

# Deeper networks

Puteaux, Fall/Winter 2020-2021

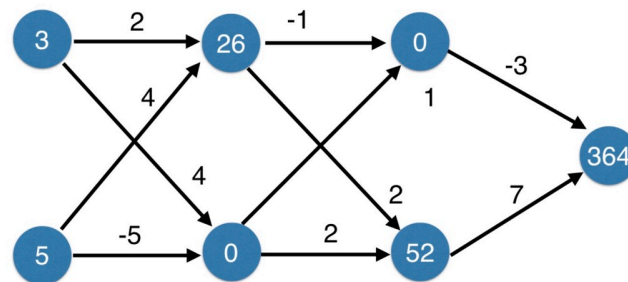
```
#####  
##                               ##  
##  Deep Learning in Python  ##  
##                               ##  
#####
```

§1 Introduction to Deep Learning in Python

§1.1 Basics of deep learning and neural networks

§1.1.4 Deeper networks

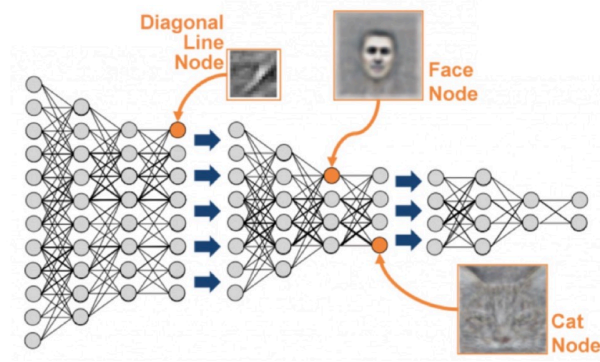
1. How do multiple hidden layers function?



Calculate with ReLU Activation Function

2. Why is deep learning also sometimes called representation learning?

- Deep networks internally build representations of patterns in the data; in this way, partially replace the need for feature engineering.
- Subsequent layers build increasingly sophisticated representations of raw data.

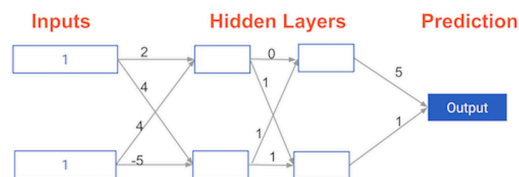


### 3. How does the deep learning process?

- The modeler doesn't need to specify the interactions.
- When training the model, the neural network gets weights that find the relevant patterns to make better predictions.

### 4. Practice question for the forward propagation in a deeper network:

- There is a model with two hidden layers. The values for an input data point are shown inside the input nodes. The weights are shown on the edges/lines. What prediction would this model make on this data point?
- Assume the activation function at each node is the *identity function*. That is, each node's output will be the same as its input. So the value of the bottom node in the first hidden layer is  $-1$ , and not  $0$ , as it would be if the ReLU activation function was used.



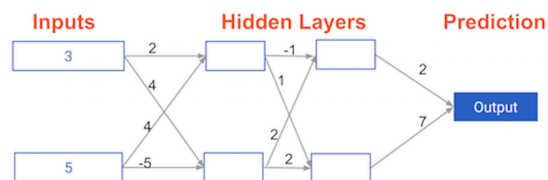
☒ 0.

☐ 7.

☐ 9.

### 5. Practice exercises for deeper networks:

#### ► Diagram of the forward propagation:



► Package pre-loading:

```
[1]: import numpy as np
```

► Data pre-loading:

```
[2]: input_data = np.array([3, 5])

weights = {
    'node_0_0': np.array([2, 4]),
    'node_0_1': np.array([4, -5]),
    'node_1_0': np.array([-1, 2]),
    'node_1_1': np.array([1, 2]),
    'output': np.array([2, 7])
}
```

► Code pre-loading:

```
[3]: def relu(input):
      output = max(0, input)
      return (output)
```

► Multi-layer neural networks practice:

```
[4]: def predict_with_network(input_data):
      # Calculate node 0 in the first hidden layer
      node_0_0_input = (input_data * weights['node_0_0']).sum()
      node_0_0_output = relu(node_0_0_input)

      # Calculate node 1 in the first hidden layer
      node_0_1_input = (input_data * weights['node_0_1']).sum()
      node_0_1_output = relu(node_0_1_input)

      # Put node values into array: hidden_0_outputs
      hidden_0_outputs = np.array([node_0_0_output, node_0_1_output])

      # Calculate node 0 in the second hidden layer
      node_1_0_input = (hidden_0_outputs * weights['node_1_0']).sum()
      node_1_0_output = relu(node_1_0_input)

      # Calculate node 1 in the second hidden layer
      node_1_1_input = (hidden_0_outputs * weights['node_1_1']).sum()
      node_1_1_output = relu(node_1_1_input)

      # Put node values into array: hidden_1_outputs
      hidden_1_outputs = np.array([node_1_0_output, node_1_1_output])

      # Calculate model output: model_output
```

```
model_output = (hidden_1_outputs * weights['output']).sum()

# Return model_output
return (model_output)

output = predict_with_network(input_data)
print(output)
```

182

**6. Practice question for learned representations:**

- How are the weights that determine the features/interactions in Neural Networks created?
  - ☐ A user chooses them when creating the model.
  - ☒ The model training process sets them to optimize predictive accuracy.
  - ☐ The weights are random numbers.

**7. Practice question for levels of representation:**

- Which layers of a model capture more complex or “higher level” interactions?
  - ☐ The first layers capture the most complex interactions.
  - ☒ The last layers capture the most complex interactions.
  - ☐ All layers capture interactions of similar complexity.