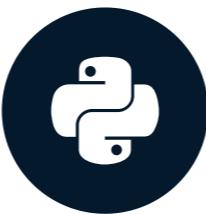


# Introduction to deep learning

INTRODUCTION TO DEEP LEARNING IN PYTHON



Dan Becker

Data Scientist and contributor to Keras  
and TensorFlow libraries

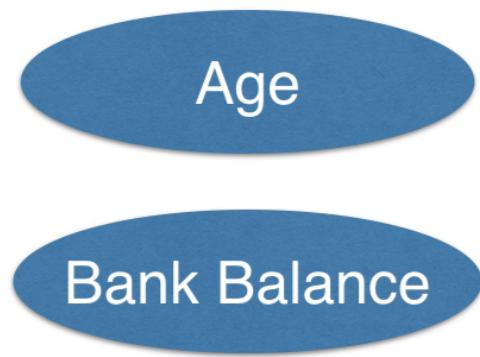
# Imagine you work for a bank

- You need to predict how many transactions each customer will make next year

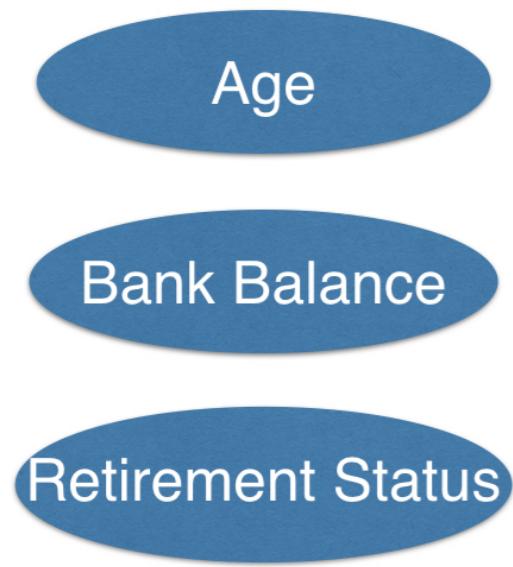
# Example as seen by linear regression



# Example as seen by linear regression



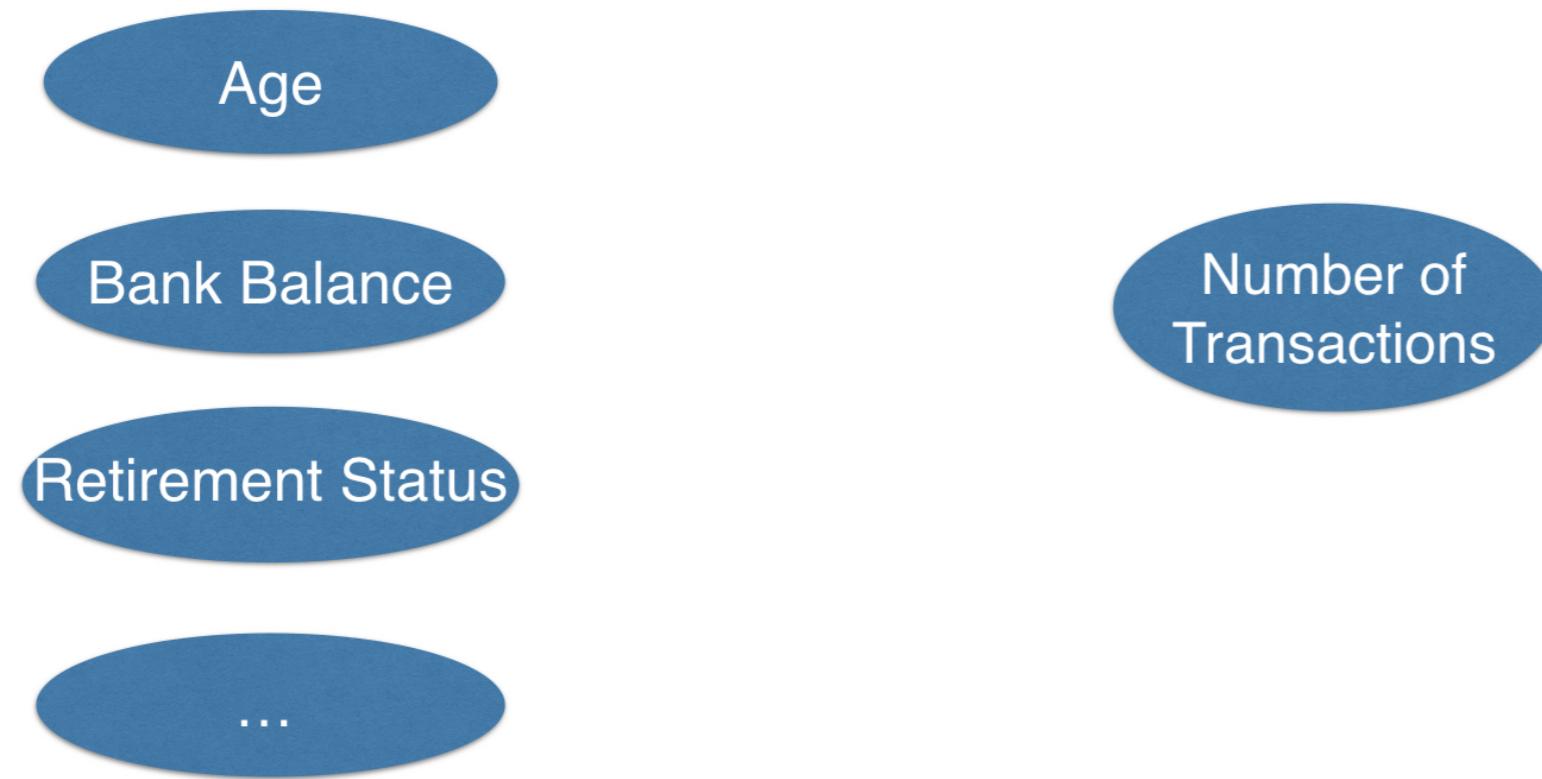
# Example as seen by linear regression



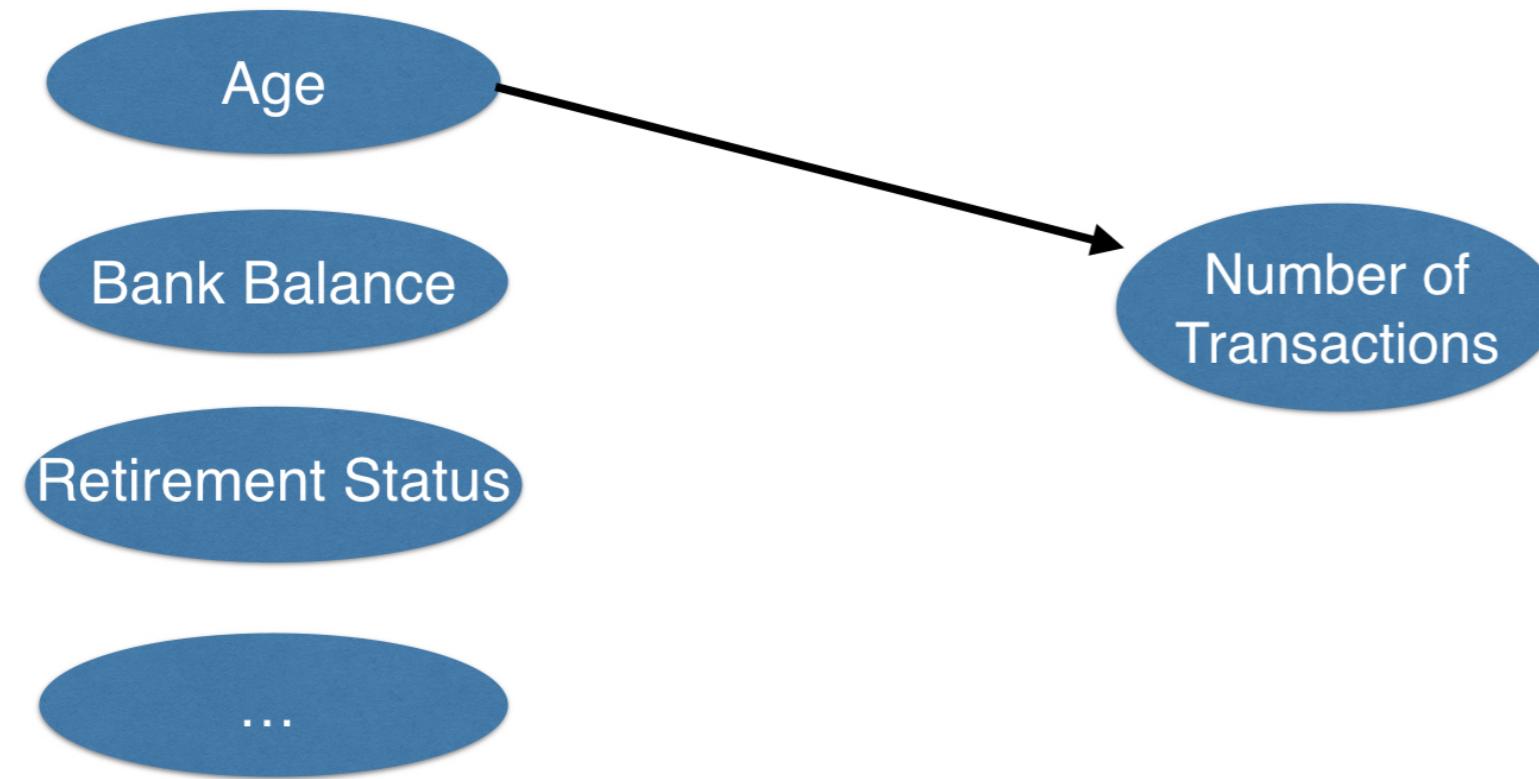
# Example as seen by linear regression



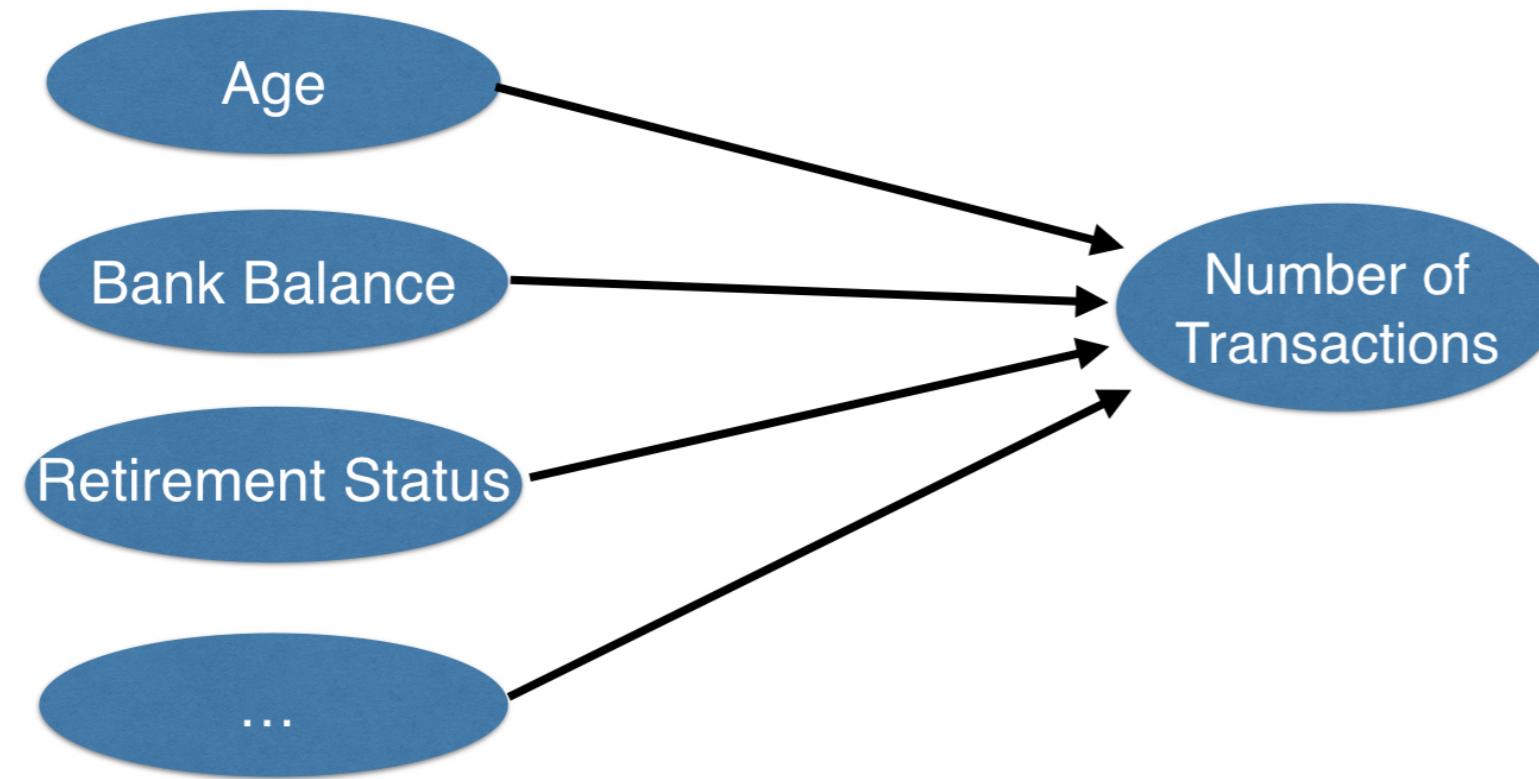
# Example as seen by linear regression



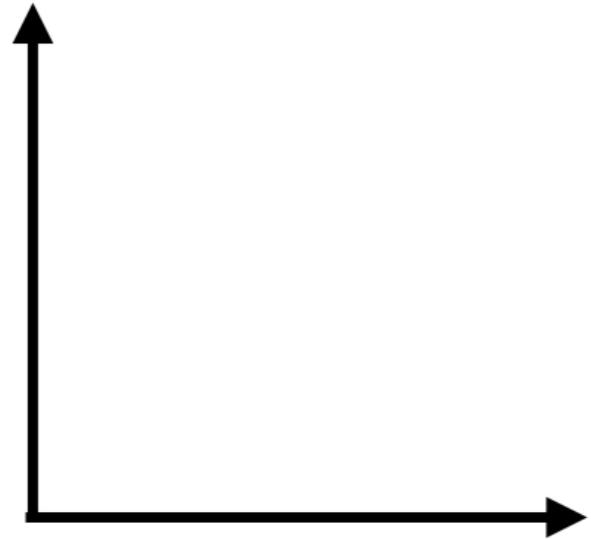
# Example as seen by linear regression



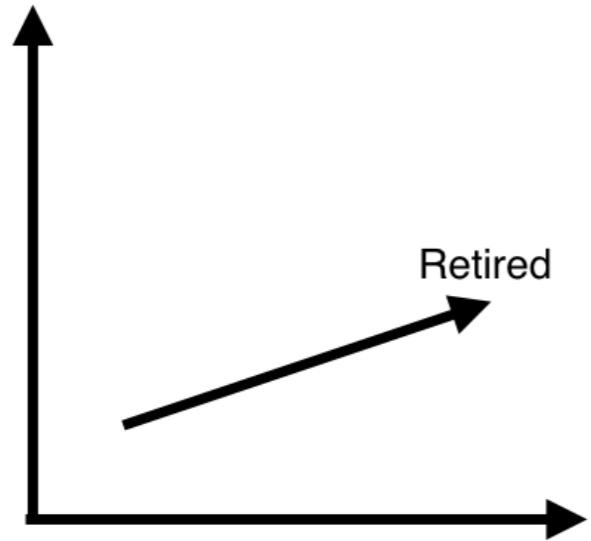
# Example as seen by linear regression



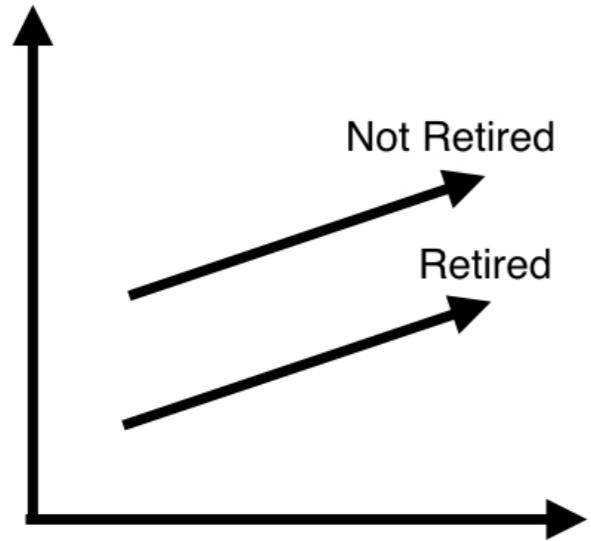
# Example as seen by linear regression



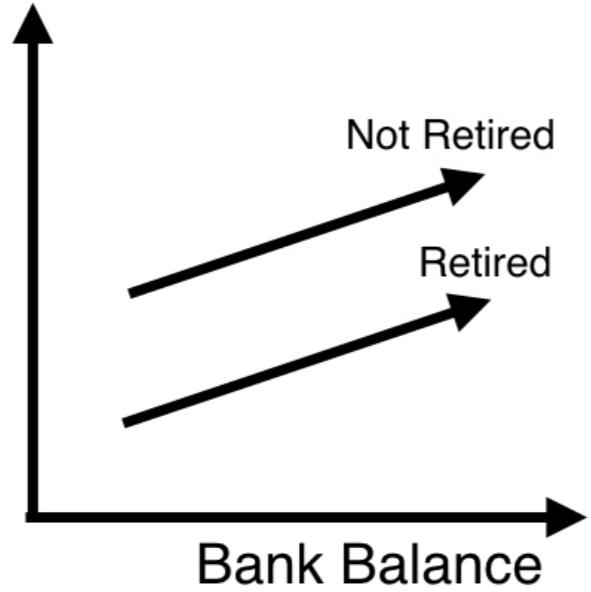
# Example as seen by linear regression



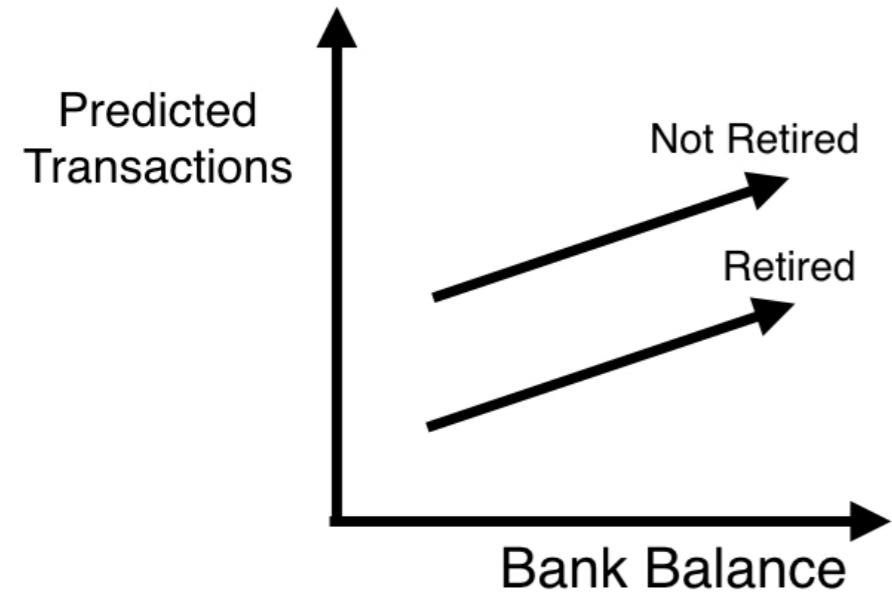
# Example as seen by linear regression



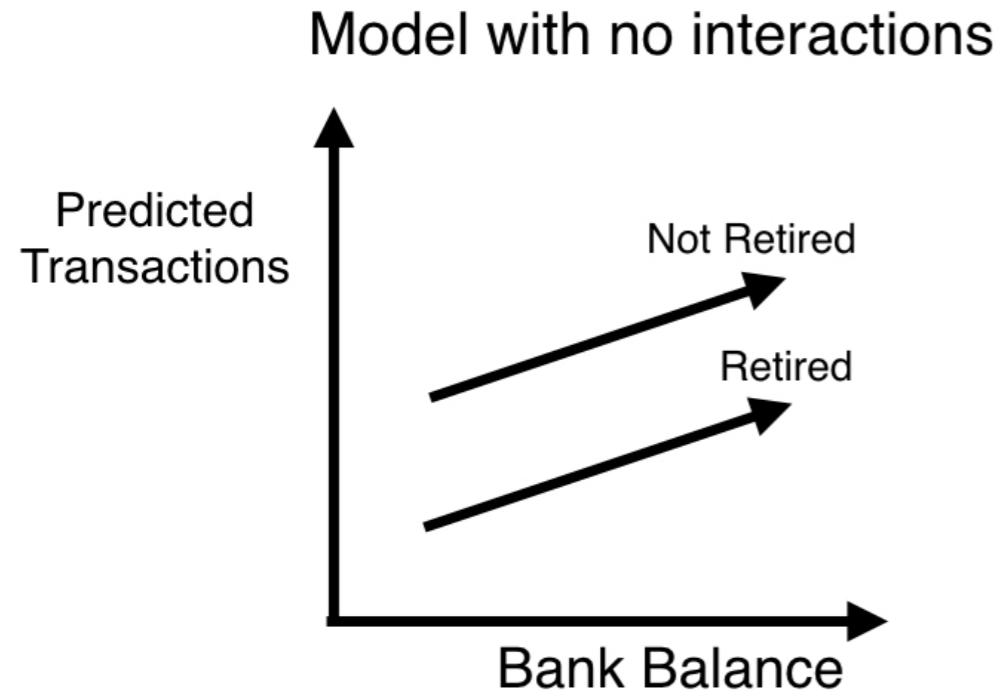
# Example as seen by linear regression



# Example as seen by linear regression

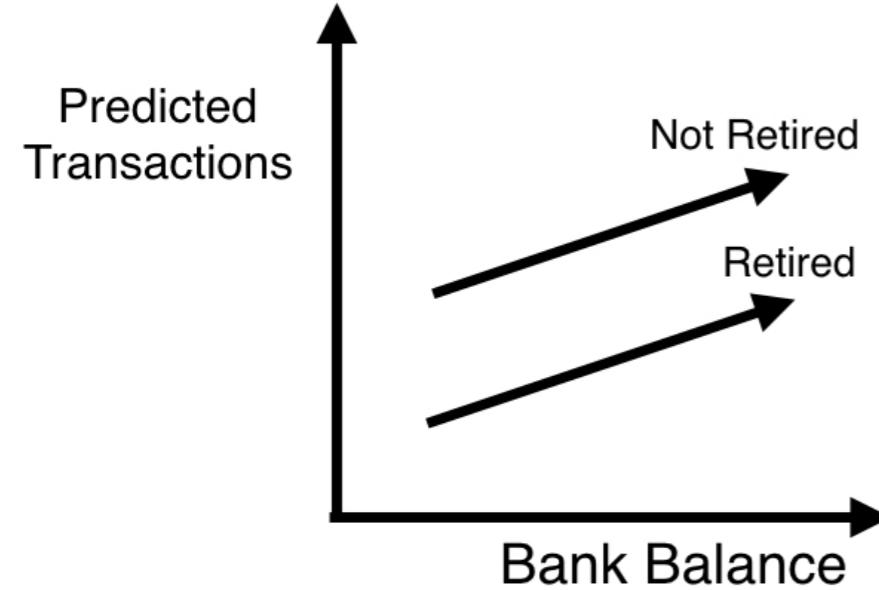


# Example as seen by linear regression

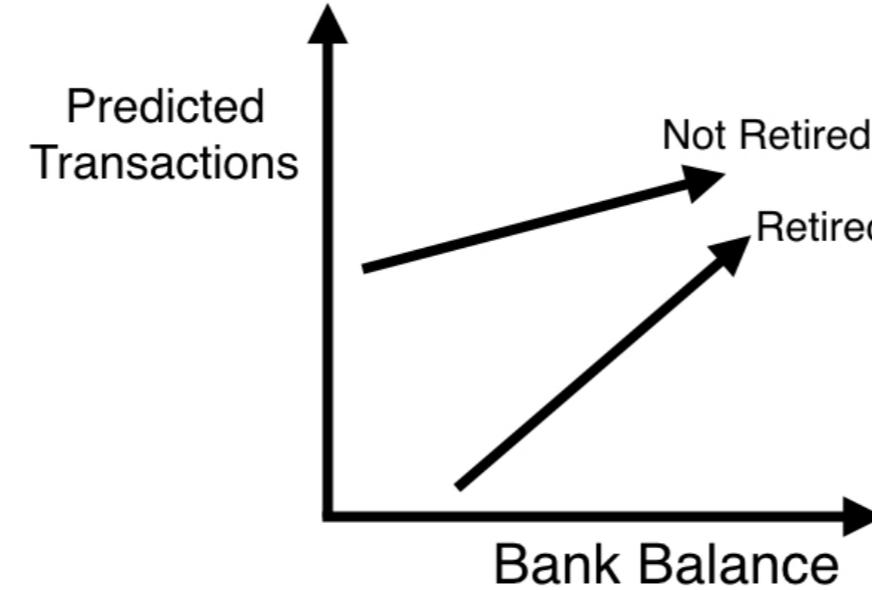


# Example as seen by linear regression

Model with no interactions



Model with interactions



# Interactions

- Neural networks account for interactions really well
- Deep learning uses especially powerful neural networks
  - Text
  - Images
  - Videos
  - Audio
  - Source code

# Course structure

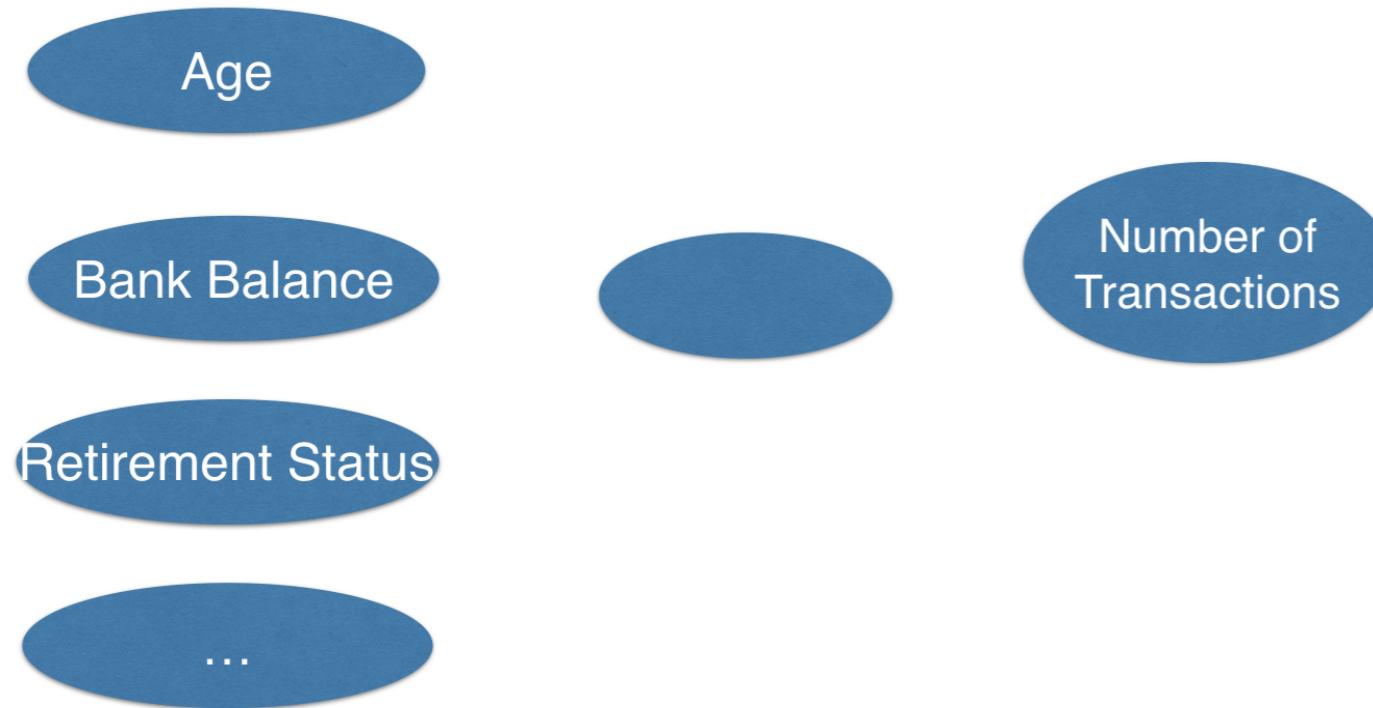
- First two chapters focus on conceptual knowledge
  - Debug and tune deep learning models on conventional prediction problems
  - Lay the foundation for progressing towards modern applications
- This will pay off in the third and fourth chapters

# Build and tune deep learning models using keras

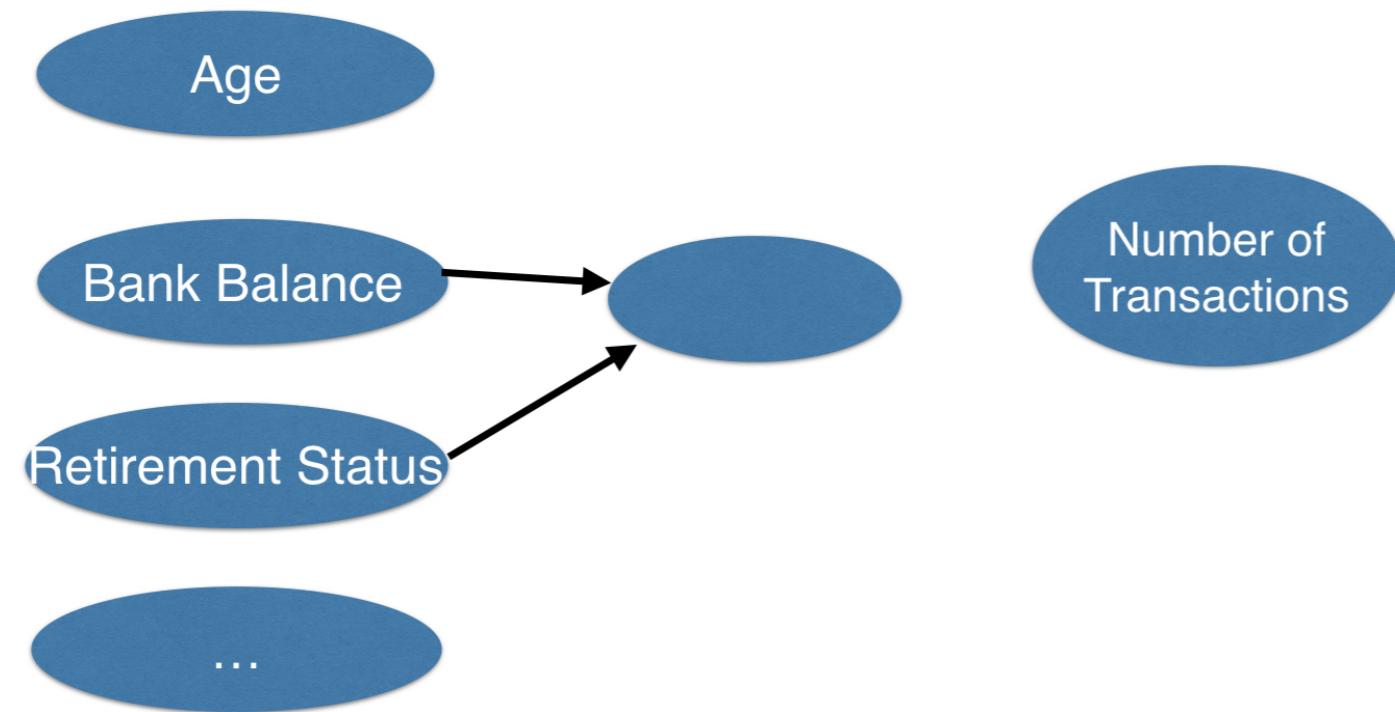
```
import numpy as np
from keras.layers import Dense
from keras.models import Sequential
predictors = np.loadtxt('predictors_data.csv', delimiter=',')
n_cols = predictors.shape[1]
model = Sequential()

model.add(Dense(100, activation='relu', input_shape = (n_cols,)))
model.add(Dense(100, activation='relu'))
model.add(Dense(1))
```

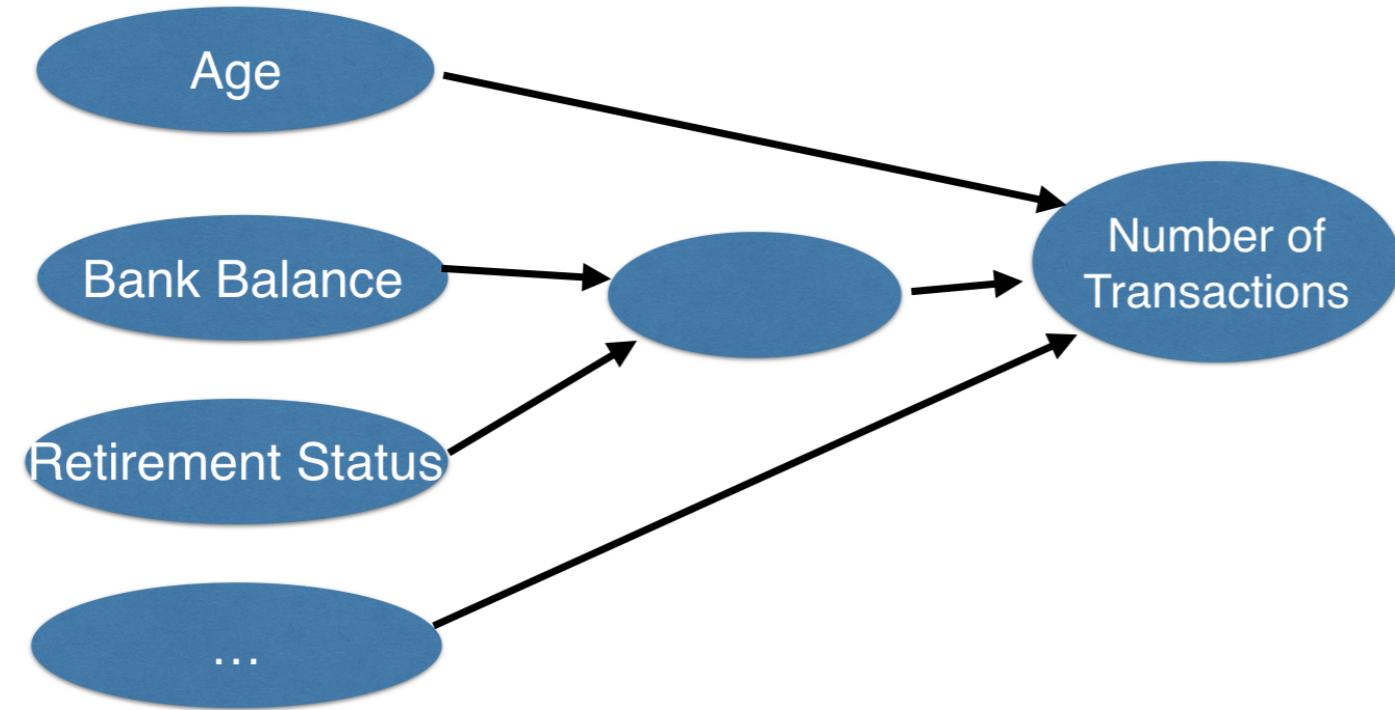
# Deep learning models capture interactions



# Deep learning models capture interactions



# Deep learning models capture interactions



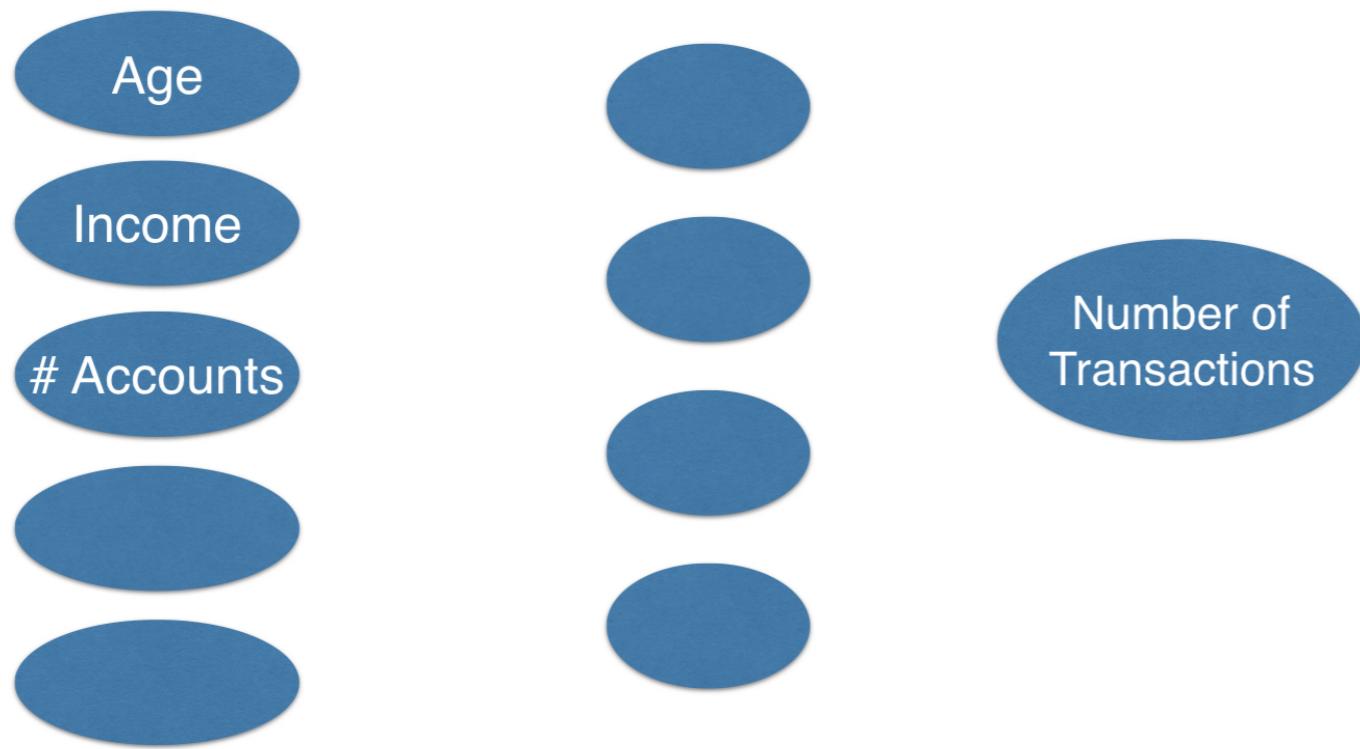
# Interactions in neural network



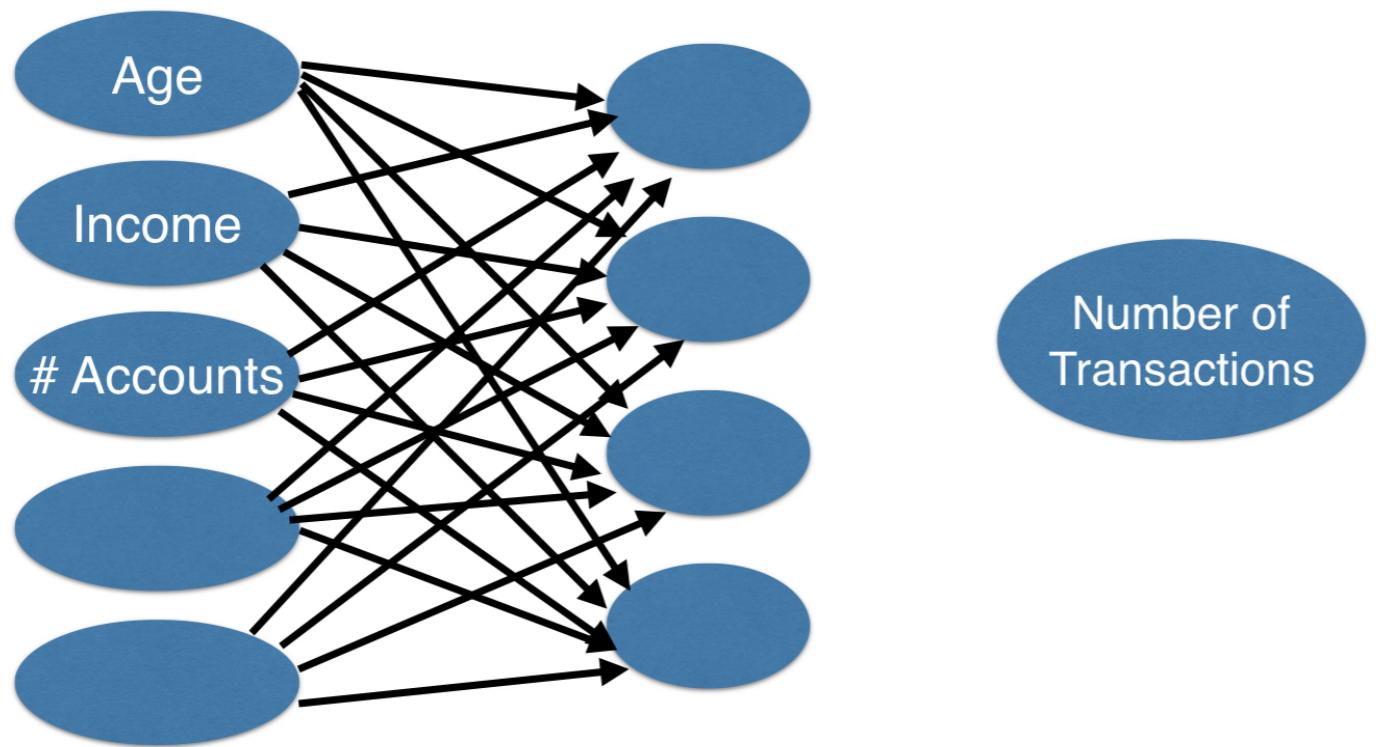
# Interactions in neural network



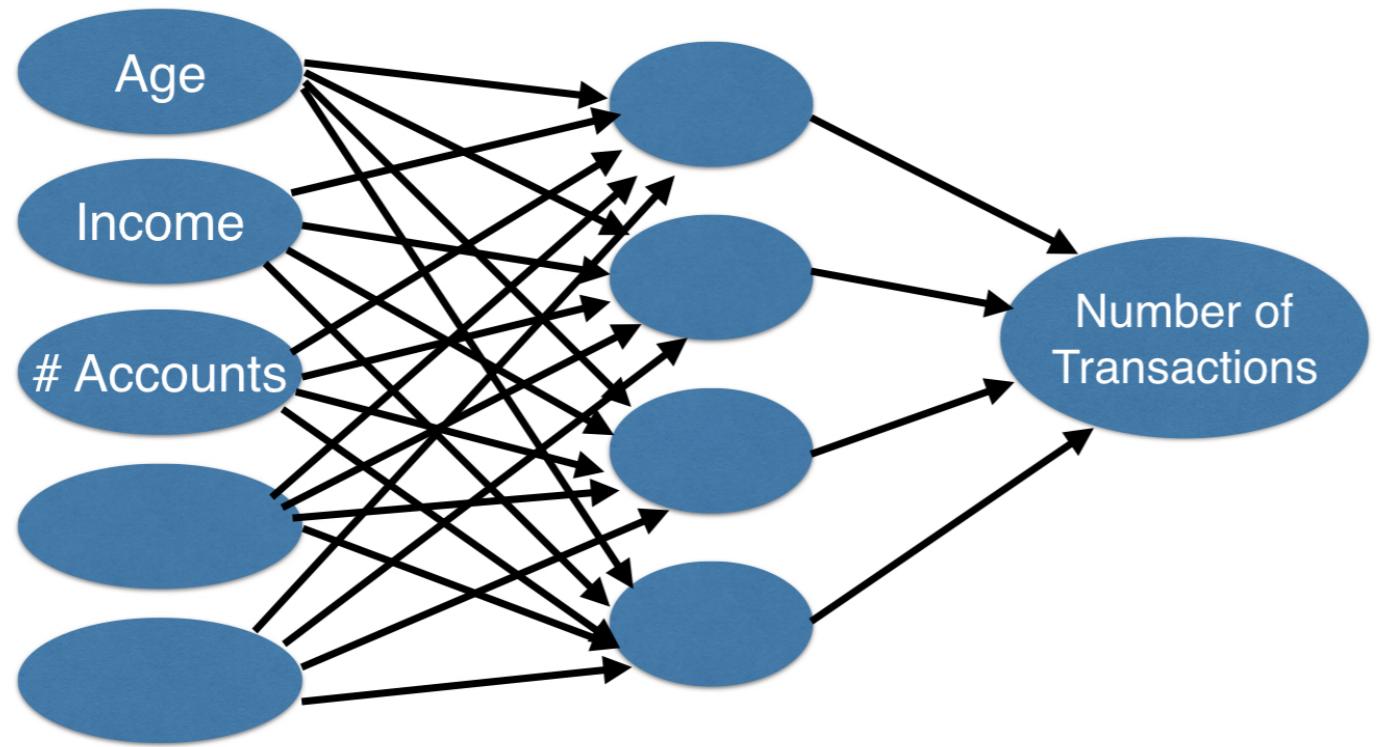
# Interactions in neural network



# Interactions in neural network



# Interactions in neural network



# **Let's practice!**

**INTRODUCTION TO DEEP LEARNING IN PYTHON**

# Forward propagation

INTRODUCTION TO DEEP LEARNING IN PYTHON



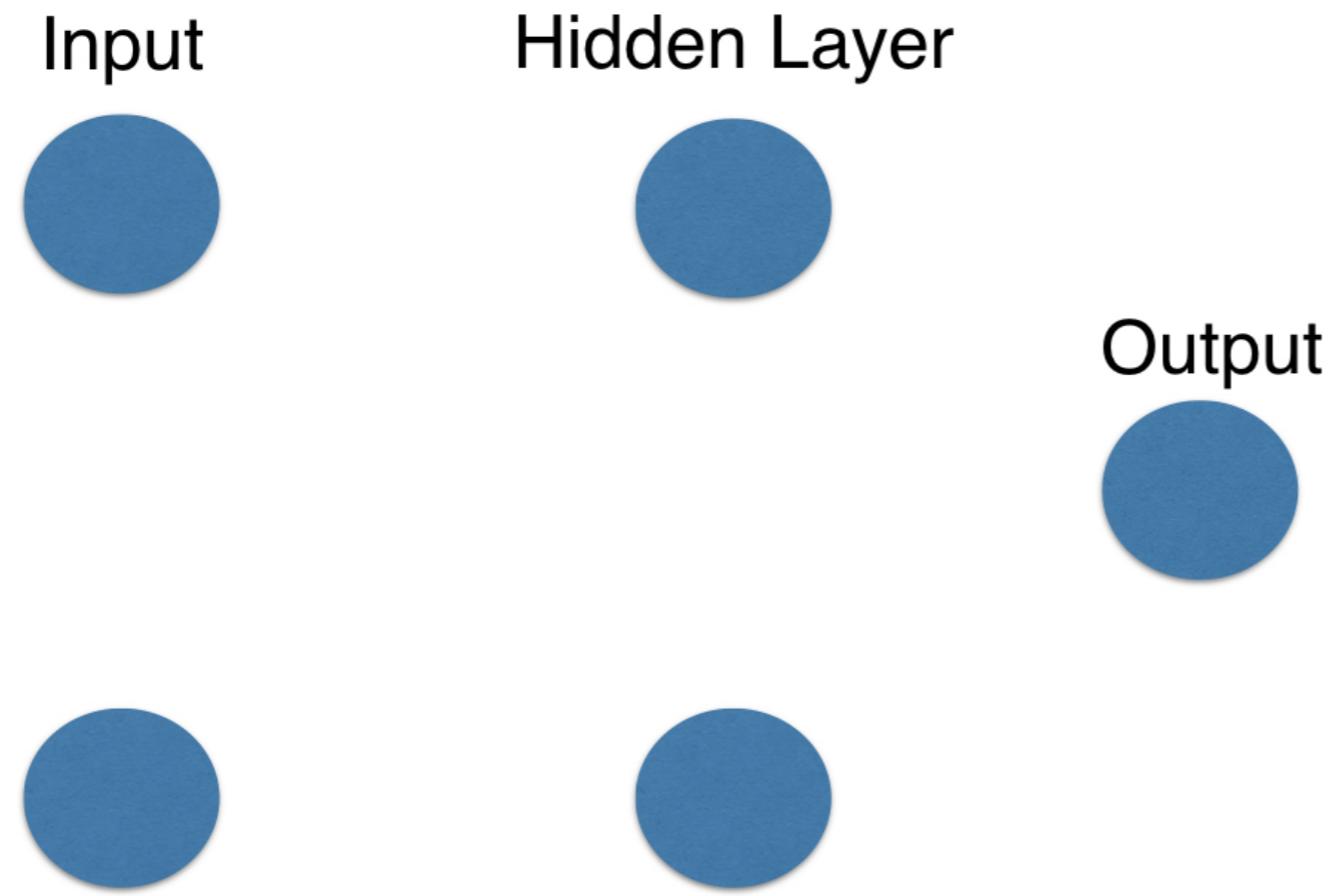
Dan Becker

Data Scientist and contributor to Keras  
and TensorFlow libraries

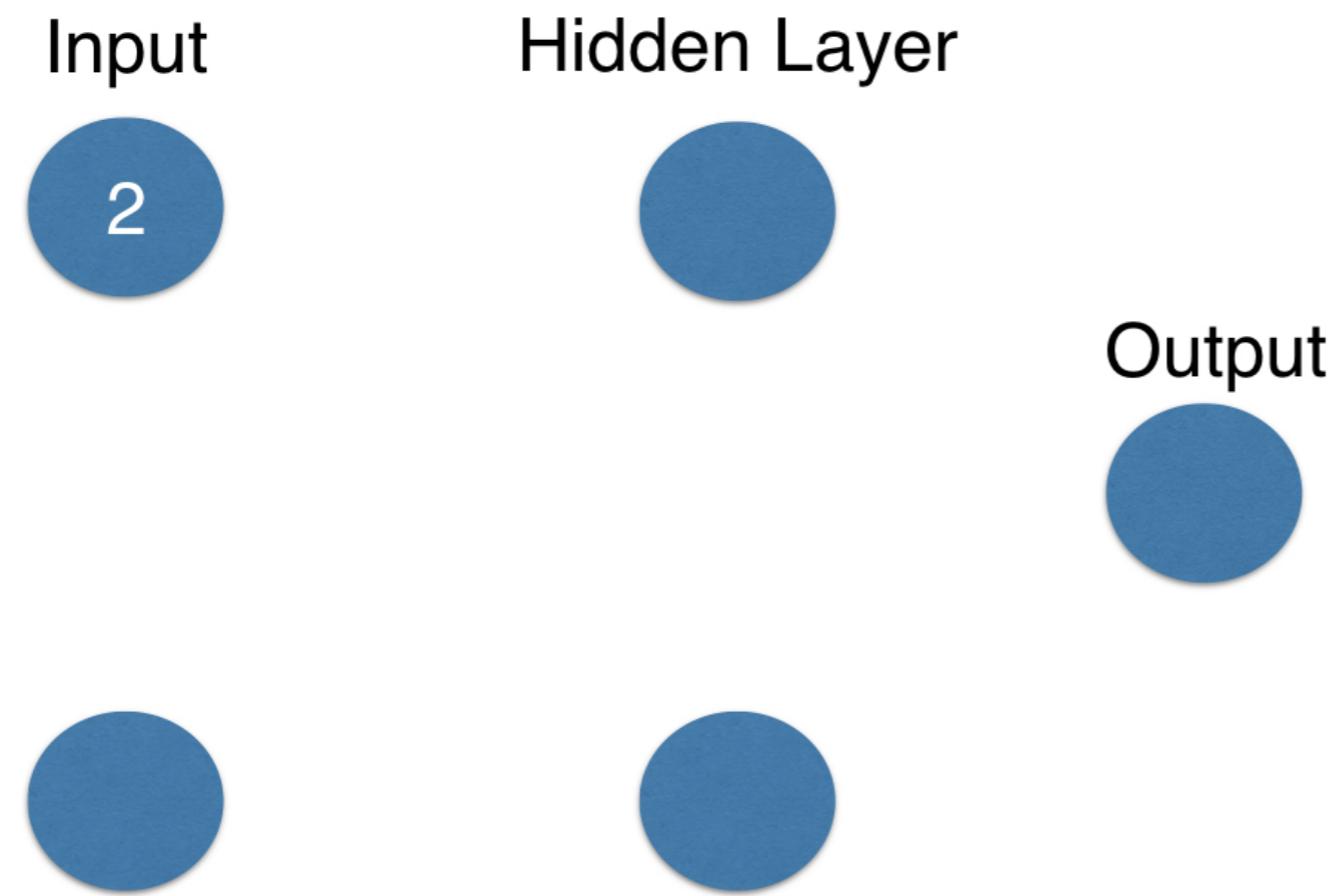
# Bank transactions example

- Make predictions based on:
  - Number of children
  - Number of existing accounts

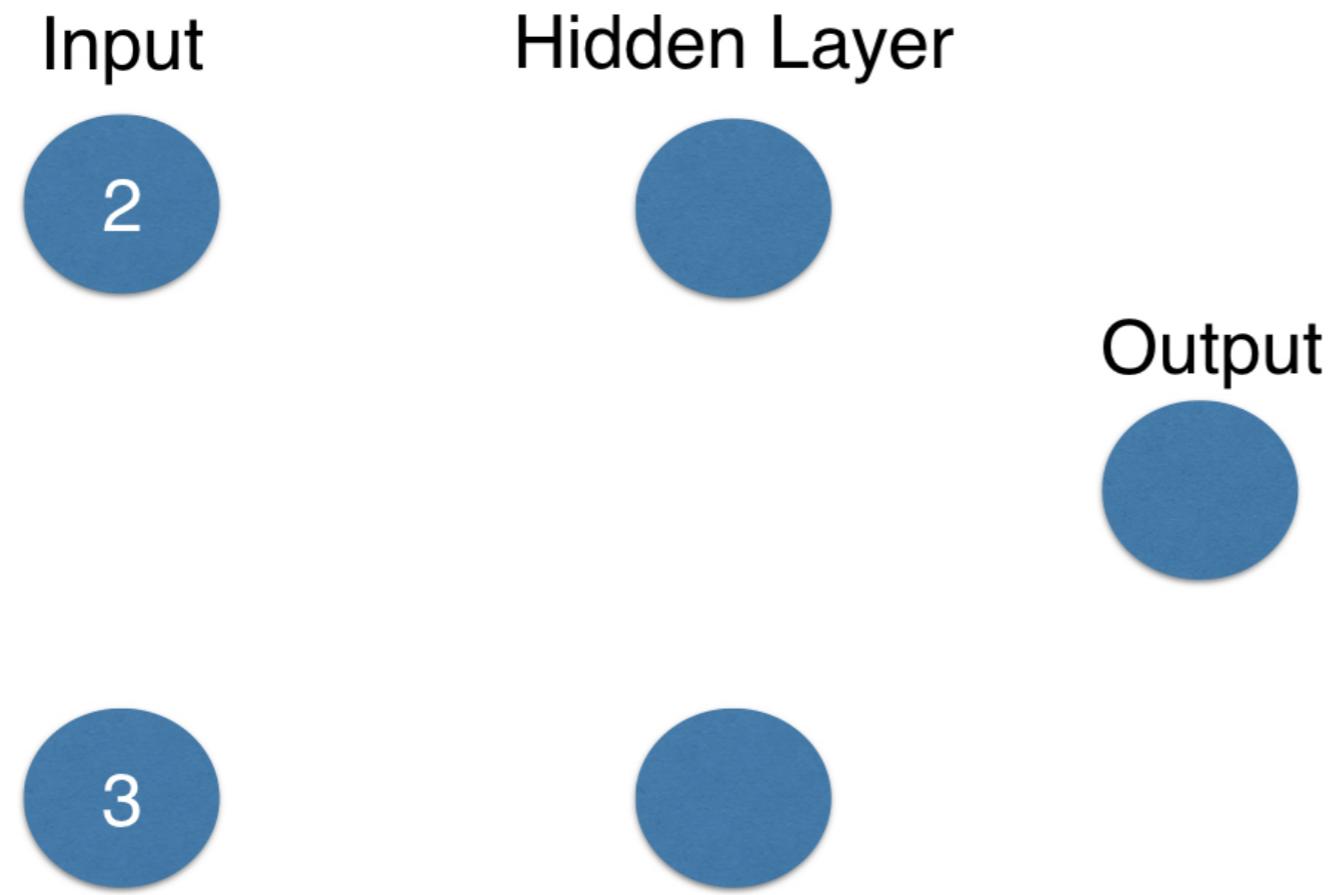
# Forward propagation



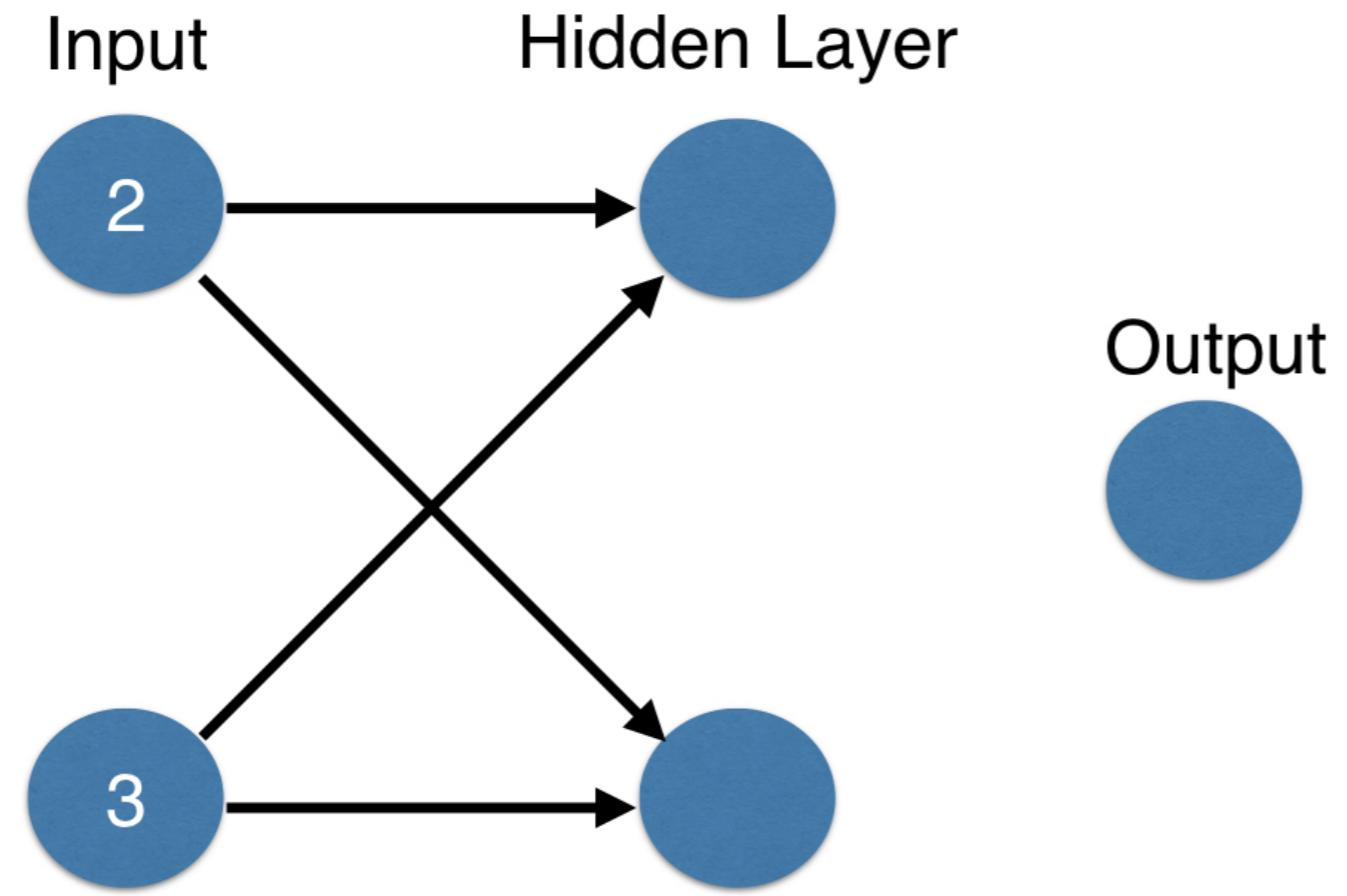
# Forward propagation



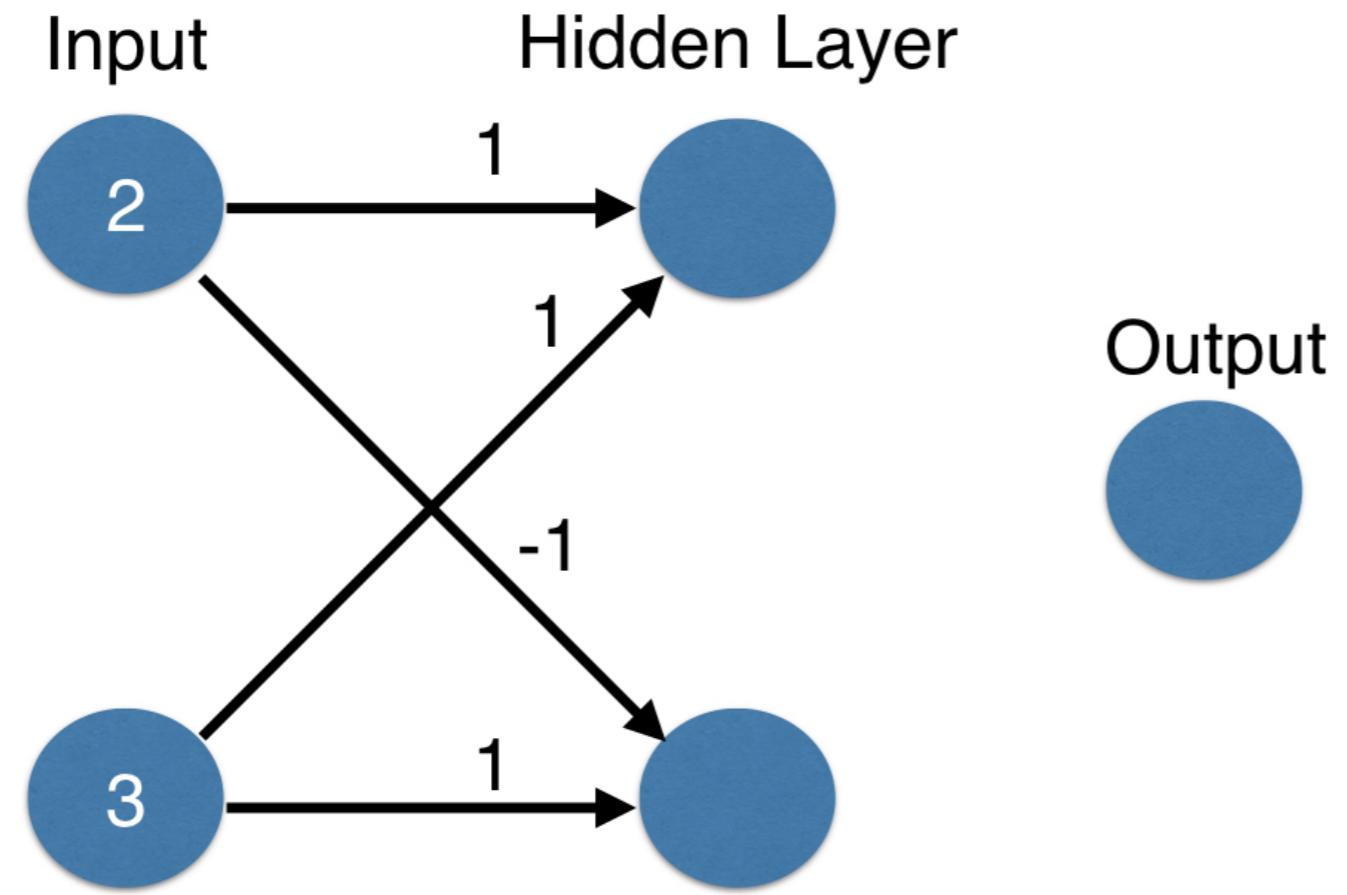
# Forward propagation



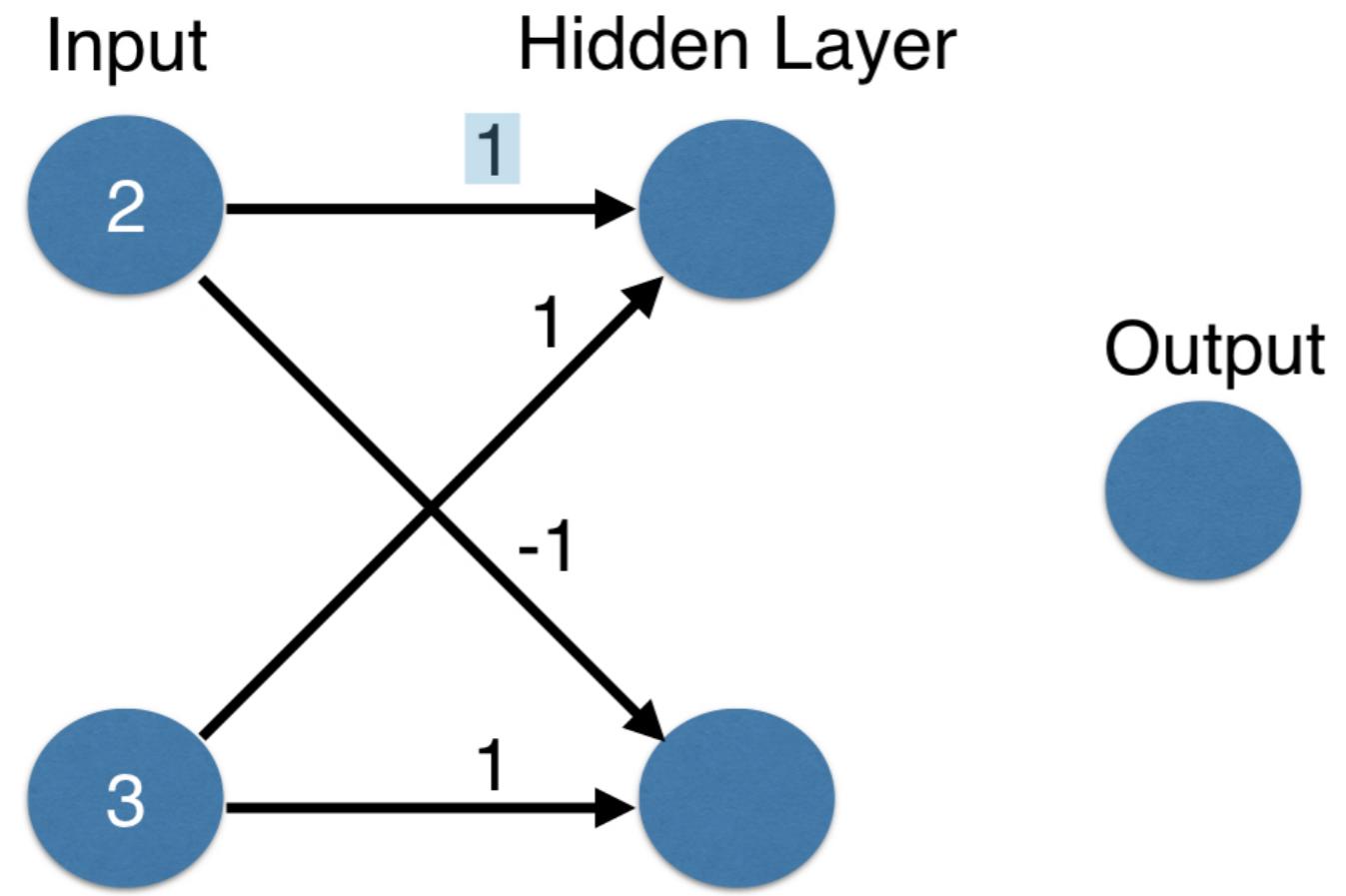
# Forward propagation



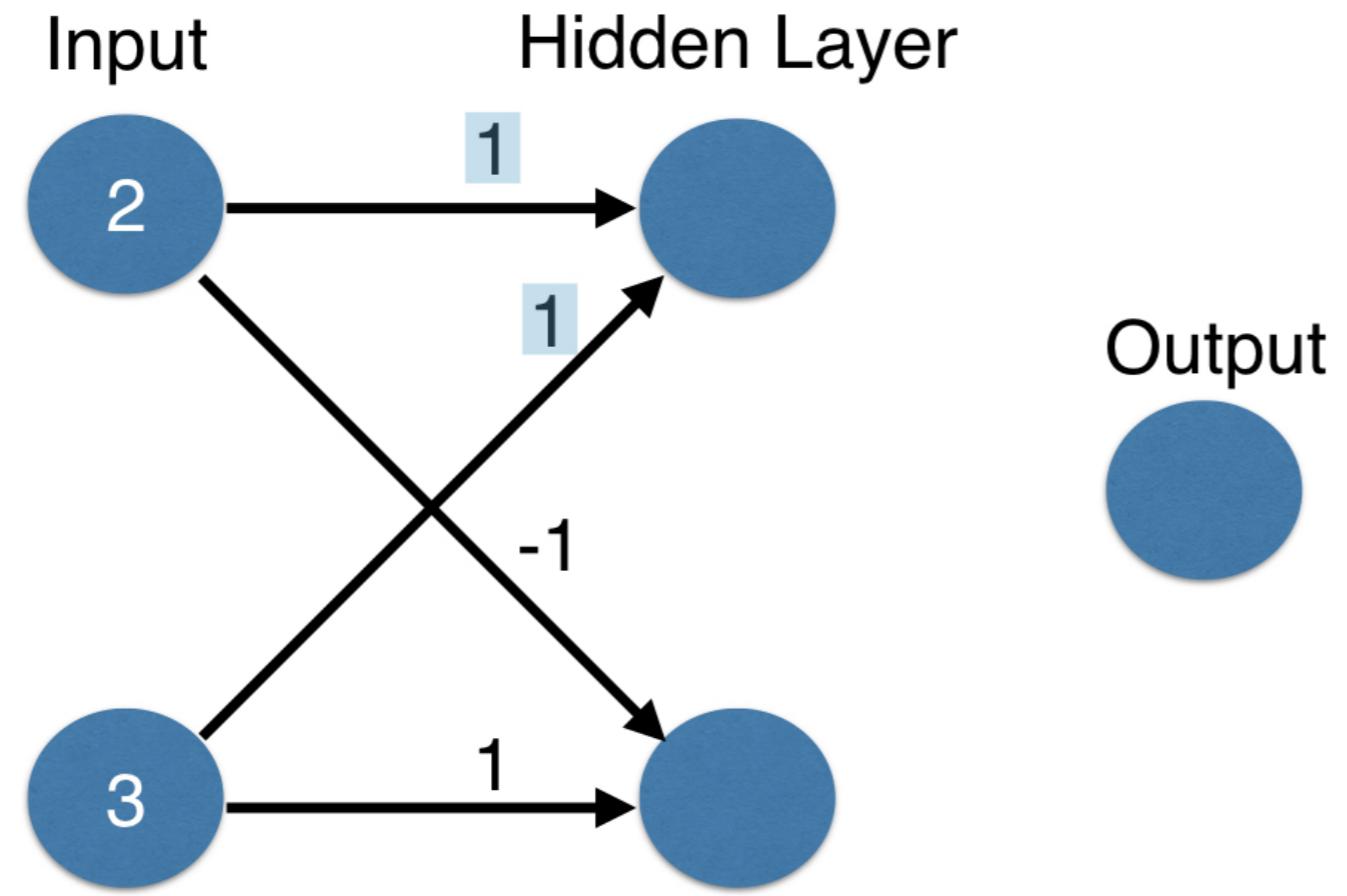
# Forward propagation



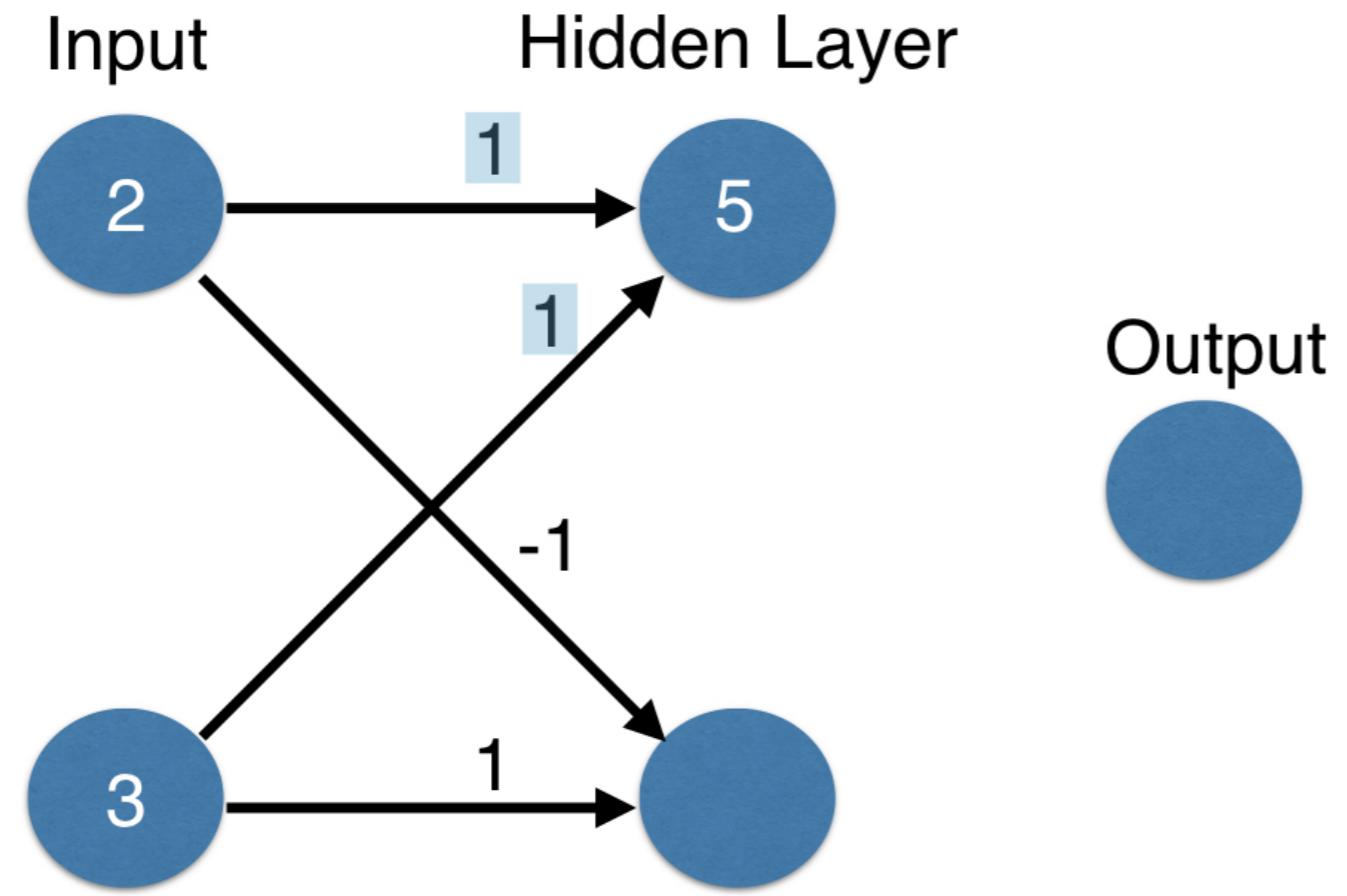
# Forward propagation



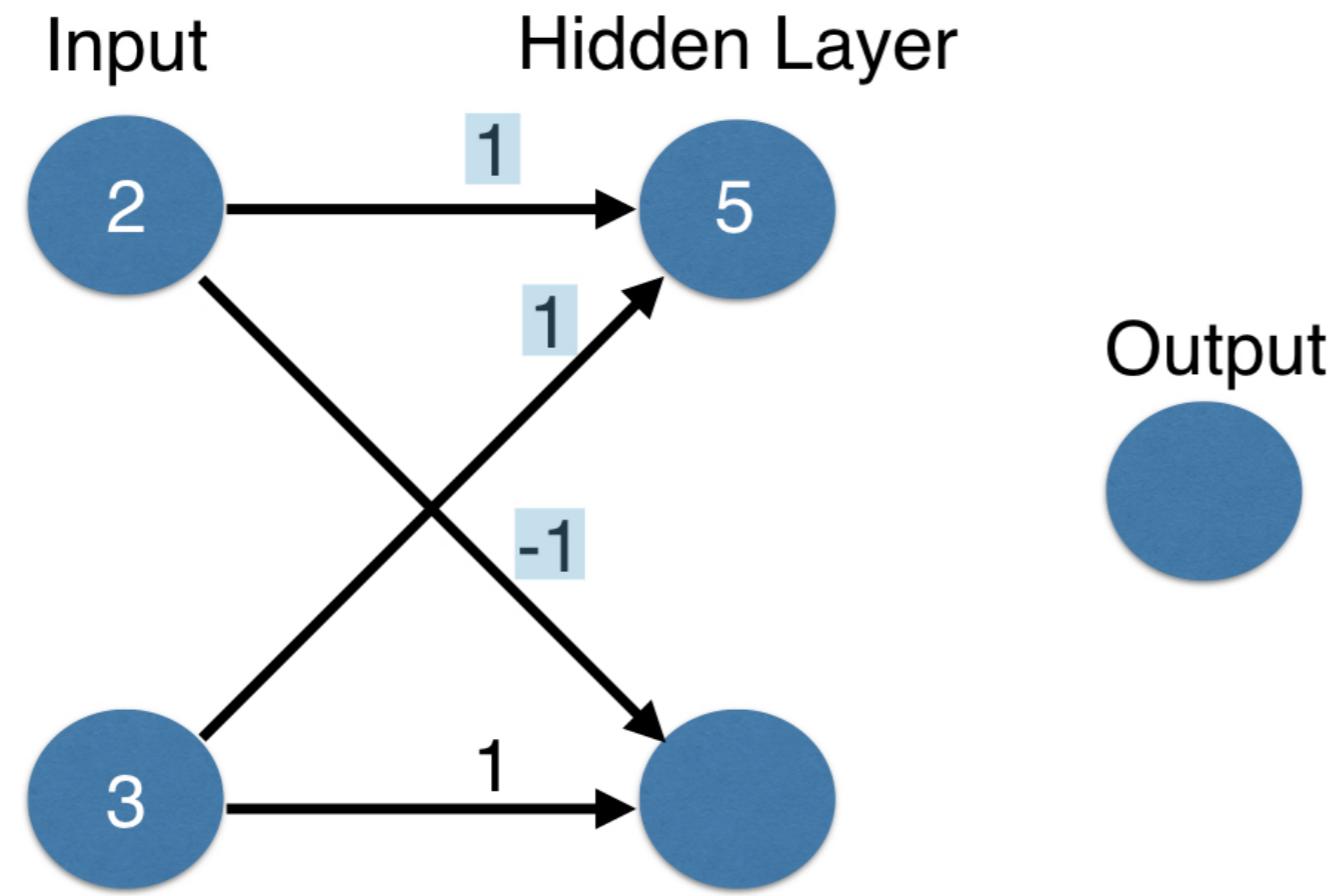
# Forward propagation



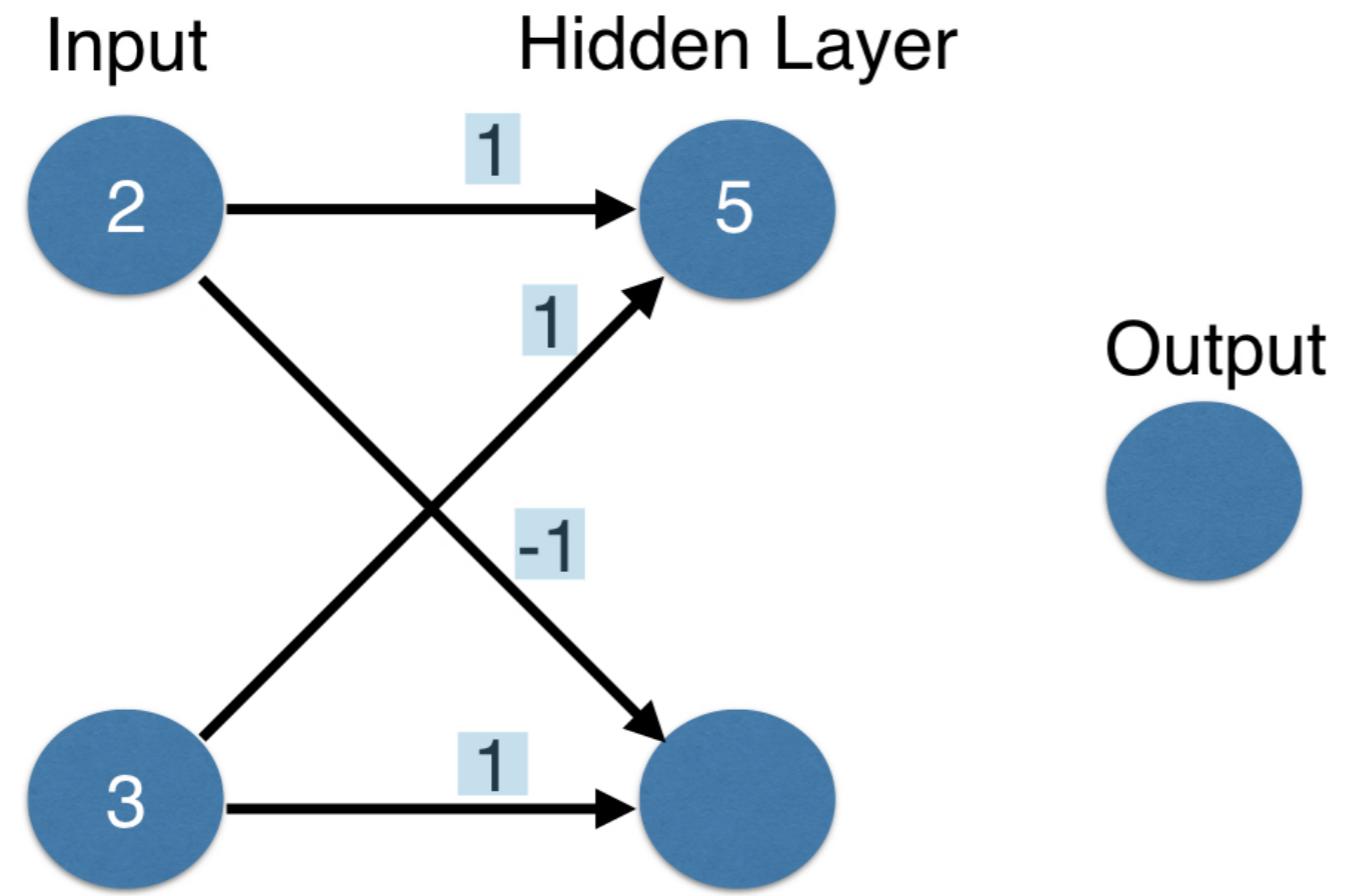
# Forward propagation



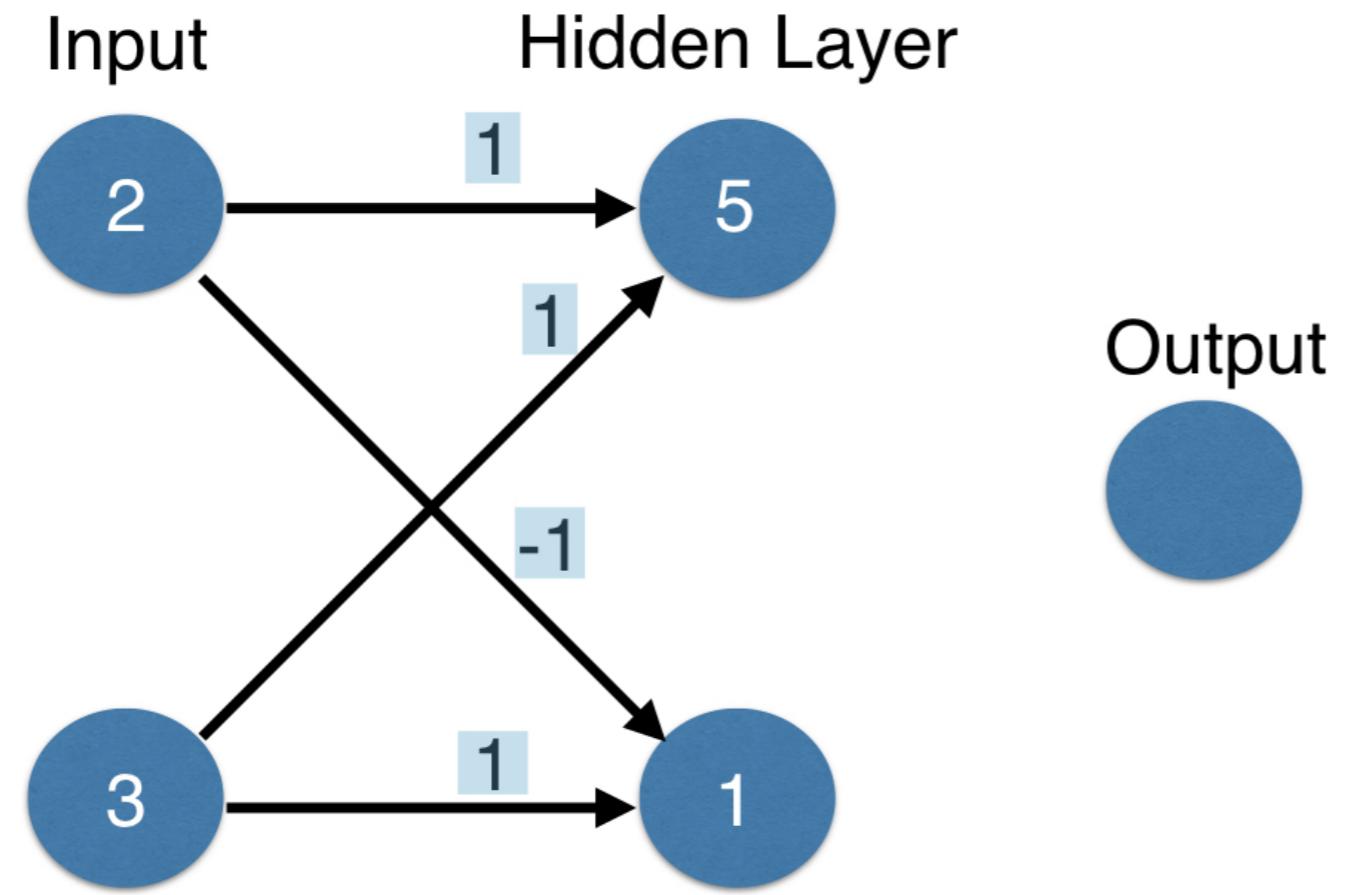
# Forward propagation



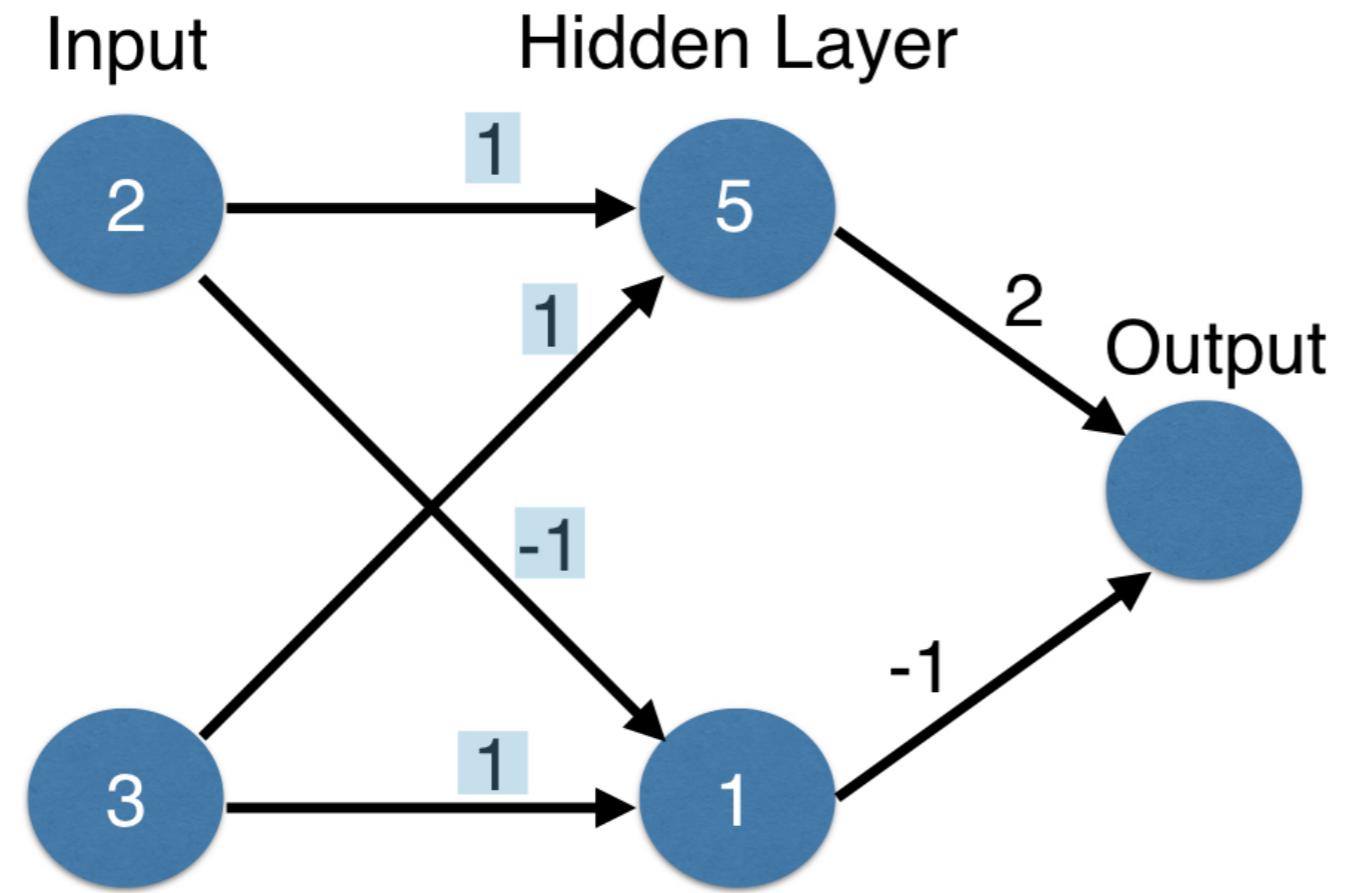
# Forward propagation



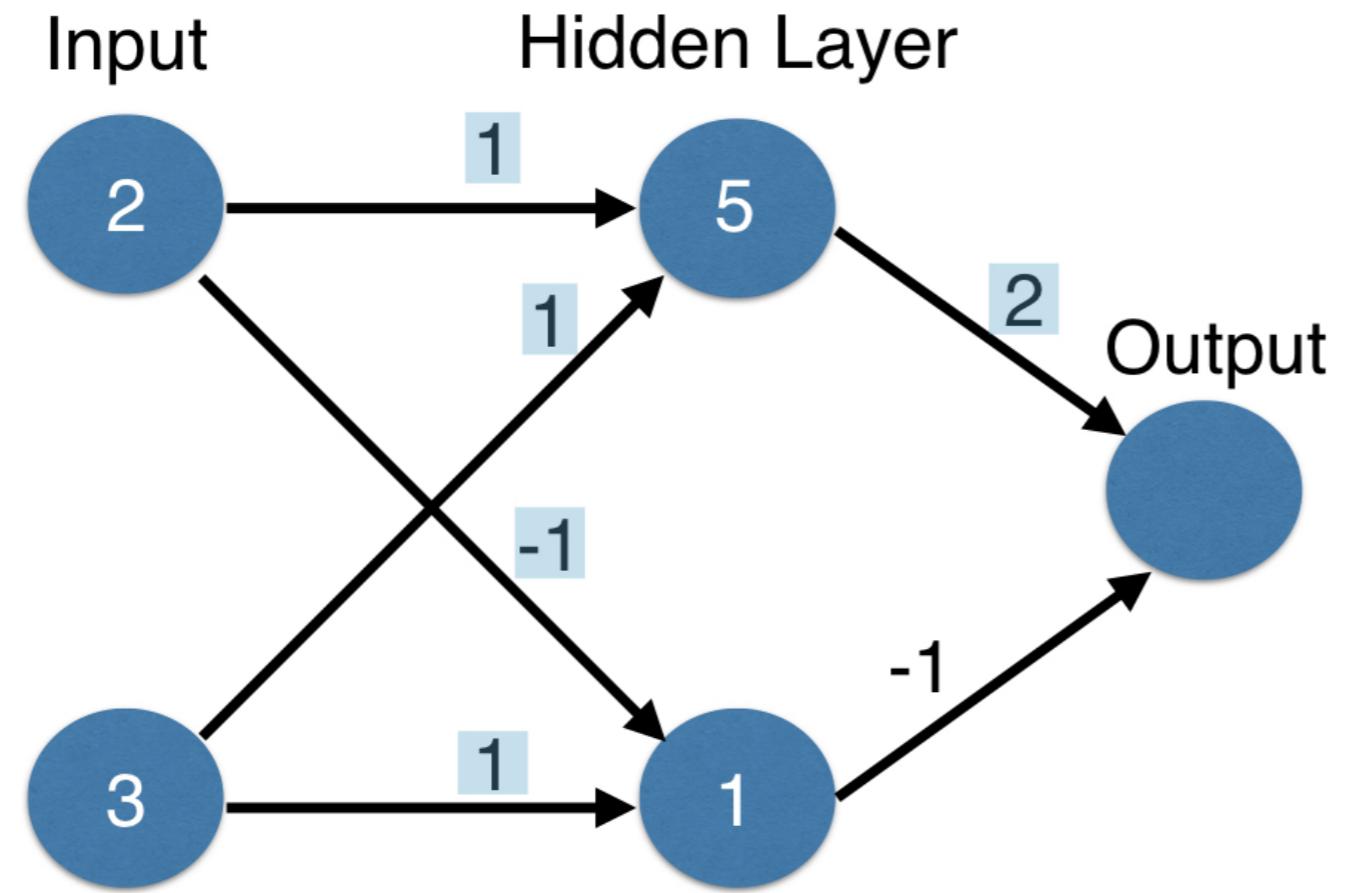
# Forward propagation



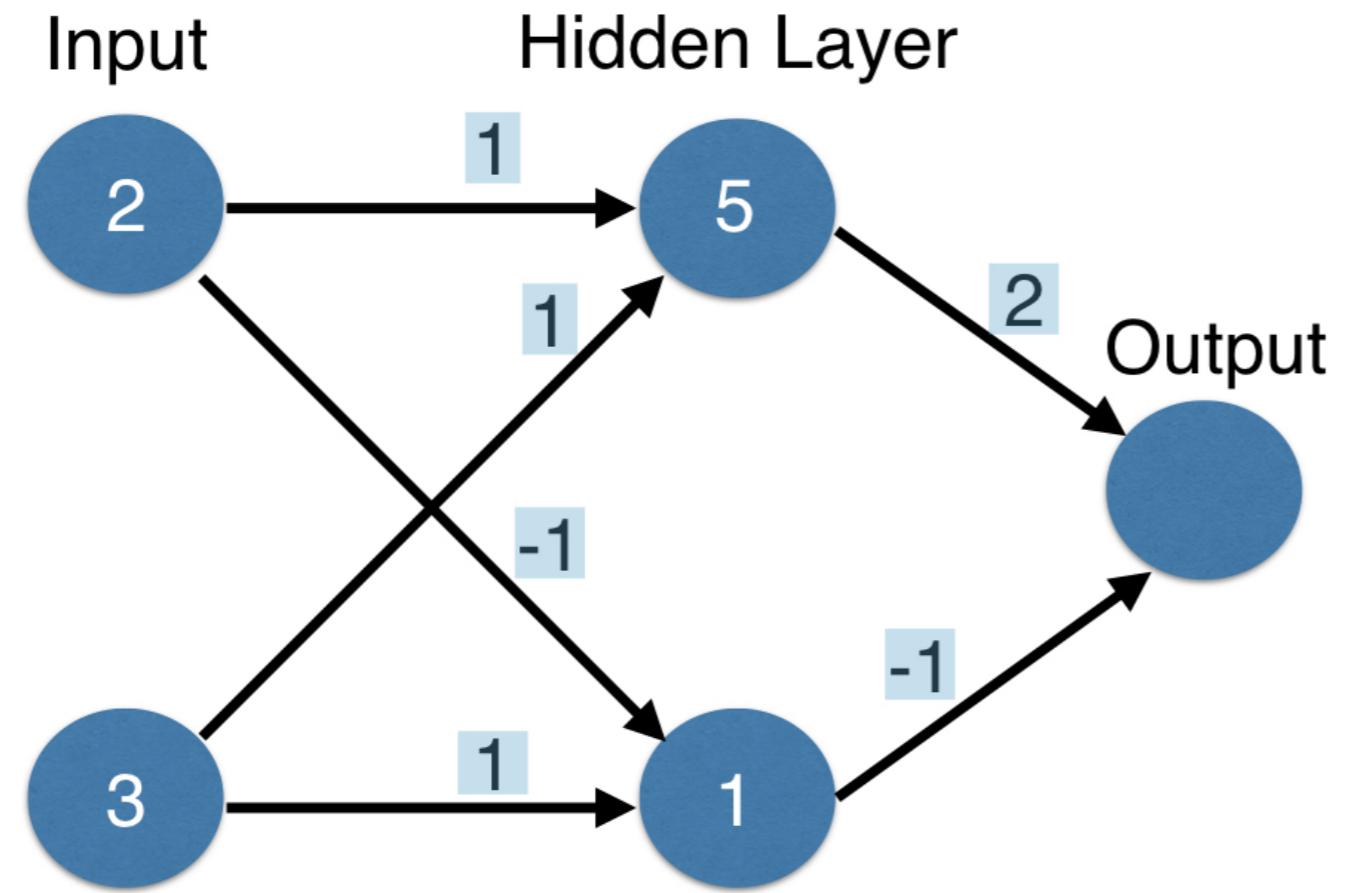
# Forward propagation



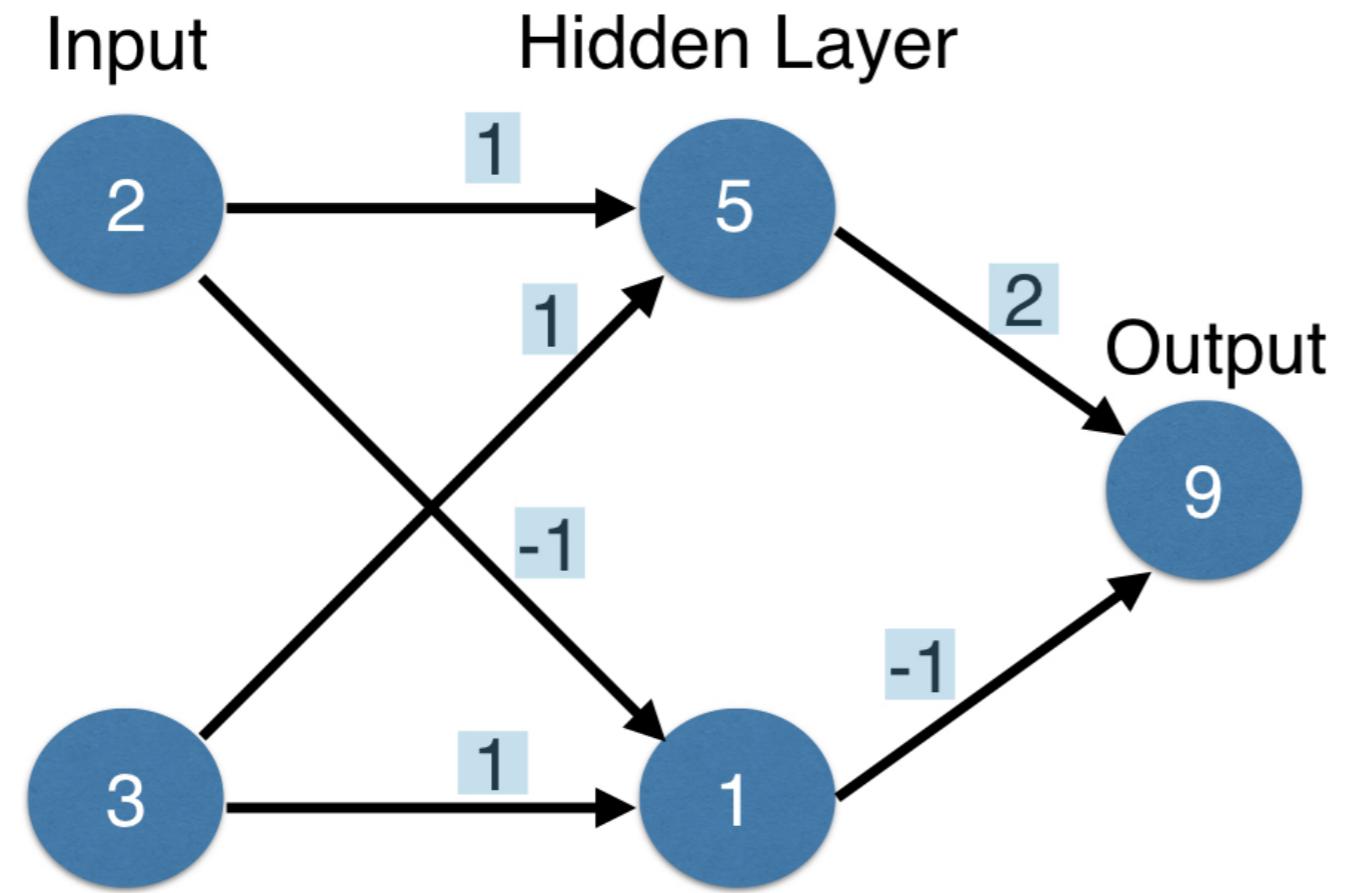
# Forward propagation



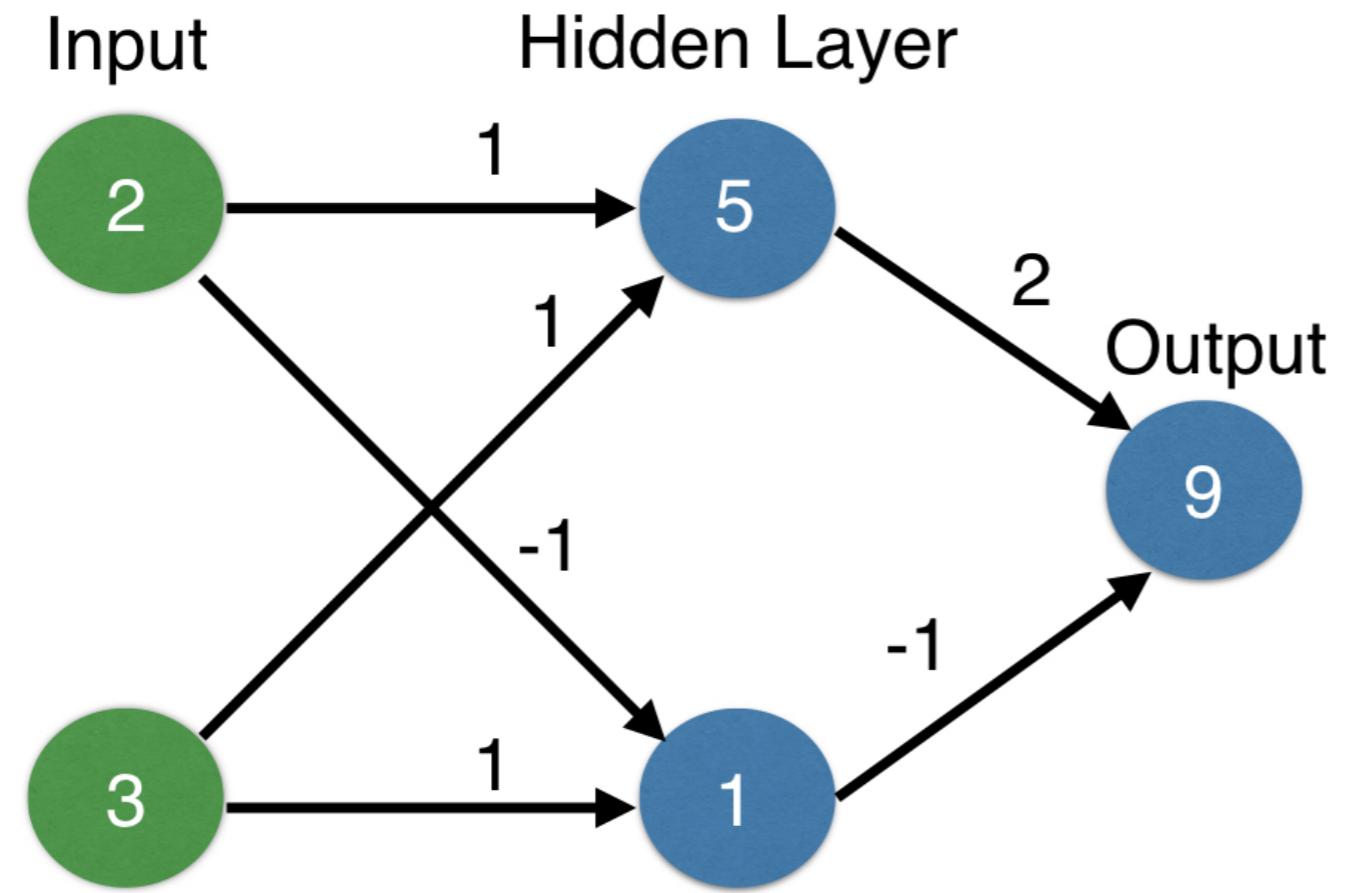
# Forward propagation



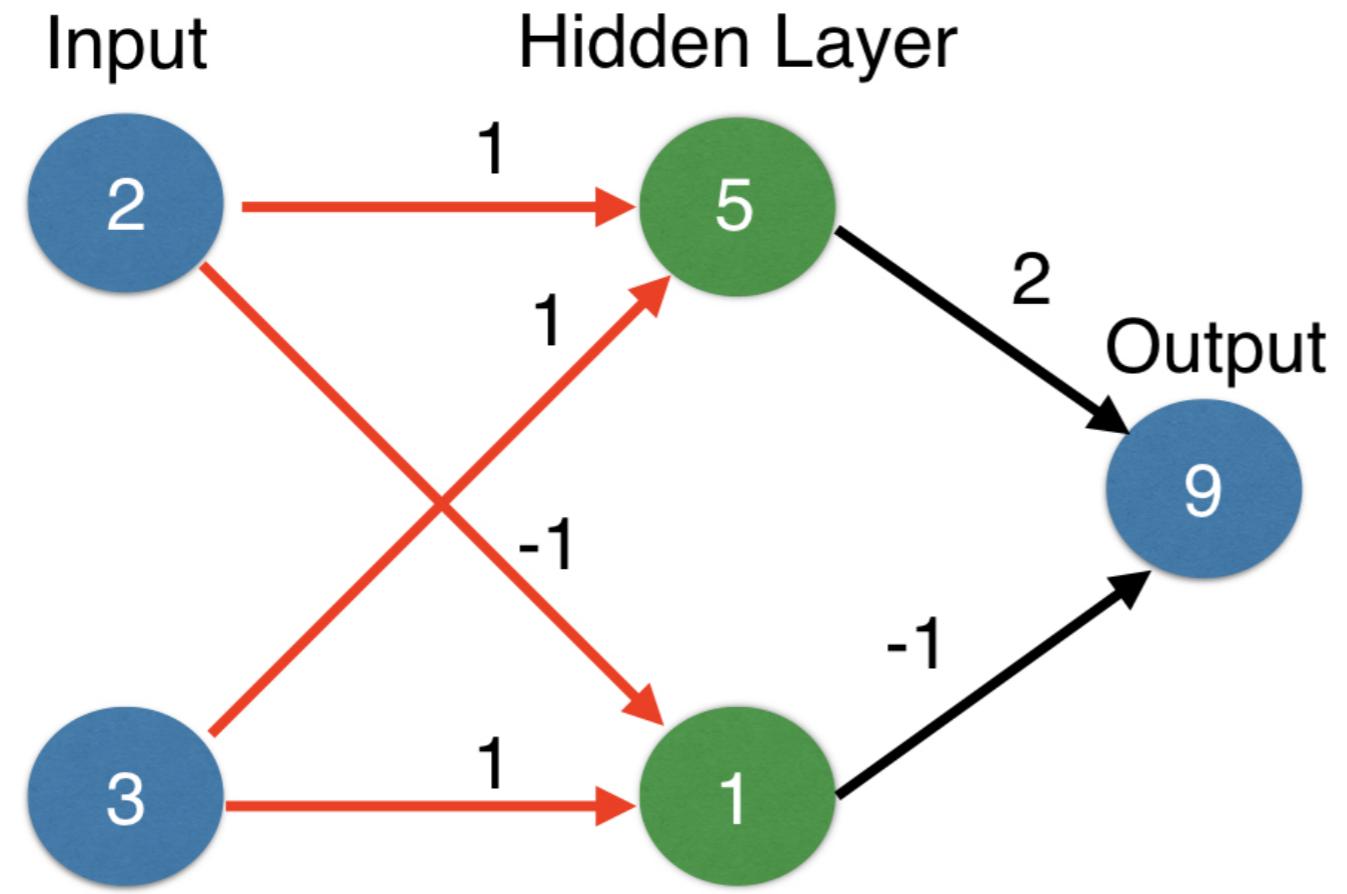
# Forward propagation



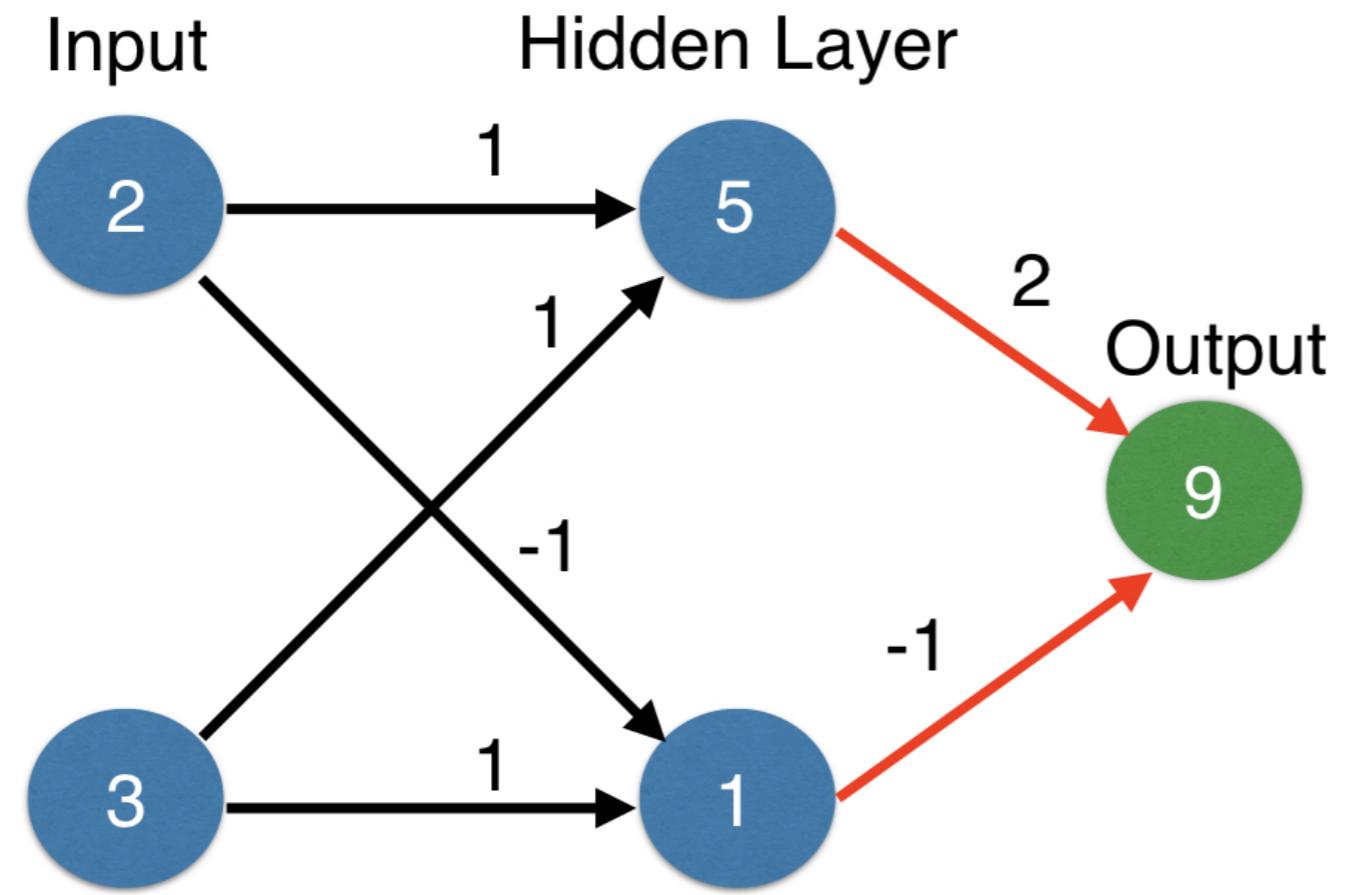
# Forward propagation



# Forward propagation



# Forward propagation



# Forward propagation

- Multiply - add process
- Dot product
- Forward propagation for one data point at a time
- Output is the prediction for that data point

# Forward propagation code

```
import numpy as np

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

weights = {'node_0': np.array([1, 1]),
           'node_1': np.array([-1, 1]),
           'output': np.array([2, -1])}

node_0_value = (input_data * weights['node_0']).sum()
node_1_value = (input_data * weights['node_1']).sum()
```

# Forward propagation code

```
hidden_layer_values = np.array([node_0_value, node_1_value])  
print(hidden_layer_values)
```

```
[5, 1]
```

```
output = (hidden_layer_values * weights['output']).sum()  
print(output)
```

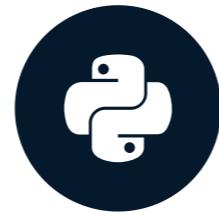
```
9
```

# **Let's practice!**

**INTRODUCTION TO DEEP LEARNING IN PYTHON**

# Activation functions

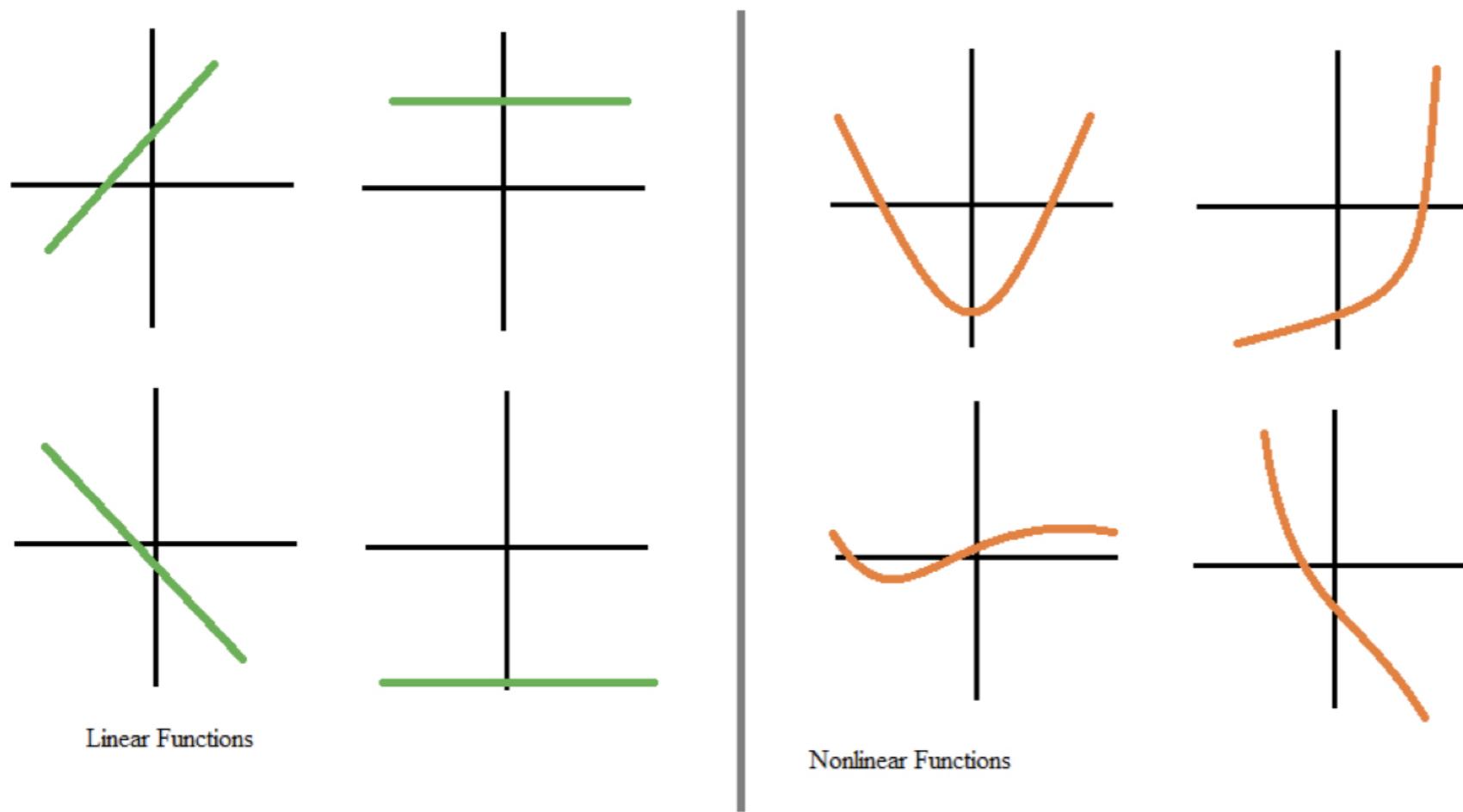
INTRODUCTION TO DEEP LEARNING IN PYTHON



Dan Becker

Data Scientist and contributor to Keras  
and TensorFlow libraries

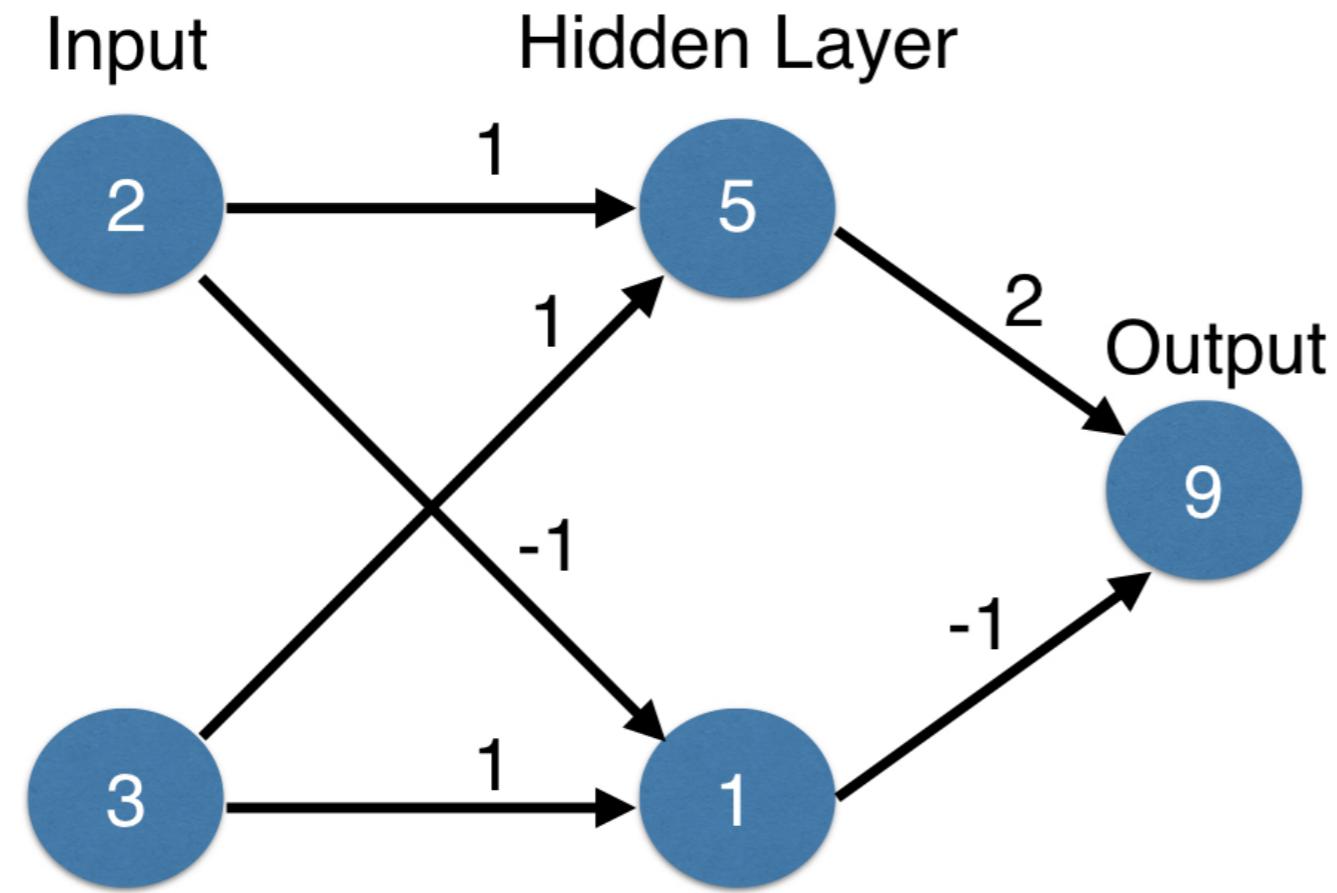
# Linear vs Nonlinear Functions



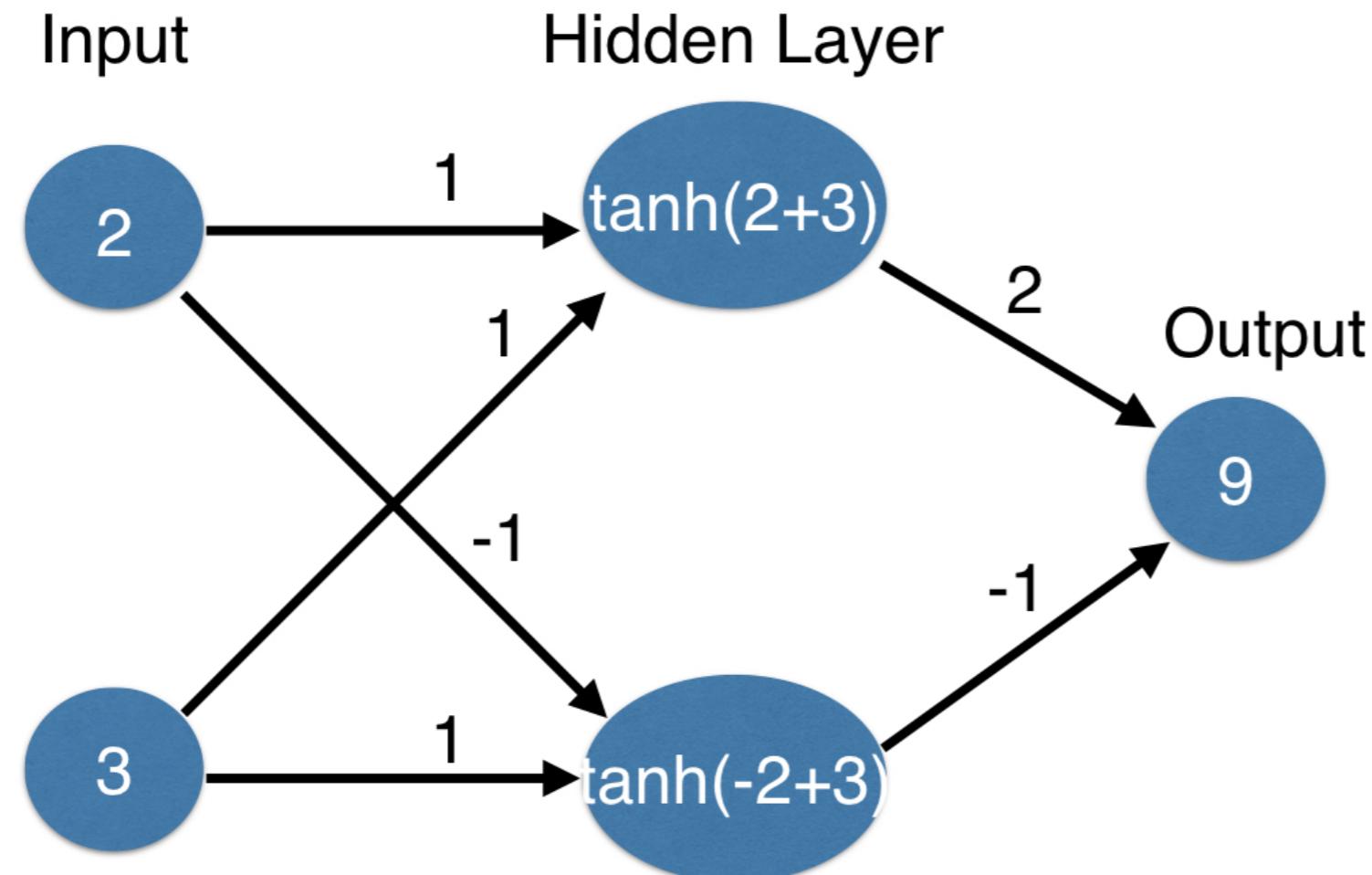
# Activation functions

- Applied to node inputs to produce node output

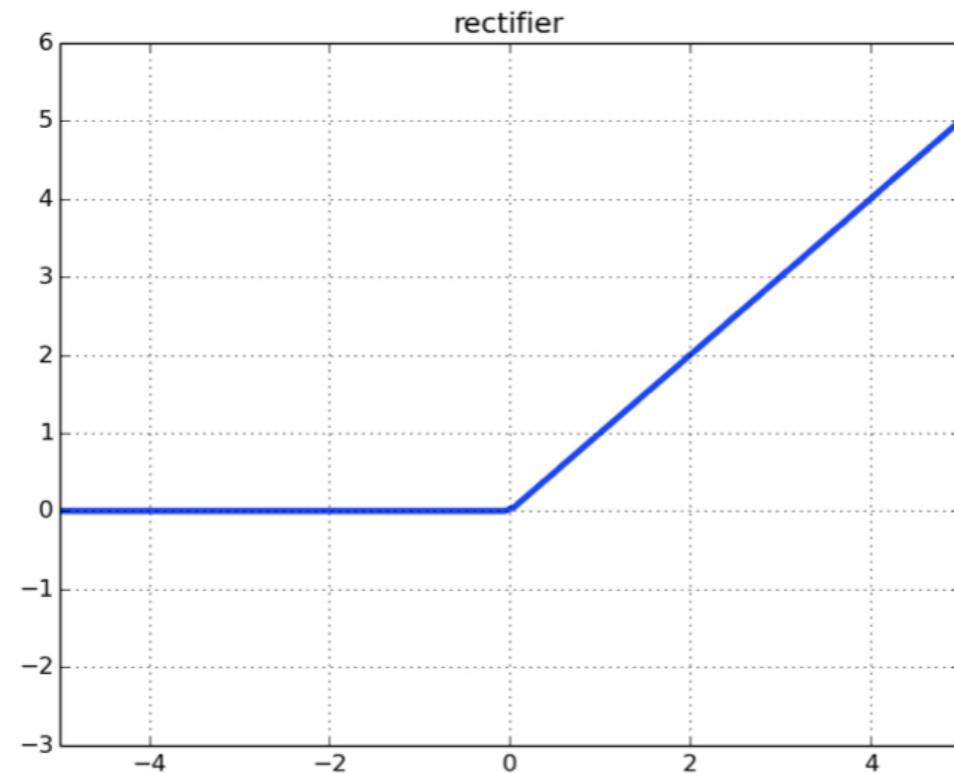
# Improving our neural network



# Activation functions



# ReLU (Rectified Linear Activation)



$$RELU(x) = \begin{cases} 0 & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases}$$

# Activation functions

```
import numpy as np  
input_data = np.array([-1, 2])  
weights = {'node_0': np.array([3, 3]),  
           'node_1': np.array([1, 5]),  
           'output': np.array([2, -1])}  
node_0_input = (input_data * weights['node_0']).sum()  
node_0_output = np.tanh(node_0_input)  
node_1_input = (input_data * weights['node_1']).sum()  
node_1_output = np.tanh(node_1_input)  
hidden_layer_outputs = np.array([node_0_output, node_1_output])  
output = (hidden_layer_outputs * weights['output']).sum()
```

```
print(output)
```

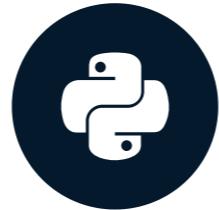
```
1.2382242525694254
```

# **Let's practice!**

**INTRODUCTION TO DEEP LEARNING IN PYTHON**

# Deeper networks

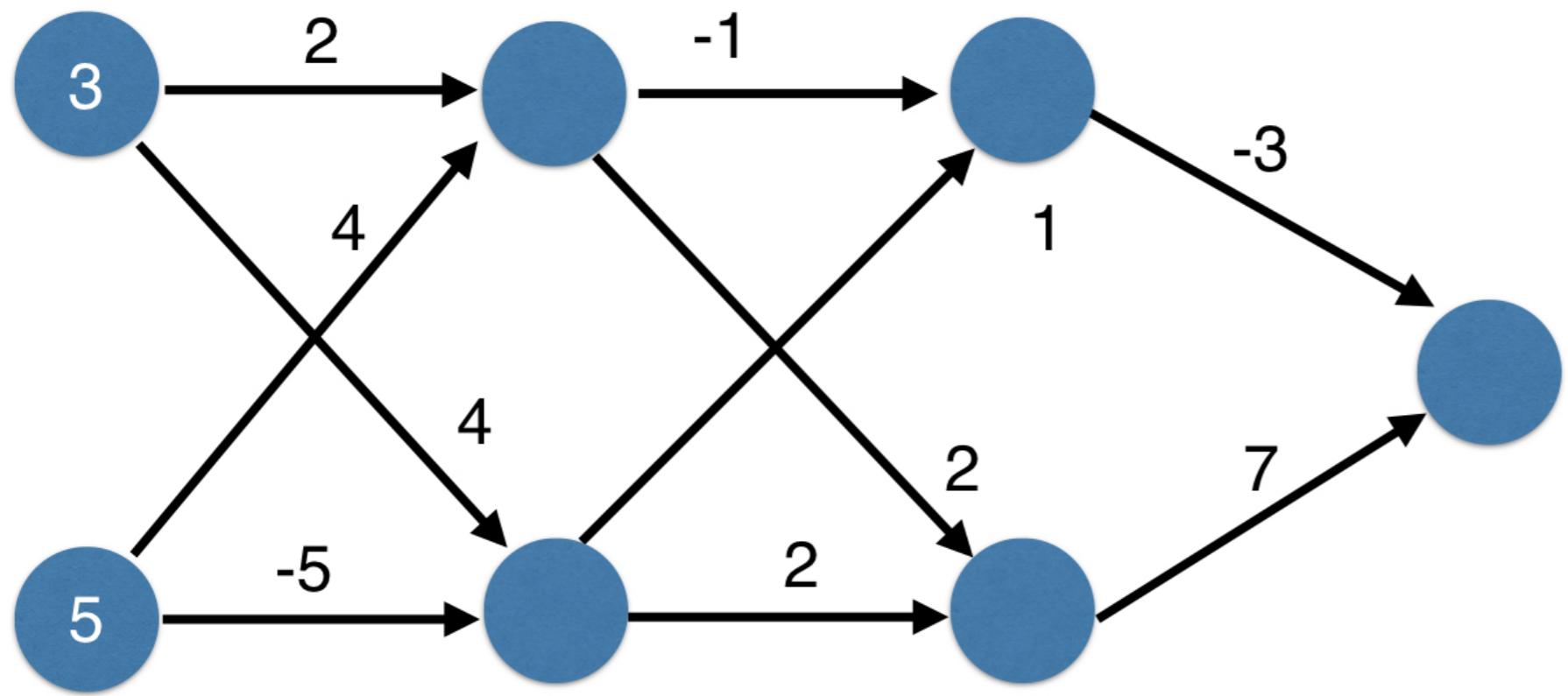
INTRODUCTION TO DEEP LEARNING IN PYTHON



Dan Becker

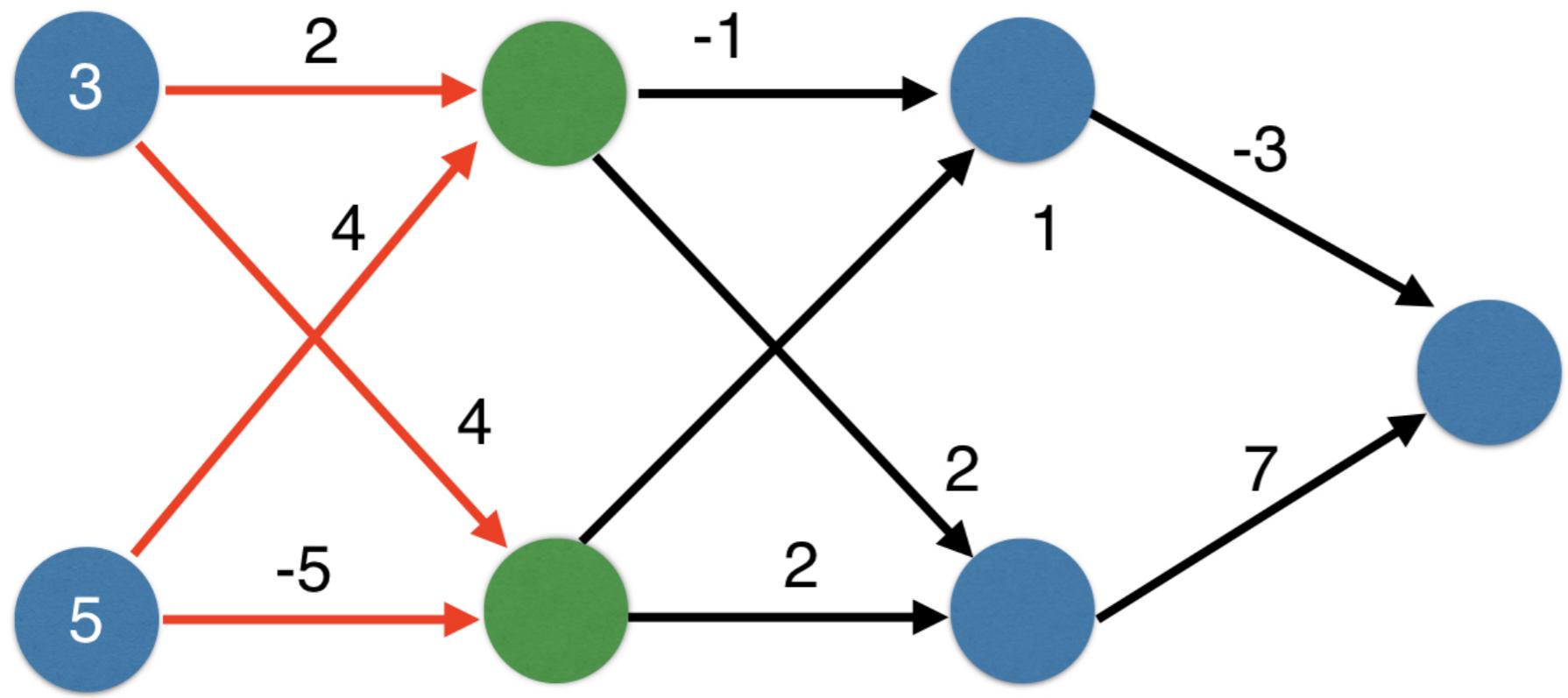
Data Scientist and contributor to Keras  
and TensorFlow libraries

# Multiple hidden layers



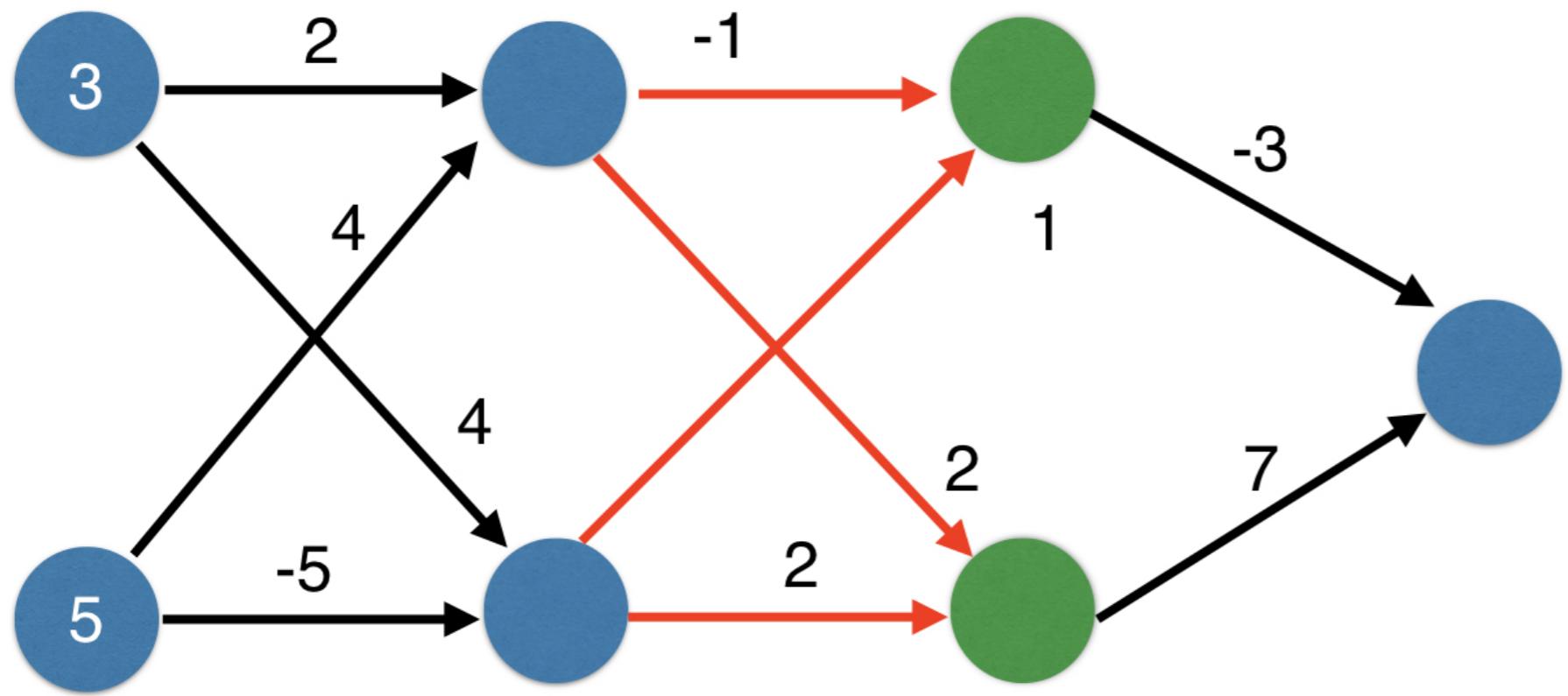
Calculate with ReLU Activation Function

# Multiple hidden layers



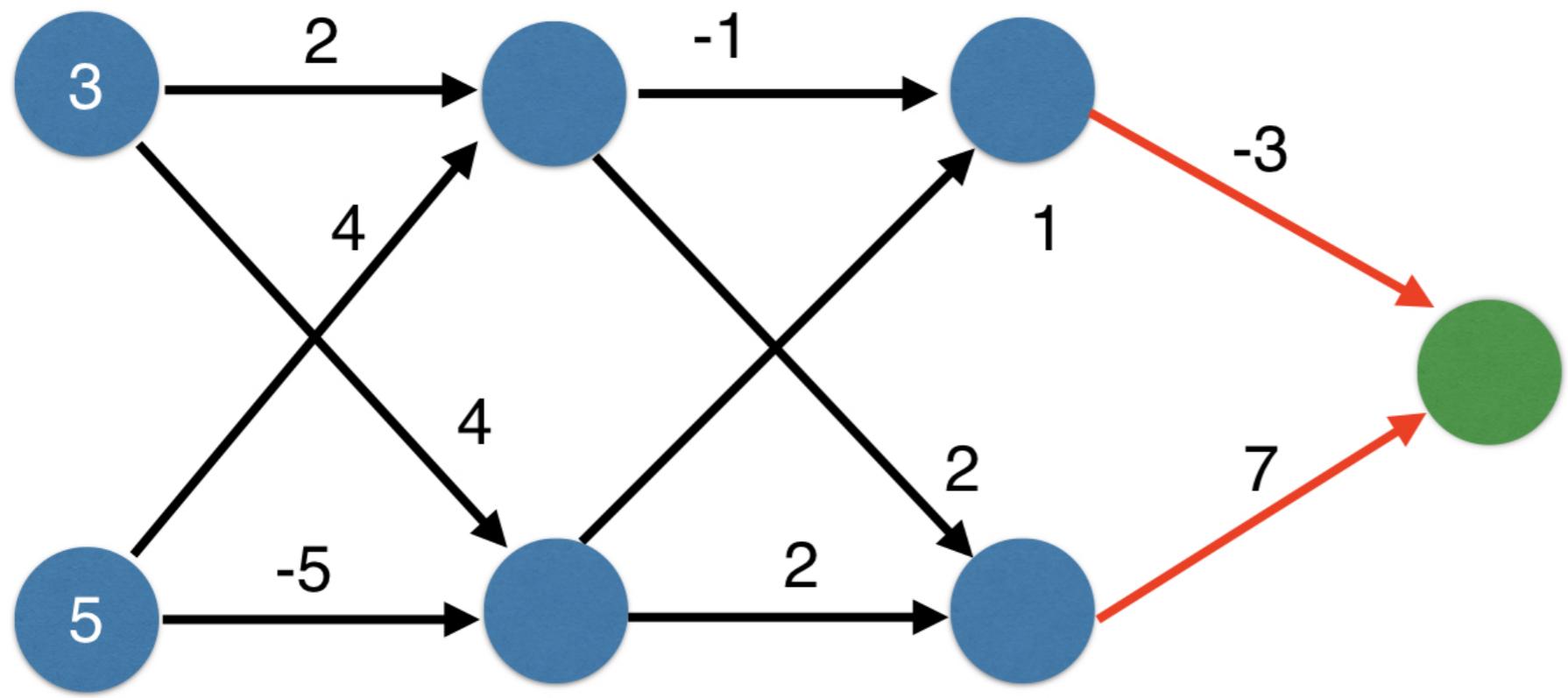
Calculate with ReLU Activation Function

# Multiple hidden layers



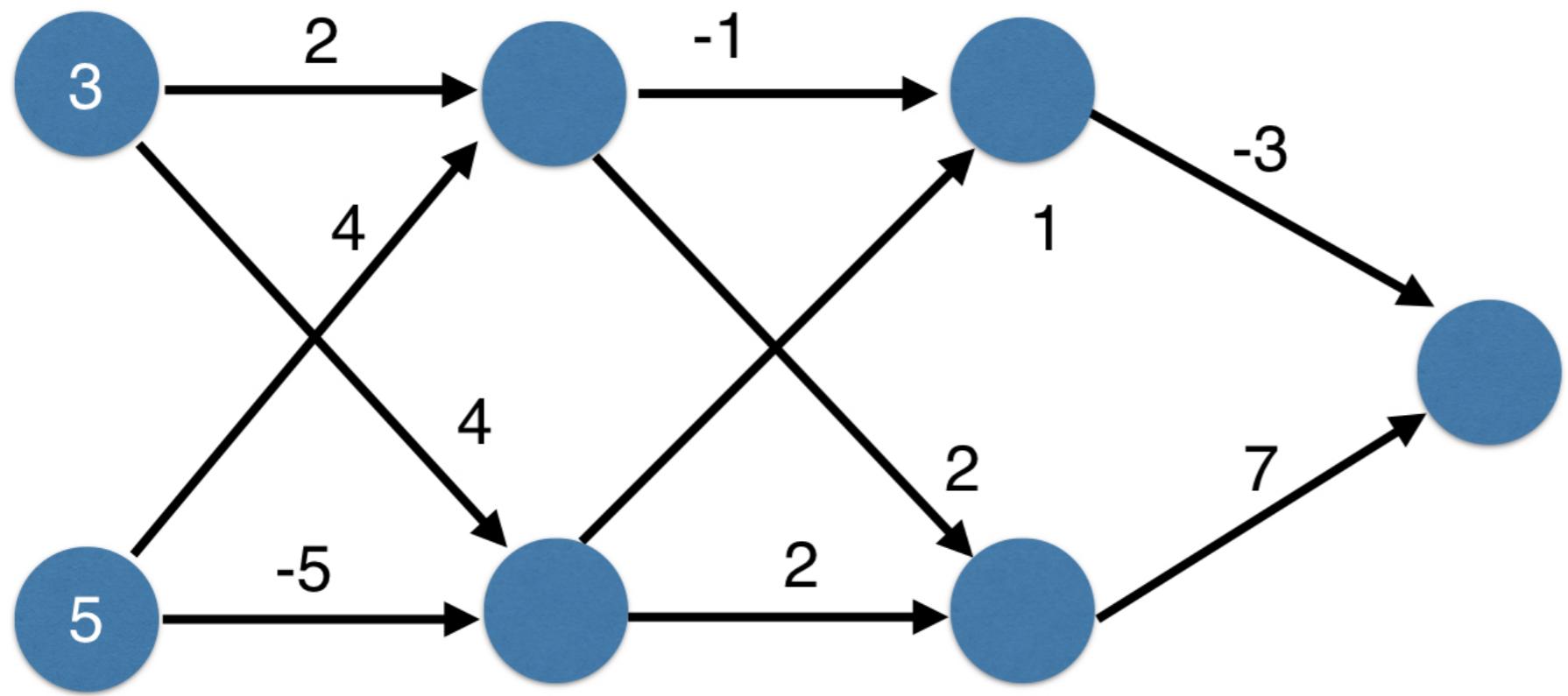
Calculate with ReLU Activation Function

# Multiple hidden layers



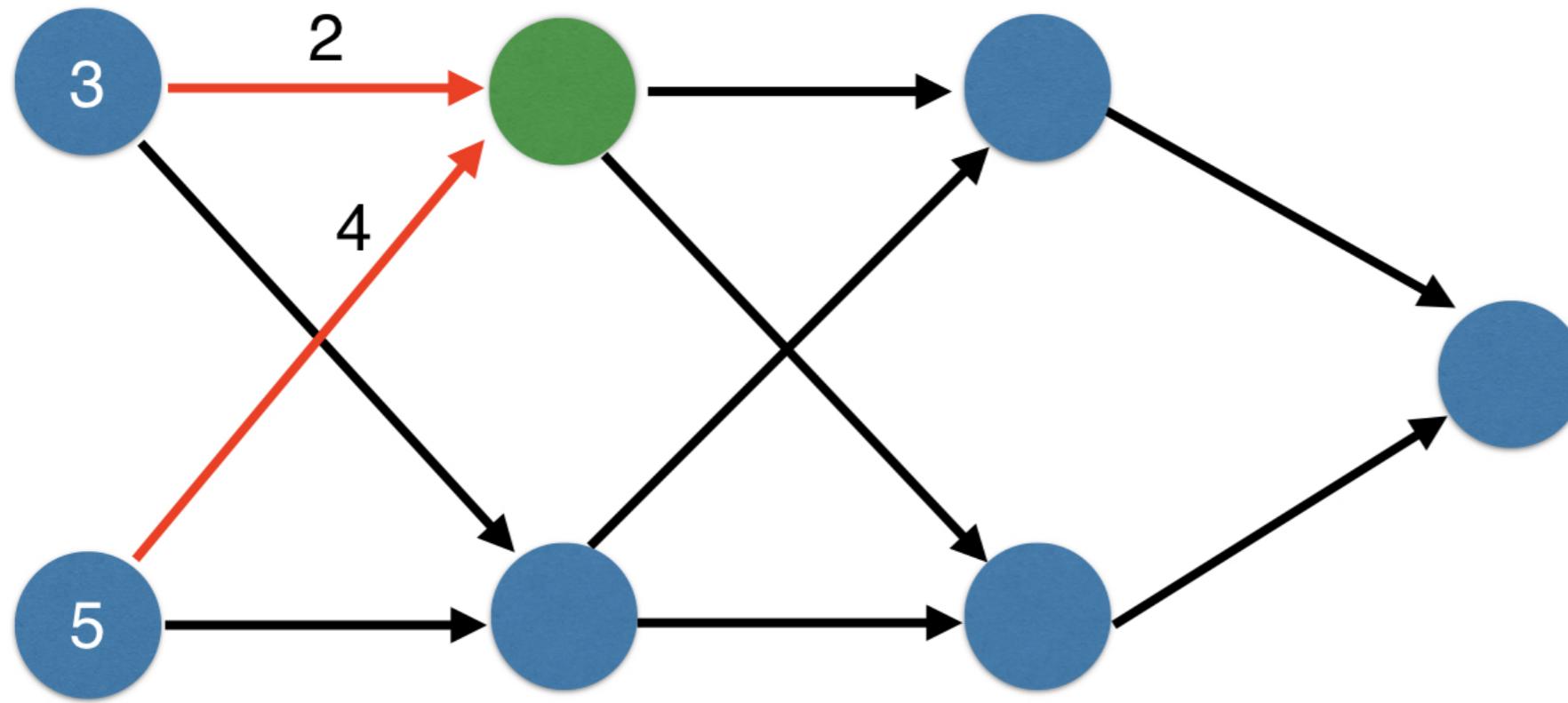
Calculate with ReLU Activation Function

# Multiple hidden layers



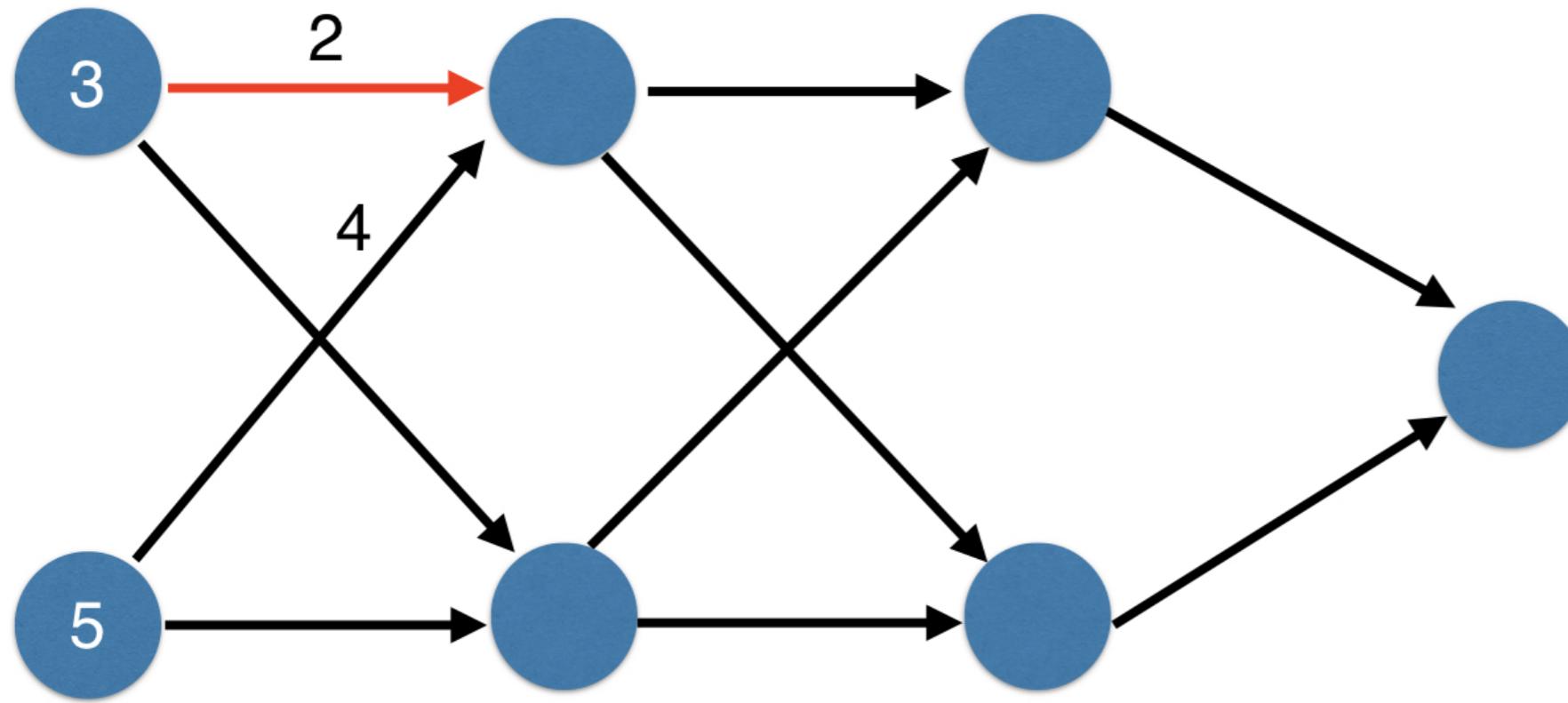
Calculate with ReLU Activation Function

# Multiple hidden layers



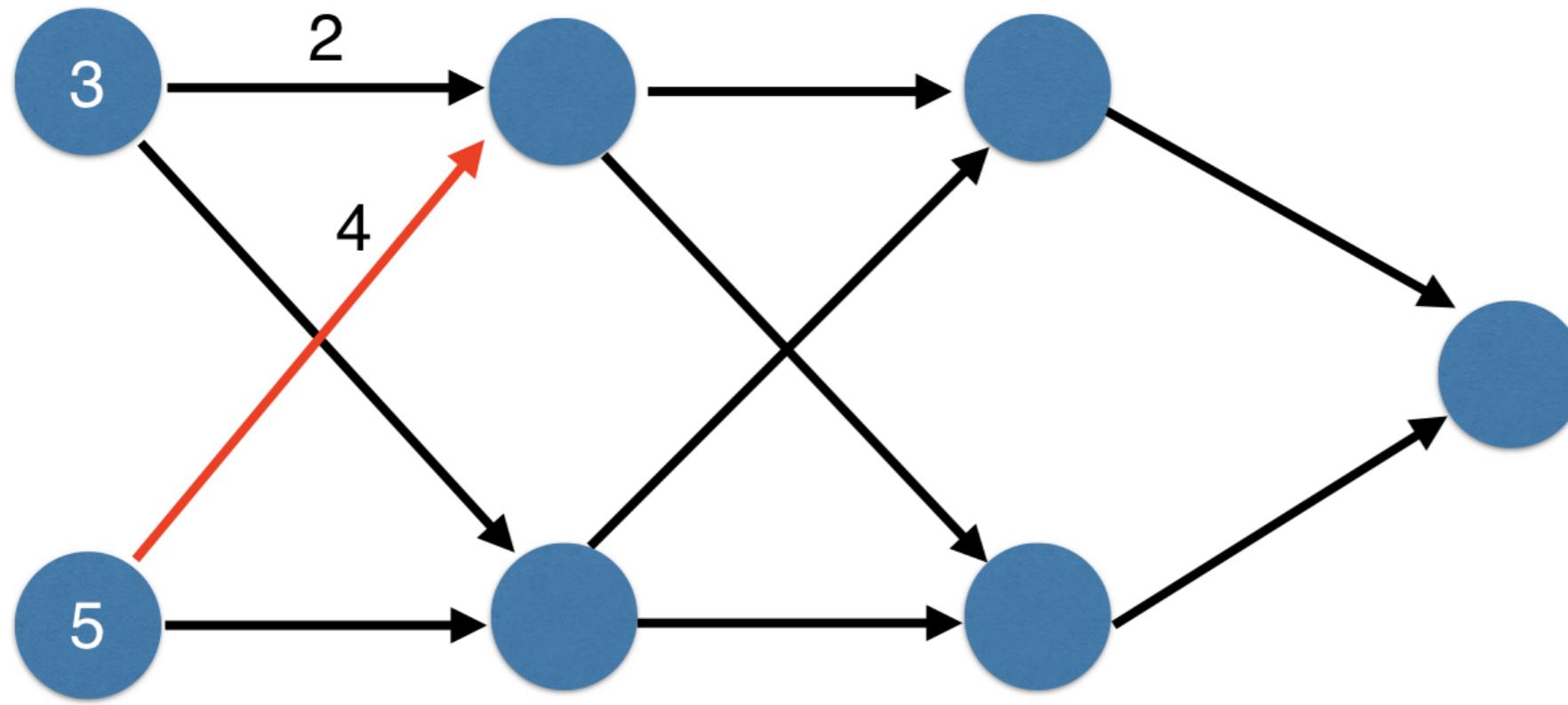
Calculate with ReLU Activation Function

# Multiple hidden layers



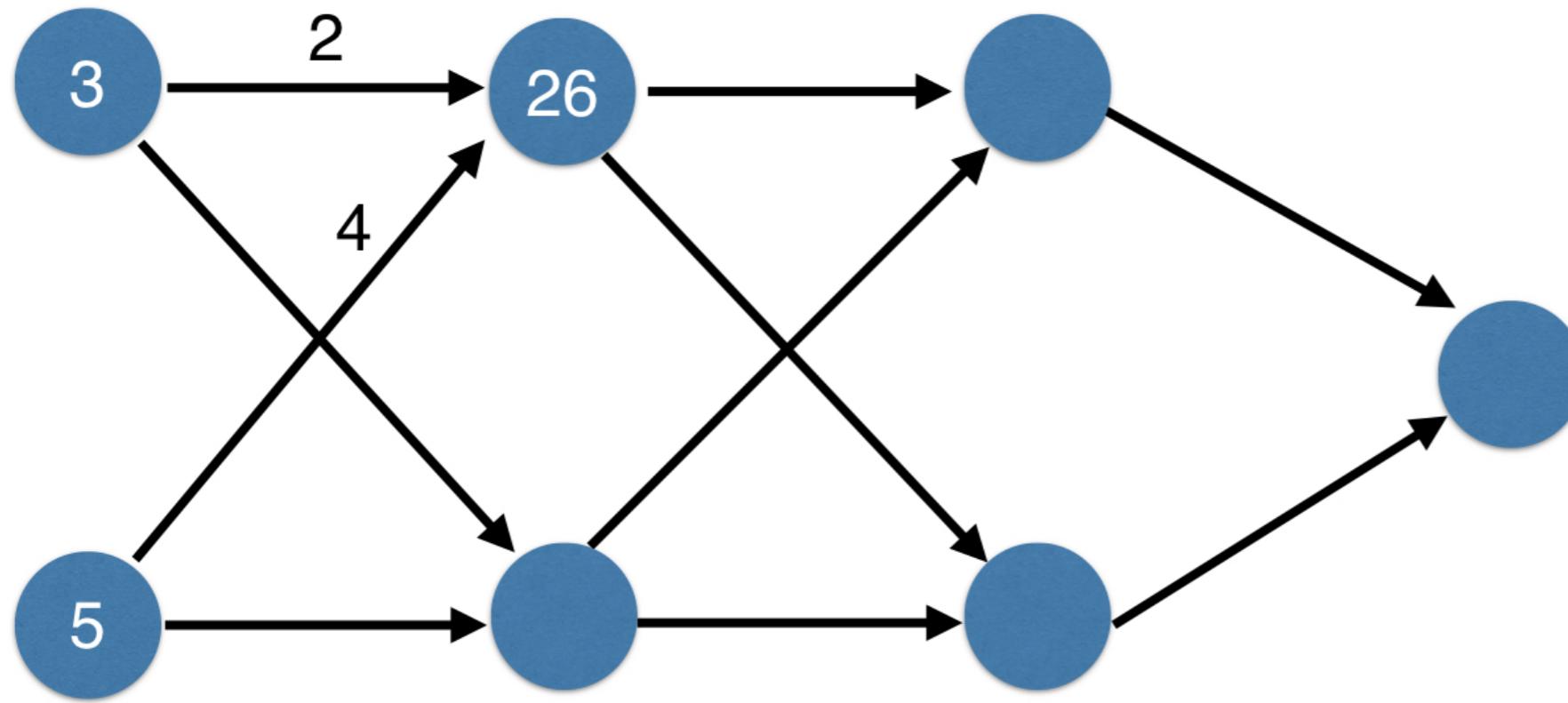
Calculate with ReLU Activation Function

# Multiple hidden layers



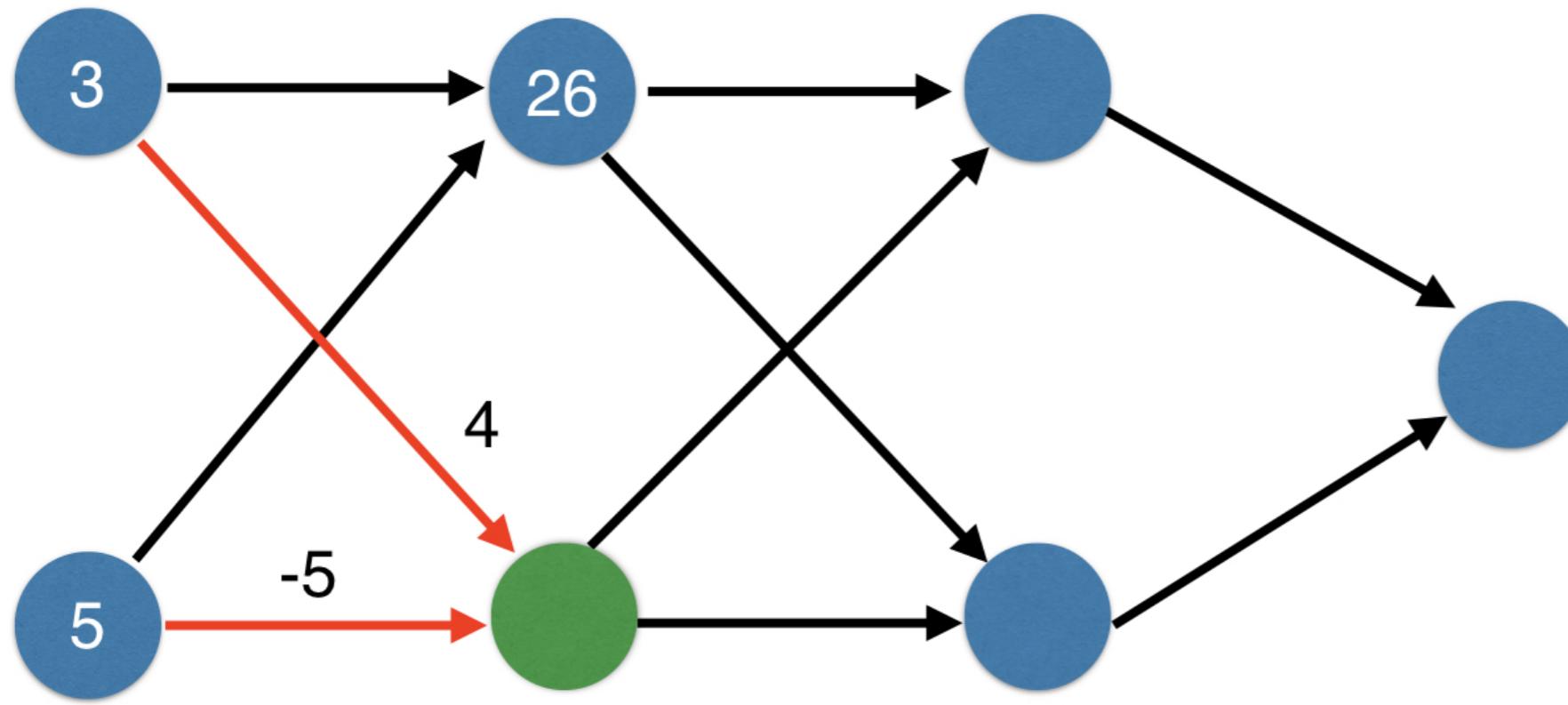
Calculate with ReLU Activation Function

# Multiple hidden layers



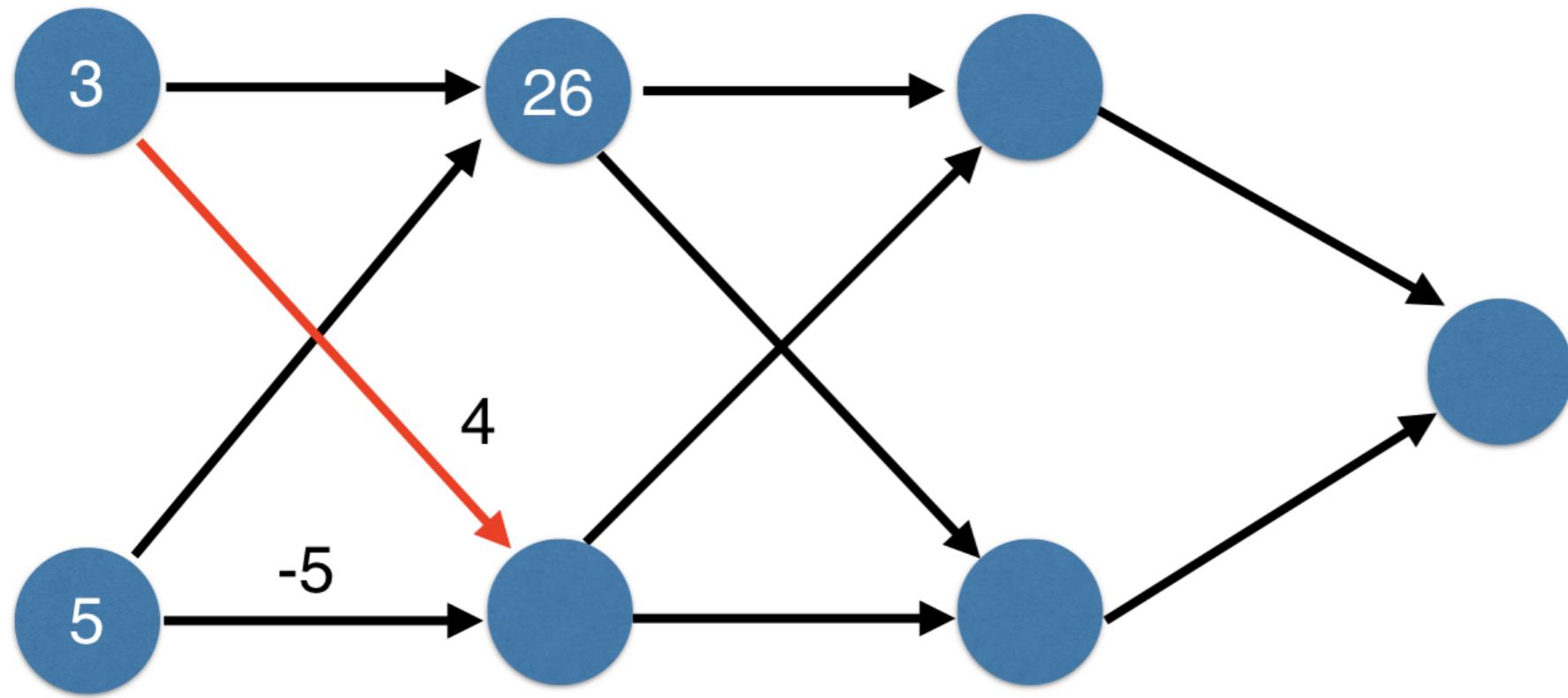
Calculate with ReLU Activation Function

# Multiple hidden layers



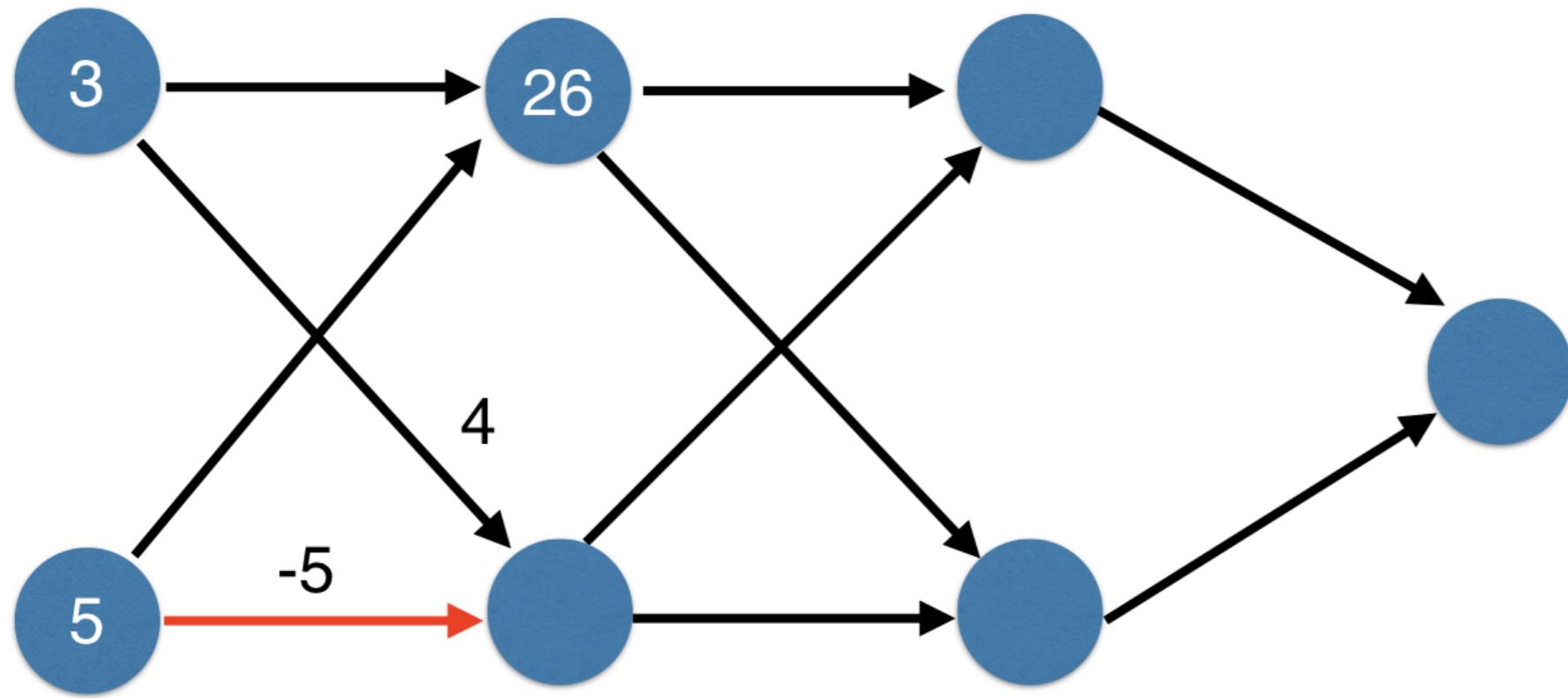
Calculate with ReLU Activation Function

# Multiple hidden layers



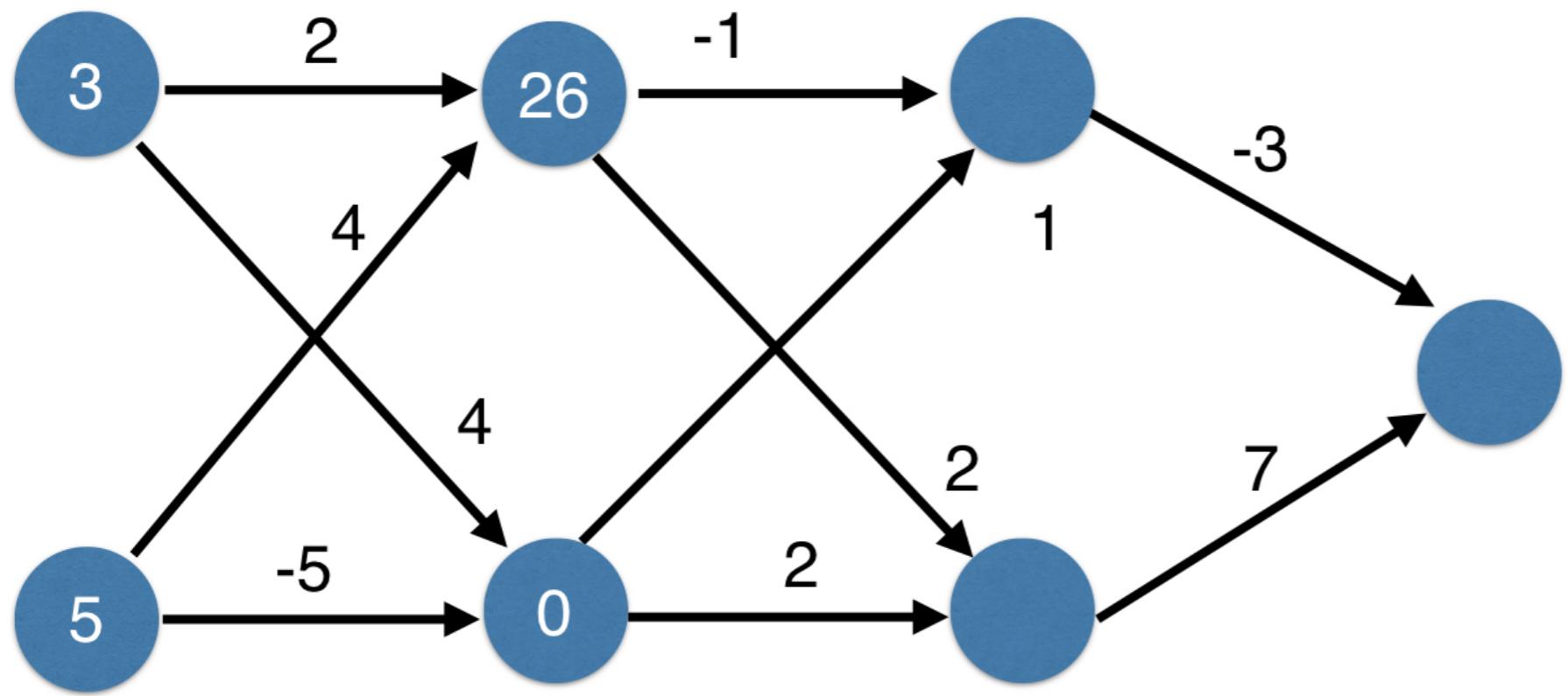
Calculate with ReLU Activation Function

# Multiple hidden layers



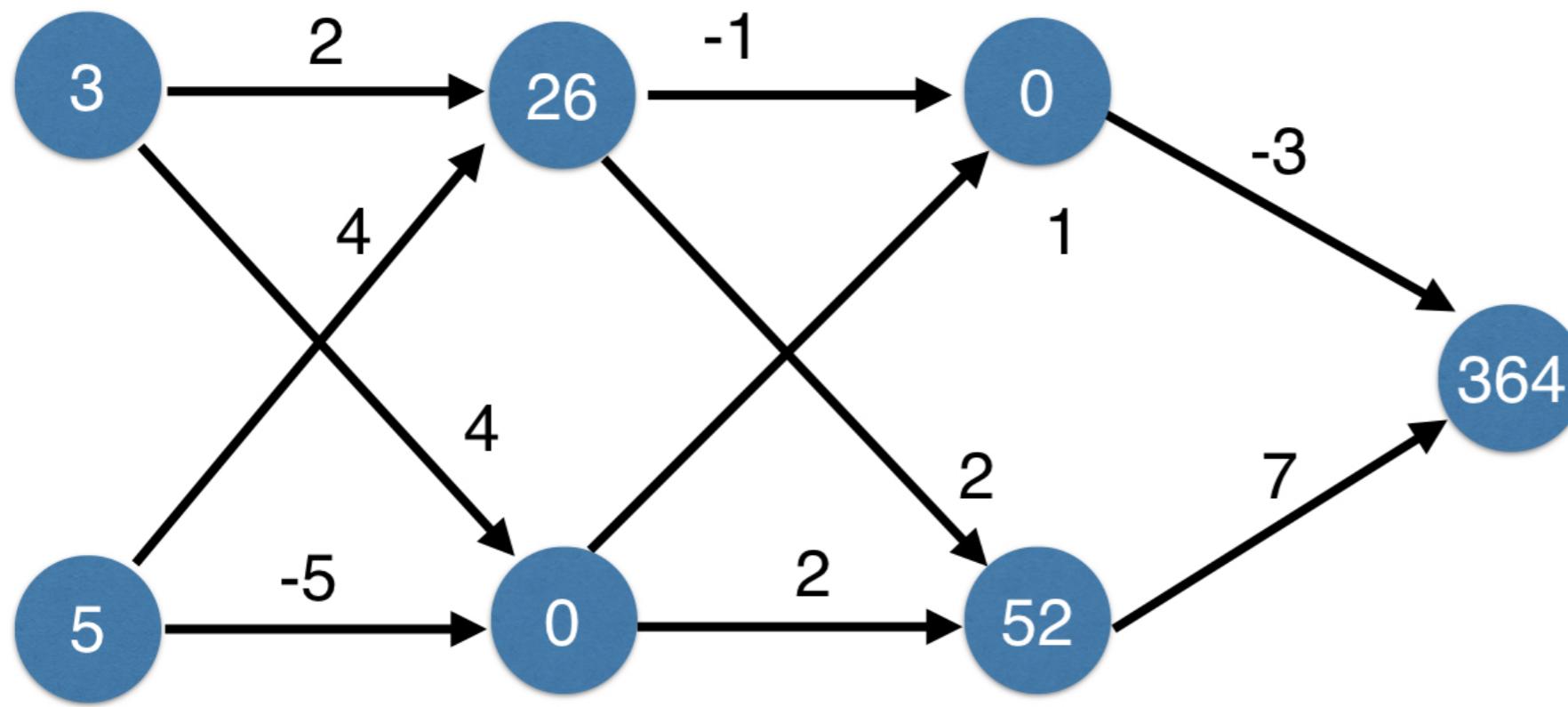
Calculate with ReLU Activation Function

# Multiple hidden layers



Calculate with ReLU Activation Function

# Multiple hidden layers

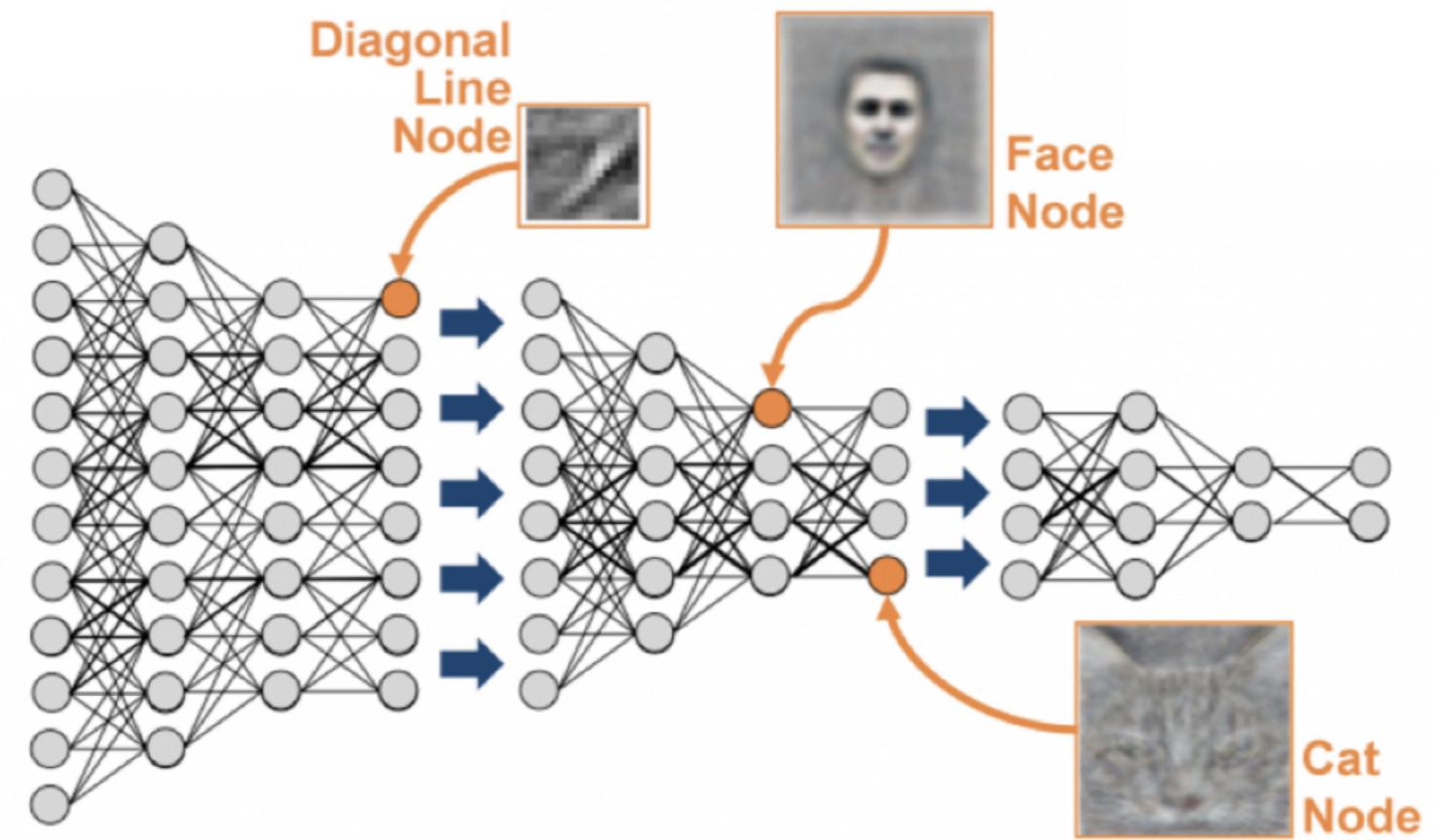


Calculate with ReLU Activation Function

# Representation learning

- Deep networks internally build representations of patterns in the data
- Partially replace the need for feature engineering
- Subsequent layers build increasingly sophisticated representations of raw data

# Representation learning



# Deep learning

- Modeler doesn't need to specify the interactions
- When you train the model, the neural network gets weights that find the relevant patterns to make better predictions

# **Let's practice!**

**INTRODUCTION TO DEEP LEARNING IN PYTHON**