Artificial Intelligence in Predictive Health and Laboratory Research

Somdatta Goswami
Civil and Systems Engineering
Johns Hopkins University

Rosy Joshi-Mukherjee and Sherry Moore
School of Pharmacy
Notre Dame of Maryland University

Hands on Exercise from Day 1

Exercise

In this project, we develop a binary classification model using PyTorch to predict a patient's gender (Male or Female) based on common blood test values:

- Hemoglobin
- Platelet Count
- White Blood Cells
- Red Blood Cells
- MCV (Mean Corpuscular Volume)
- MCH (Mean Corpuscular Hemoglobin)
- MCHC (Mean Corpuscular Hemoglobin Concentration)

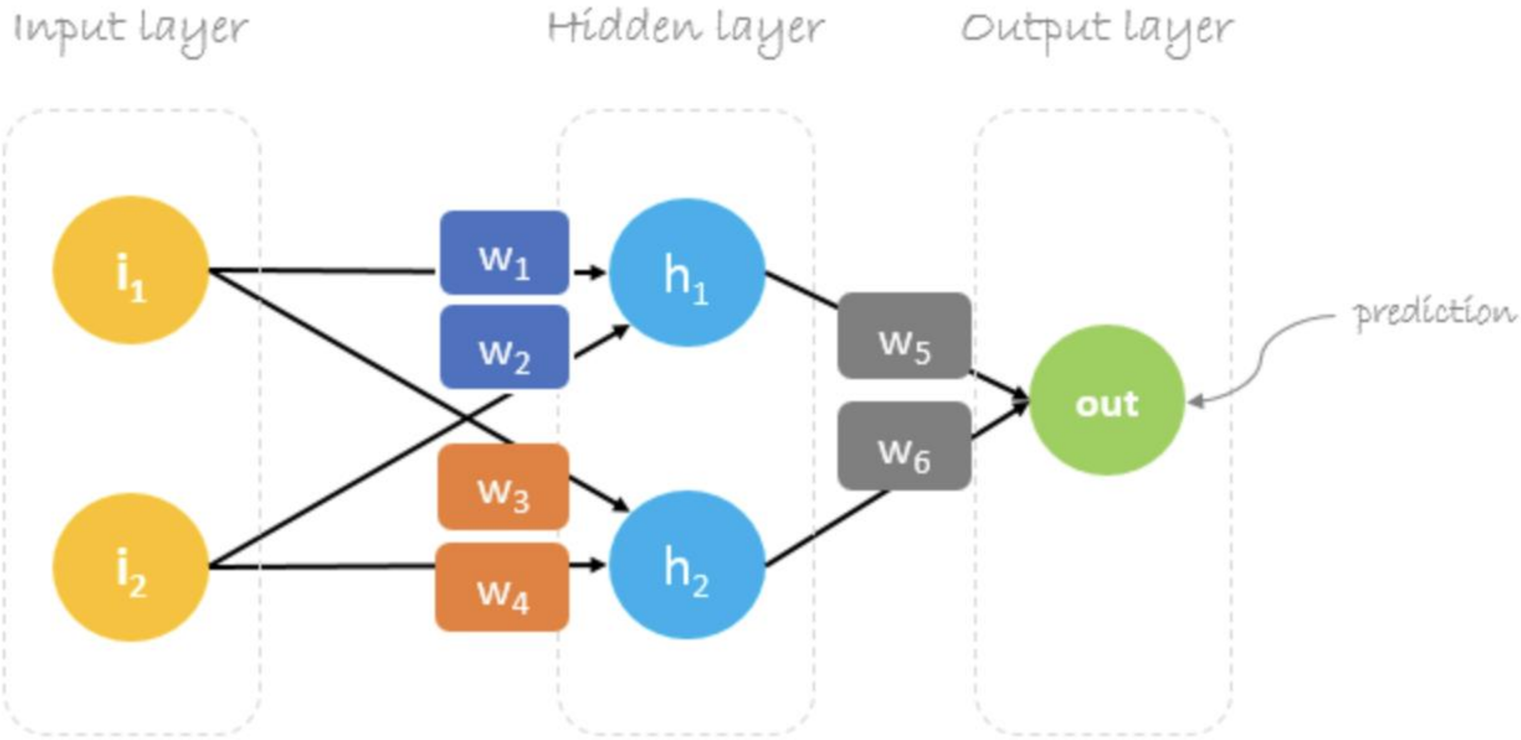
Link: https://github.com/Centrum-IntelliPhysics/Artificial-Intelligence-in-Predictive-Health-and-Laboratory-Research/tree/main for the code and the dataset

Test case

- Hemoglobin: 11.8
- Platelet Count: 280000
- White Blood Cells: 7300
- Red Blood Cells: 4.3
- MCV: 85
- MCH: 28
- MCHC: 32

Can you predict the patient's gender based on the given blood test values?

Neural Networks



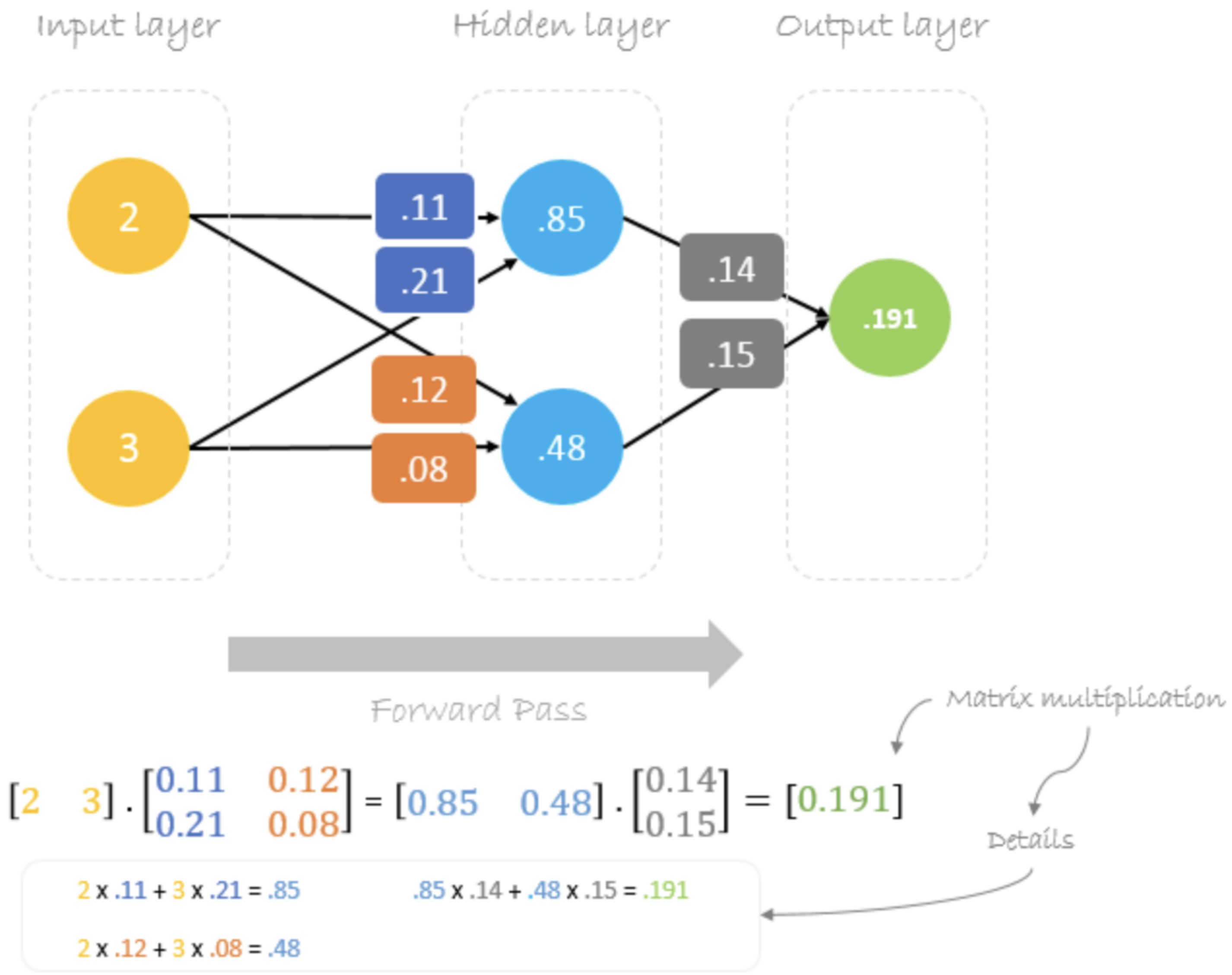
Data

Our dataset has one sample with two inputs and one output.

Our single sample is as following `inputs=[2, 3]` and `output=[1]`.

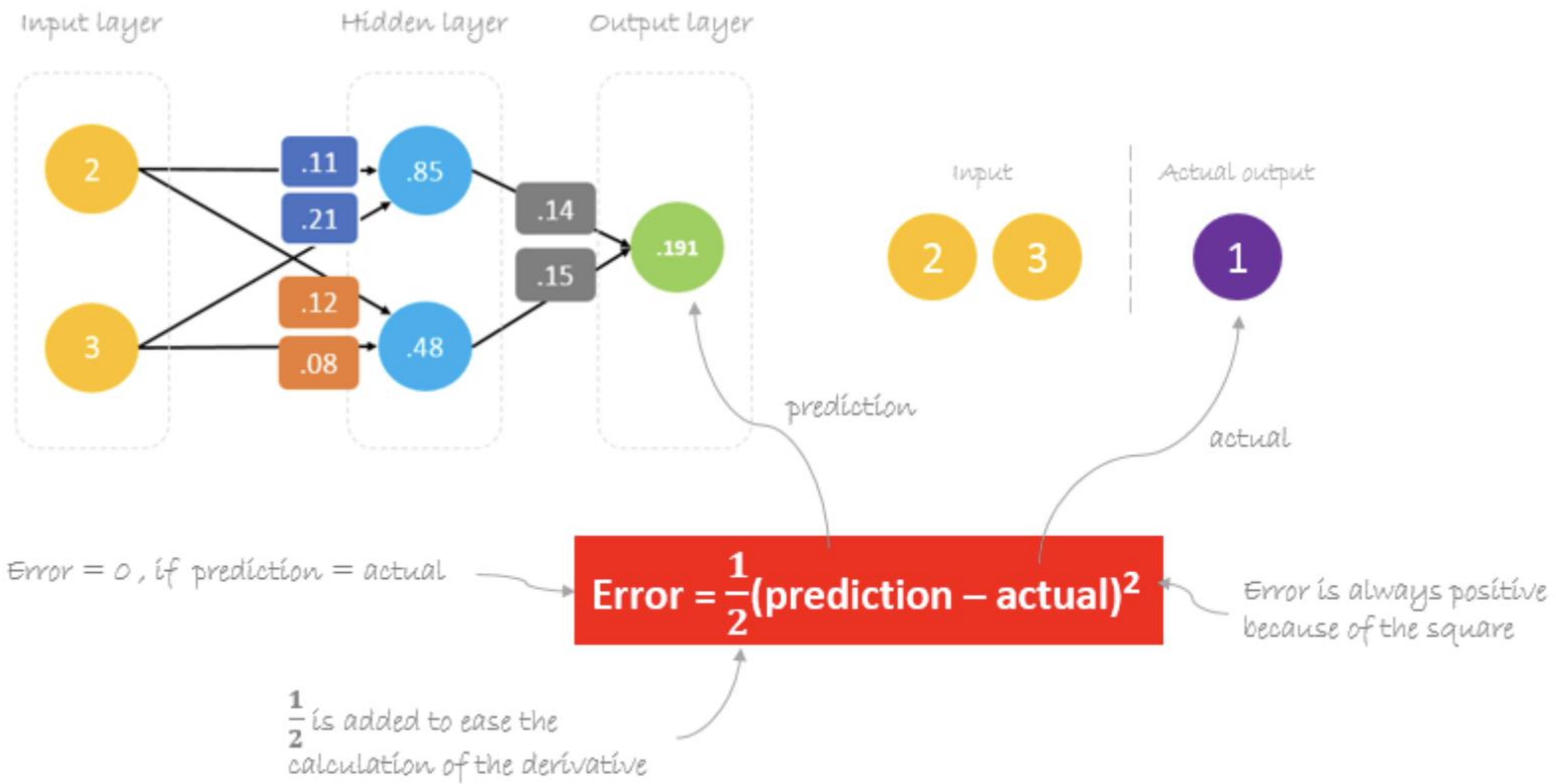
Forward pass

We will use given weights and inputs to predict the output. Inputs are multiplied by weights; the results are then passed forward to next layer.



Calculating Error

Now, it's time to find out how our network performed by calculating the difference between the actual output and predicted one. It's clear that our network output, or **prediction**, is not even close to **actual output**. We can calculate the difference or the error as following.



$$\text{Error} = \frac{1}{2}(0.191 - 1.0)^2 = 0.327$$

Reducing Error

Our main goal of the training is to reduce the **error** or the difference between **prediction** and **actual output**. Since **actual output** is constant, “not changing”, the only way to reduce the error is to change **prediction** value. The question now is, how to change **prediction** value? By decomposing **prediction** into its basic elements, we can find that **weights** are the variable elements affecting **prediction** value. In other words, in order to change **prediction** value, we need to change **weights** values.

$$\textbf{prediction} = out$$

$$\textbf{prediction} = (h_1)\, w_5 + (h_2)\, w_6$$

$$h_1 = i_1 w_1 + i_2 w_2$$
$$h_2 = i_1 w_3 + i_2 w_4$$

$$\textbf{prediction} = (i_1\, \boldsymbol{w_1} + i_2\, \boldsymbol{w_2})\, \boldsymbol{w_5} + (i_1\, \boldsymbol{w_3} + i_2\, \boldsymbol{w_4})\, \boldsymbol{w_6}$$

to change ***prediction*** value,
we need to change ***weights***

The question now is **how to change or update the weights value so that the error is reduced?**
The answer is **Backpropagation!**

Backpropagation

Backpropagation, short for “backward propagation of errors”, is a mechanism used to update the **weights** using gradient descent. It calculates the gradient of the error function with respect to the neural network’s weights. The calculation proceeds backwards through the network. **Gradient descent** is an iterative optimization algorithm for finding the minimum of a function; in our case we want to minimize th error function. To find a local minimum of a function using gradient descent, one takes steps proportional to the negative of the gradient of the function at the current point.

Derivative of Error with respect to weight

Old weight

$$ {}^{*}W_x = W_x - a \left(\frac{\partial Error}{\partial W_x} \right) $$

New weight

Learning rate

Backpropagation

For example, to update w_6, we take the current w6 and subtract the partial derivative of **error** function with respect to w_6. Optionally, we multiply the derivative of the **error** function by a selected number to make sure that the new updated **weight** is minimizing the error function; this number is called ***learning rate***.

$$^{*}W_6 = W_6 - a\left(\frac{\partial Error}{\partial W_6}\right)$$

Hands on Exercise

Exercise

This project demonstrates how to use a simple regression neural network in PyTorch to predict hemoglobin concentration (Hb) based on Red and Infrared (IR) light intensity values. These are inspired by signal readings commonly used in non-invasive health monitoring devices like pulse oximeters.

Data: `Final Dataset Hb PPG.csv` file contains historical results and haemoglobin values.

Link: https://github.com/Centrum-IntelliPhysics/Artificial-Intelligence-in-Predictive-Health-and-Laboratory-Research/tree/main for the code and the dataset

Thank you