PyTorch: Computation Graphs and Autograd

In this lab you will learn how to construct a computation graph to calculate gradients manually. You will then use PyTorch to calculate the same gradients to check your results.

The main data structure in PyTorch is called a **tensor**, a multi-dimensional data container. Tensors are built on NumPy arrays, so basic operations are similar to NumPy arrays. But tensors can also **automatically compute gradients** and **run on GPUs**.

Lessons 2, 3, and 4 of the Patrick Loeber PyTorch Tutorial will be helpful in completing the exercises.

- Lesson2: Tensor Basics
- Lesson3: Autograd
- Lesson4: Backpropagation

Other helpful resources from the PyTorch site:

- Tensor Tutorial
- torch.Tensor official docs

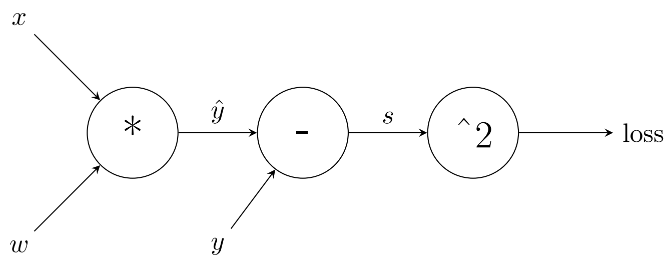
Before you begin

1. Create a new python project and a virtual environment for it in the usual way.
2. Add the starter code in Lab4.py to your project.
3. Install the libraries used in the import statements.

Task 1: Simple Computation Graph, Backprop

1. Do this part on paper: Follow the example in Patrick Loeber's Backpropagation video (lesson 4) to calculate $\frac{\partial loss}{\partial w}$, where $\hat{y} = x * w$ and loss $= (\hat{y} - y)^2$. The bias is omitted in this example.

   First we draw the computation graph for our loss function:



   ToDo: Forward pass and loss: given x=4, y=1, w=0.5, compute the loss, and all node outputs ($\hat{y}$ and s).

   ToDo: Backpropagation: use the chain rule to calculate $\frac{\partial loss}{\partial w}$, the gradient of the loss function with respect to w:

   $$\frac{\partial loss}{\partial w} = \frac{\partial loss}{\partial s} * \frac{\partial s}{\partial \hat{y}} * \frac{\partial \hat{y}}{\partial w}$$

   $$loss = s^2 \quad \Bigg| \quad \frac{\partial loss}{\partial s} = \frac{\partial s^2}{\partial s} = 2s$$

   $$s = \hat{y} - y \quad \Bigg| \quad \frac{\partial s}{\partial \hat{y}} = \frac{\partial(\hat{y} - y)}{\partial \hat{y}} = 1$$

   $$\hat{y} = x * w \quad \Bigg| \quad \frac{\partial \hat{y}}{\partial w} = \frac{\partial(x * w)}{\partial w} = x$$

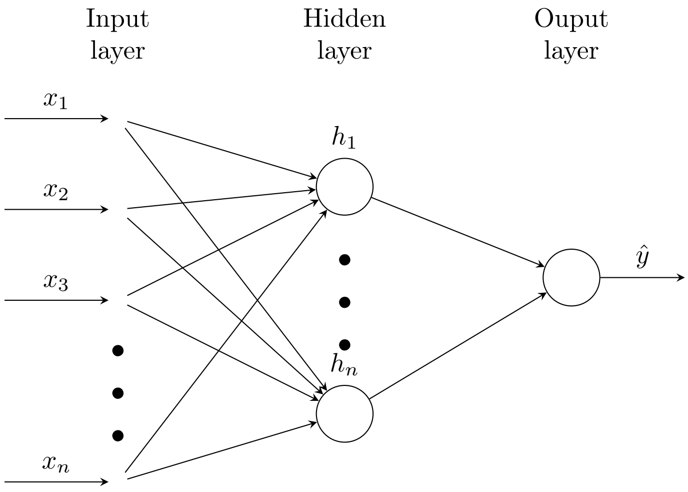
   ToDo: put the pieces together to calculate $\frac{\partial loss}{\partial w}$.

2. Verify your result by running check_result() in Lab4.py, which implements one forward and one backward pass.

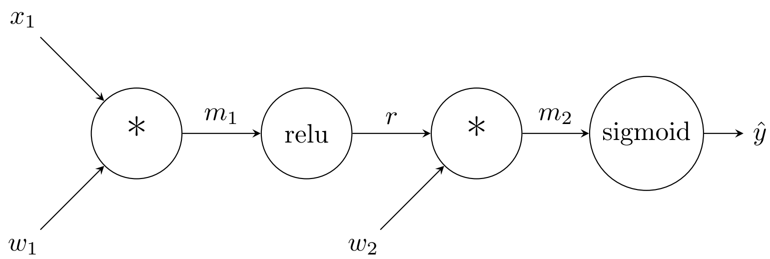
3. Implement a training loop in train() according to the instructions in the code.

Task 2: FNN Computation Graph, Backprop

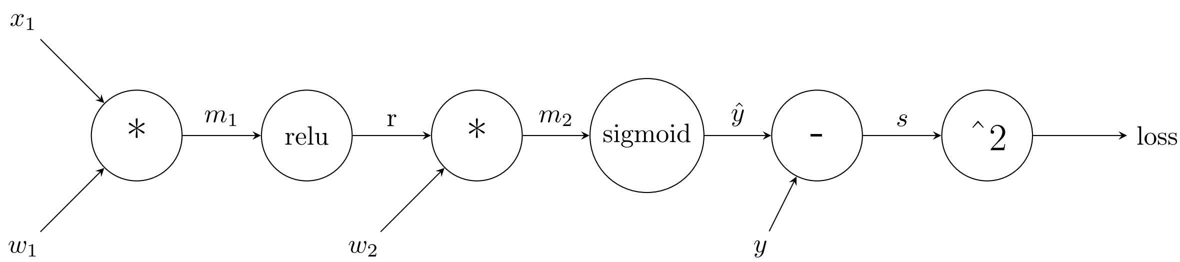
Consider the following FeedForward Network with 1 hidden layer:



Looking only at x_1, and including a ReLU activation function on the hidden layer and a sigmoid activation function on the output layer, we get the computation graph for the forward pass $\hat{y}$:



The loss function, $(\hat{y} - y)^2$, must also be included in the computation graph. Remember that our goal is to compute the gradients of the loss function with respect to the weights w_1 and w_2.



ToDo: Calculate the loss and all node outputs, given $x_1 = 4$, $w_1 = 0.5$, $w_2 = 0.2$, y = 1.

Recall the formulas for ReLU and sigmoid:

$$ReLU(x) = \begin{cases} x & x > 0, \\ 0 & \text{otherwise} \end{cases} \qquad sigmoid(x) = \frac{1}{(1 + exp(-x))}$$

Backpropagation

Now that we have the loss function represented as a computation graph, and we know the output values of all nodes (m_1, r, m_2,...), we can begin backpropagation.

You will need the partial derivatives of ReLU and sigmoid:

$$\frac{\partial ReLU(x)}{\partial x} = \begin{cases} 1 & x > 0, \\ 0 & \text{otherwise} \end{cases} \qquad \frac{\partial sigmoid(x)}{\partial x} = x(1-x)$$

Start with the gradient wrt w_2:

Moving from right to left, calculate the derivatives at each node, working back to w_2:

$$\frac{\partial loss}{\partial w_2} = \frac{\partial loss}{\partial s} * \frac{\partial s}{\partial \hat{y}} * \frac{\partial \hat{y}}{\partial m_2} * \frac{\partial m_2}{\partial w_2}$$

$loss = s^2$	$\frac{\partial loss}{\partial s} = \frac{\partial s^2}{\partial s} = 2s$
$s = \hat{y} - y$	$\frac{\partial s}{\partial \hat{y}} = \frac{\partial(\hat{y} - y)}{\partial \hat{y}} = 1$
$\hat{y} = \text{sigmoid}(m_2)$	$\frac{\partial \hat{y}}{\partial m_2} = \frac{\partial sigmoid(m_2)}{\partial m_2} = m_2(1 - m_2)$
$m_2 = r * w_2$	$\frac{\partial m_2}{\partial w_2} = \frac{\partial(r * w_2)}{\partial w_2} = r$

ToDo: put the pieces together to calculate the gradient of the loss with respect to w_2

$$\frac{\partial loss}{\partial w_2} =$$

ToDo: Finish the backpropagation and calculate $\frac{\partial loss}{\partial w_1}$. You have already calculated the partial derivatives up to node m_2. Start with $\frac{\partial m_2}{\partial r}$, **NOT** $\frac{\partial m_2}{\partial w_2}$, and continue working back to w_1.

$$\frac{\partial loss}{\partial w_1} =$$