Homework 1 - Perceptron Forward Propagation

Advanced Topics in Neural Networks

03 October 2023

1 Perceptron

A perceptron is one of the simplest forms of a neural network, consisting of a single layer. It takes multiple input features and produces one binary output. The output is calculated as a weighted sum of its input, added to a bias and then passed through an activation function.

$$y = \begin{cases} 0 & \text{if } w \cdot x + b \leq 0 \\ 1 & \text{if } w \cdot x + b > 0 \end{cases} \tag{1}$$

where

- w is the weight vector,
- x is the input vector,
- b is the bias,
- y is the output of the perceptron after applying an activation function to the weighted sum of the inputs.

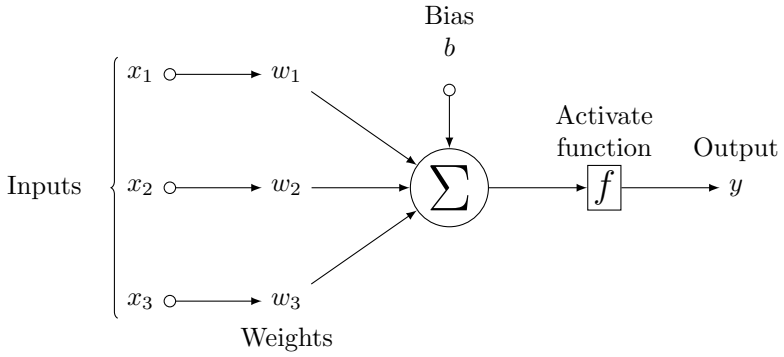

Figure 1: Neuron structure

Forward Propagation refers to the initial phase of training a neural network, where the input data is fed forward through the network

2 Forward Propagation in a Perceptron

The process of forward propagation in a perceptron involves the following steps:

1. Calculate the weighted sum of the inputs. The weights and bias are usually initialized randomly.
$$z = w \cdot x + b$$
2. Pass the result through an activation function to get the output of the perceptron. For binary classification tasks, the Heaviside step function is often used as the activation function.
$$y = \begin{cases} 0 & \text{if } z \leq 0 \\ 1 & \text{if } z > 0 \end{cases}$$
3. Compare the perceptron's output with the actual target value to compute the error.
$$\text{Error} = \text{Target} - \text{Output}$$
4. Update the weights and bias in the backpropagation step (not covered in this document).

3 Exercise

In this exercise, you are tasked with implementing the forward propagation process for a neural network with 784 inputs and 10 outputs using PyTorch. This network can be thought of as consisting of 10 perceptrons, each responsible for predicting one of the 10 output classes. Your task is to implement this forward propagation manually, using only basic PyTorch operations. Do **not** use built-in layers like `nn.Linear` or similar high-level APIs.

3.1 Problem Statement

Given an input tensor `X` of shape $(m, 784)$, where m is the number of examples and 784 is the number of features (input neurons), and a weight tensor `W` of shape $(784, 10)$, and a bias tensor `b` of shape $(10,)$, compute the output of the network for each example in the batch. Each column in the weight tensor corresponds to the weights for one of the 10 output neurons (perceptrons).

3.2 Constraints

- Use only basic PyTorch operations like matrix multiplication, element-wise addition, and indexing. Don't use API's such as torch.nn or torch.functional.
- Assume the activation function to be the sigmoid function, defined as
$$y = \sigma(z) = \frac{1}{1 + \exp(-z)}$$

- The bias tensor `b` should be added to the weighted sum before the activation function is applied.

3.3 Expected Outcome

Write a Python function named `forward_propagation` that takes `X`, `W`, and `b` as inputs and returns the output tensor after applying the forward propagation steps.

Function Signature:

```
def forward_propagation(X: Tensor, W: Tensor, b: Tensor) -> Tensor:
    pass  # Your implementation here
```

Input:

- `X`: A 2D PyTorch tensor of shape $(m, 784)$ containing the input features.
- `W`: A 2D PyTorch tensor of shape $(784, 10)$ containing the weights for the 10 perceptrons.
- `b`: A 1D PyTorch tensor of shape $(10,)$ containing the biases for the 10 perceptrons.

Output:

- A 2D PyTorch tensor of shape $(m, 10)$ containing the outputs of the 10 perceptrons for each of the m examples.

Evaluation: Your implementation will be considered correct if the output tensor is computed correctly according to the forward propagation steps described earlier in this document, adhering to the specified constraints.

Extra Download the MNIST dataset, load the images, and forward propagate them thought your network.