The idea here is to create a two layers network to solve a binary classification problem using the Sigmoid activation function to perform linear transformation to input x:

$$1/(1+e^{(-z)})$$

with below specifications:

- The first layer will have two neurons A and B. The purpose of this layer is to compute two separate conditions based on the input:

  A. Check if $y < 2x + 3$:
     rearrange the function to $0 < 2x - y + 3$;
     assign weight [2.0, -1.0] to x and y;
     assign bias 3.0;
     The output of this neuron will be high if the input values (x, y) satisfy the inequality, and low if otherwise.

  B. Check if $x > 0$:
     Assign weight [1.0, 0.0] to x and y; (there is no y in this condition so we assign 0 to it);
     Assign bias 0;
     The output of this neuron will be high if the input value x is greater than 0, and low if otherwise.

  After computing these 2 conditions, the outputs are passed through the sigmoid function, which squashes the values between 0 and 1.

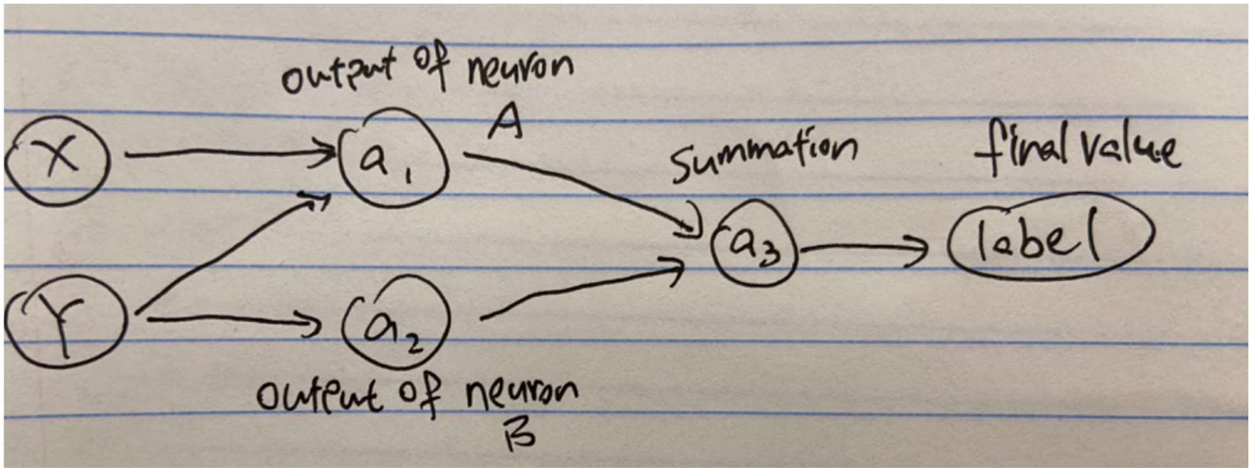
- The second layer contains only a single neuron (AND logic). The purpose of this layer is to combine the results of two conditions from layer 1 using a logical AND operation. However, we need to manually input and fine-tune the weights and bias.

  After rounds of adjustments, I have found that the weight of [10, 10] and the bias of [-14] works fine for most inputs.

  For example, when the outputs of neuron A and neuron B from layer 1 are both close to 1, the summation before the activation function in this neuron will be $10 * 1 + 10 * 1 - 14 = 6$. Passing this value to the sigmoid function will get a value very close to 1 (the probability of getting 1), yielding a final output of 1.

  Whereas if the outputs from either or both of the outputs from layer 1 are very low (or if both are relatively low), lets say (0.9, 0.4), then the value to be passed to the activation function will be $10 * 0.9 + 10 * 0.4 - 14 = -1$. Passing this value to the sigmoid function will get a value closer to 0 (the probability of getting 1), yielding a final result of 0.

Visualization:



Refer to my colab link for a model I made using pytorch to solve this problem:

https://colab.research.google.com/drive/1_8X-LkL50pysFPbrT68saHrC1MMKqAM_#scrollTo=bhH4Bgmr_4hI

```
import torch
import torch.nn as nn
torch.set_printoptions(precision=4, sci_mode=False)
```

```
# define the neural network
class binary_classification(nn.Module):
    def __init__(self):
        super(binary_classification, self).__init__()

        # first Layer
        self.layer1 = nn.Linear(2, 2)  # 2 neurons: one for each condition

        # second Layer to combine the two conditions
        self.layer2 = nn.Linear(2, 1)

        # Sigmoid activation to squash values between 0 and 1
        self.sigmoid = nn.Sigmoid()

    def forward(self, x):
        x = self.layer1(x)
        x = self.sigmoid(x)
        x = self.layer2(x)
        x = self.sigmoid(x)
        #x = self.quantizer(x) #uncomment this line to see a binary final output
        return x

    def quantizer(self, x):
        return (x > 0.5).to(torch.float)


# create the network
net = binary_classification()


# custom weights and biases
with torch.no_grad():
    # neuron A for y < 2x + 3
    net.layer1.weight[0] = nn.Parameter(torch.tensor([2.0, -1.0]))
    # neuron B for x > 0
    net.layer1.weight[1] = nn.Parameter(torch.tensor([1.0, 0.0]))
    net.layer1.bias = nn.Parameter(torch.tensor([3.0, 0.0]))

    # Neuron to combine A and B with AND logic
    net.layer2.weight = nn.Parameter(torch.tensor([[10.0, 10.0]]))  # high positive weights to ensure both conditions are met
    net.layer2.bias = nn.Parameter(torch.tensor([-14.0]))  # bias to ensure the neuron only activates if both prior neurons output high value
```

```
# test
inputs = torch.tensor([[1.0, 2.0],
                       [-1.0, 3.0],
                       [1.0, 3.9],
                       [2.0, 10.0]])
outputs = net(inputs)
print(outputs)
```

```
tensor([[    0.9446],
        [    0.0000],
        [    0.6928],
        [    0.0089]], grad_fn=<SigmoidBackward0>)
```

0s completed at 10:18 PM