.

   (c) Update the parameter w_8 using one step of gradient descent with a learning rate $\gamma = 0.0005$.

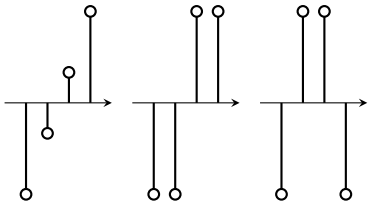
   (d) Verify that the value of the loss function has decreased after the update.

Exercise 6 Convolution

Represent the convolution $x \circledast u$ of each of the following signals x



with each of the following convolution kernels u.



Exercise 7 Learning to play Pacman (August 2020)

You observe a Grandmaster agent playing Pacman. How can you use the moves you observe to train your own agent?

1. Describe formally the data you would collect, the inference problem you would consider, and how you would solve it.
2. How would you design a neural network to control your agent? Define mathematically the neural network architecture, its inputs, its outputs, its parameters, as well as the loss you would use to train it.
3. Discuss the expected performance of the resulting agent when (a) the Grandmaster agent is optimal, and (b) the Grandmaster agent is suboptimal.

Quiz

We build a MLP with 2 hidden layers of 32 units, ReLU activations and squared error loss. We discard the biases from our linear layers. We know that our input is a vector of 2 elements and the output is a scalar. How much parameters do we have in the network?

- ☐ 1024.
- ☐ 1120.
- ☐ 1185.
- ☐ Another value than the ones proposed above.

We add biases in the network proposed above. What is the number of parameters of the network?

- ☐ Another value than the ones proposed below.
- ☐ 1089.
- ☐ 1185.
- ☐ 2048.
- ☐ 2240.

We have an image of shape (H, W), we want to pass it through a layer which reduces each dimension by a constant K, i.e. produces an output image of shape $(H - K, W - K)$.

- ☐ It can be done with a convolution layer with a kernel (K, K) or a linear layer with weight matrix $(HW, (H - K)(W - K))$.
- ☐ It can be done with a convolution layer with a kernel $(K + 1, K + 1)$ or a linear layer with weight matrix $(HW, (H - K)(W - K))$.
- ☐ It can be done with a convolution layer with a kernel $(K - 1, K - 1)$ or a linear layer with weight matrix (HW, KK).
- ☐ It can be done with a convolution layer with a kernel $(K + 1, K + 1)$ or a linear layer with weight matrix (HW, KK).

From the previous question, we double the number of input pixels by setting the shape of the input image to (H', W) with $H' = 2H$. The latter implies that the output image should be of shape $(H' - K, W - K)$. If only look at weight parameters (no biases)...

- ☐ The number of parameters of both the linear and convolution layer are doubled.
- ☐ The linear layer will have twice more parameters and the convolution twice less.
- ☐ The linear layer will have twice more parameters and the convolution will not change.
- ☐ Nothing changes neither for the linear layer nor the convolution one.

In deep learning, a layer in a multi-layer perceptron is defined as ...

- ☐ $\mathbf{h} = \sigma(\mathbf{W}^T\mathbf{x} + \mathbf{b})$, where σ is the standard deviation function.
- ☐ $\mathbf{h} = \sigma(\mathbf{W}^T\mathbf{x} + \mathbf{b})$, where σ is an activation function, such as the sigmoid function.
- ☐ $\mathbf{h} = \sigma(\mathbf{W}^T + \mathbf{x} - \mathbf{b})$, where $\mathbf{W} \in \mathbb{R}^{d \times q}$ is matrix of weights.
- ☐ $\mathbf{h} = \sigma(\mathbf{W}^T\mathbf{x} + \mathbf{b})$, where $\mathbf{b} \in \mathbb{R}^d$ is the most likely vector of hidden states given $\mathbf{x}$.

Arnaud is trying to perform gradient descent on a function $f(x)$ using the update

$$x_{t+1} := x_t - \frac{\partial f}{\partial x}(x_t).$$

Is gradient descent guaranteed to converge to the global minimum of f?

- ☐ Yes, since he's updating using the gradient of x.
- ☐ Yes, but not for the reason above.
- ☐ No, since he is updating updating x in the wrong direction.
- ☐ No, but not for the reason above.