Lecture 12 & 13:
Introduction to Deep Learning

Habib Gültekin
DOĞUŞ TEKNOLOJİ – ML/AI Solutions

OUTLINE

- Introduction to Deep Learning
- Forward Propagation
- Deeper Networks
- Deep Learning Models with Keras

Doğuş Technology

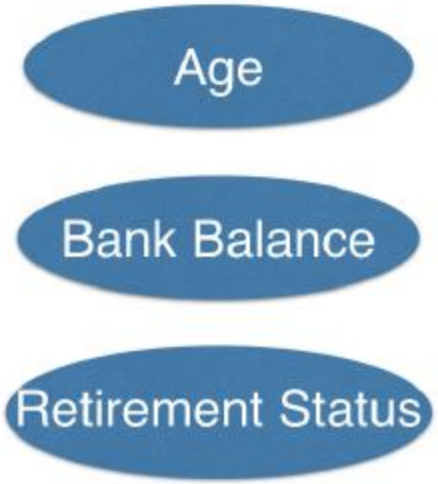
- Imagine you work for a bank
- You need to predict how many transactions each customer will make next year

Introduction to DL Example

Age

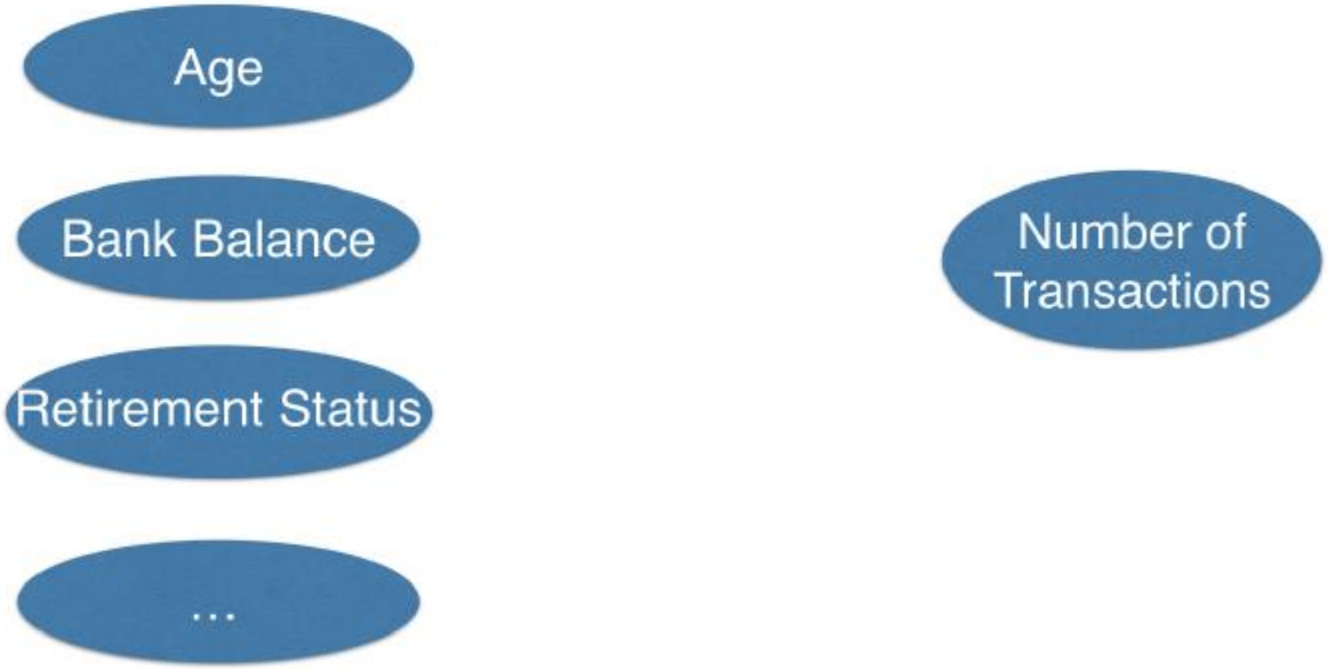
Introduction to DL Example

Introduction to DL Example

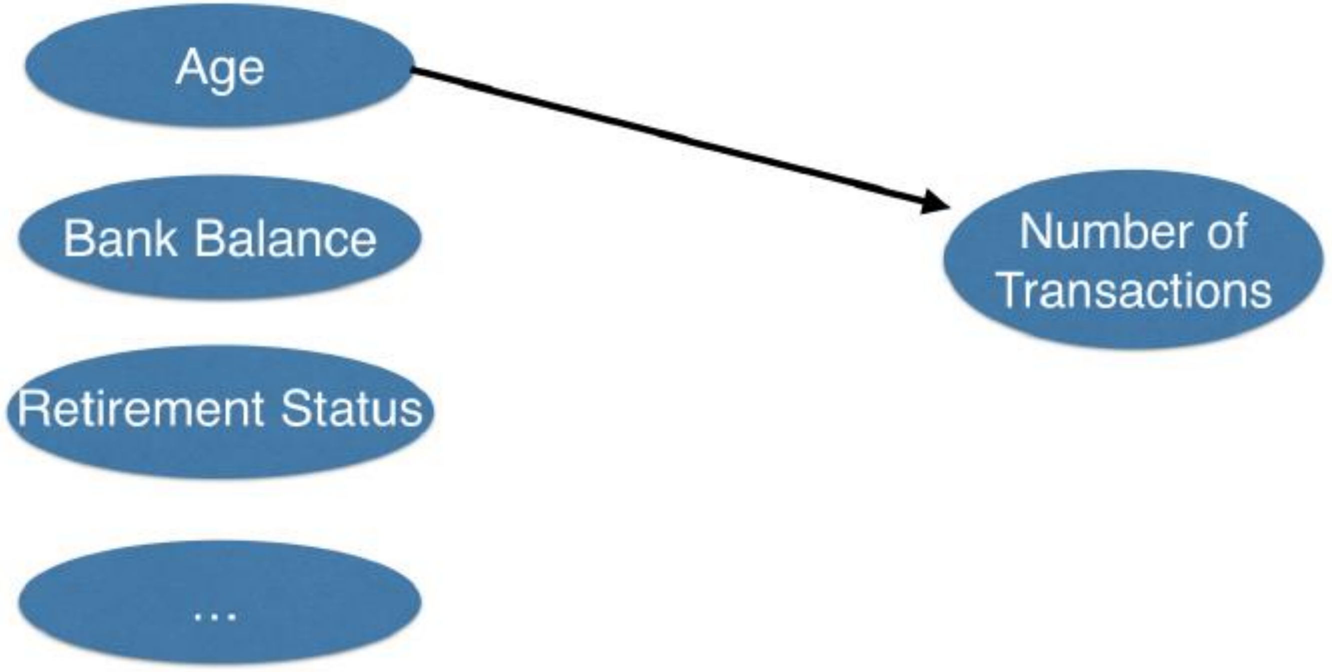


Introduction to DL Example

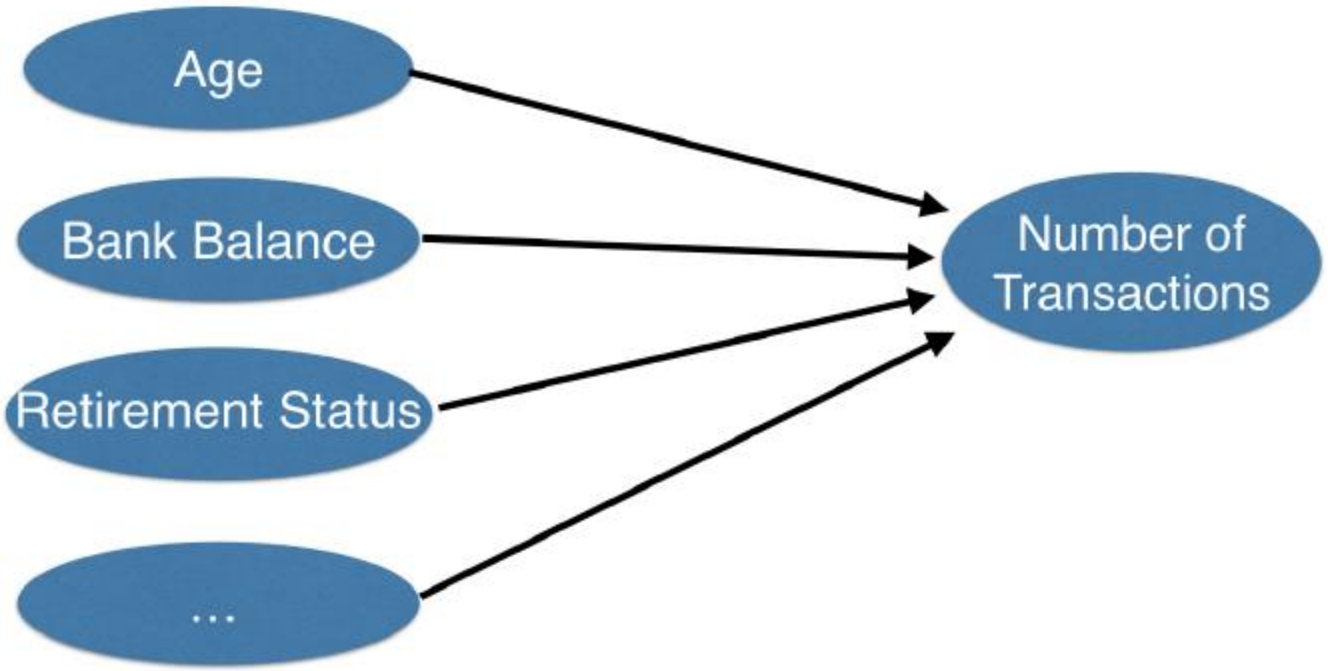
Introduction to DL Example



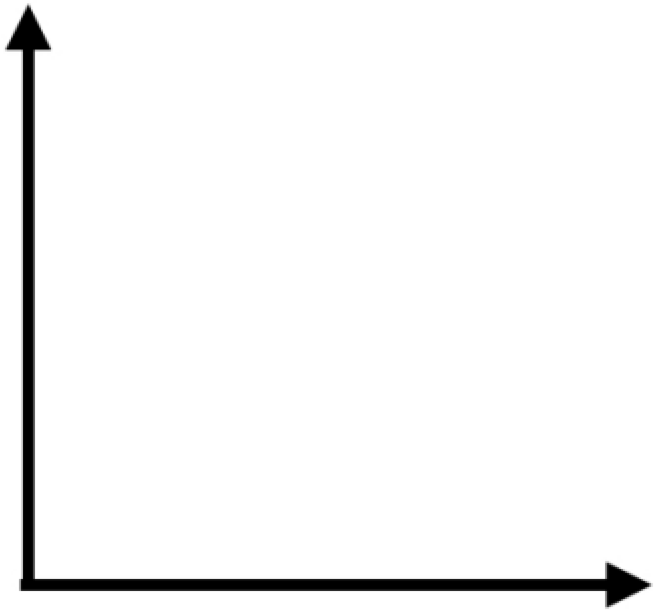
Introduction to DL Example



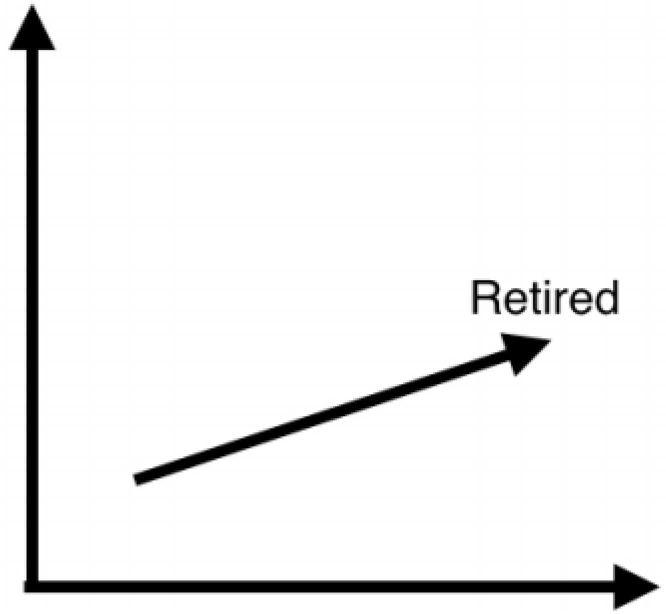
Introduction to DL Example



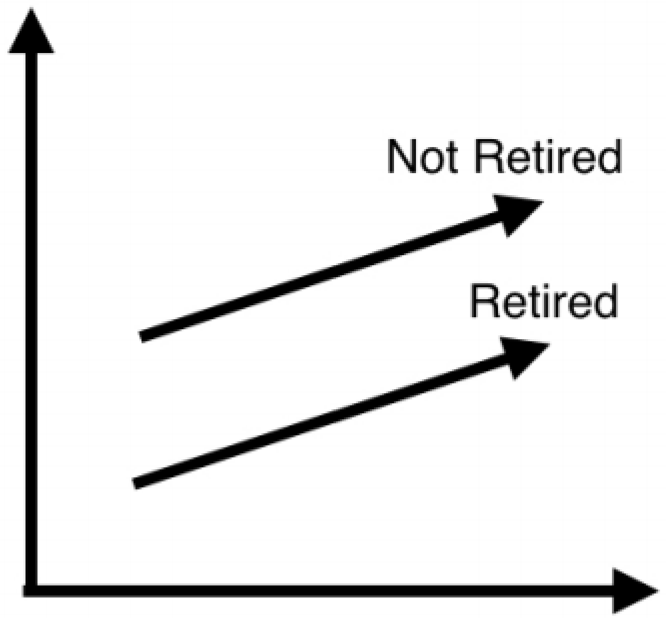
Introduction to DL Example



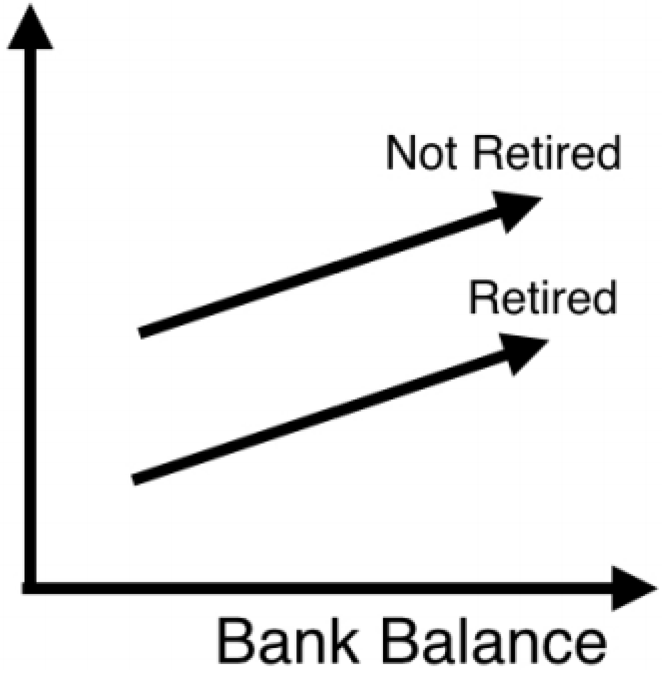
Introduction to DL Example



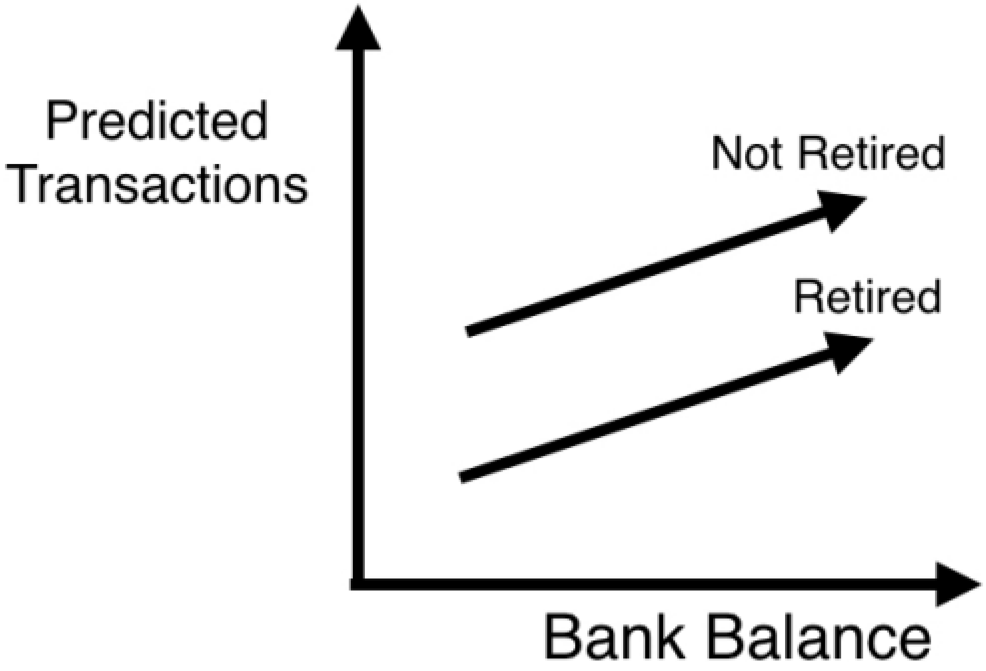
Introduction to DL Example

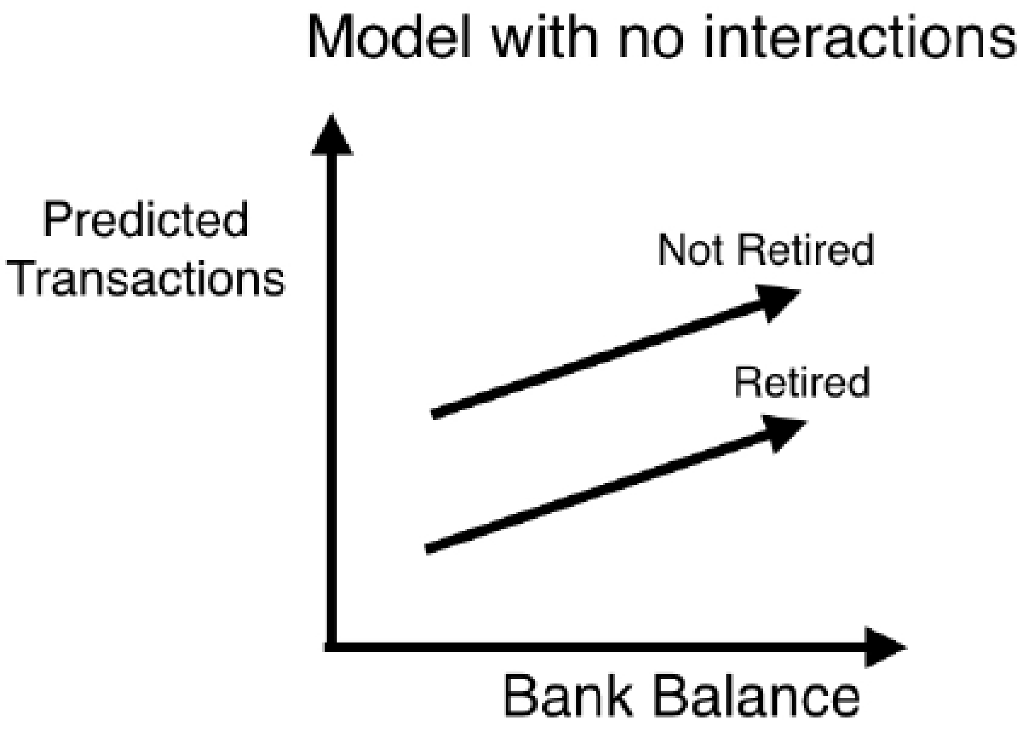


Introduction to DL Example

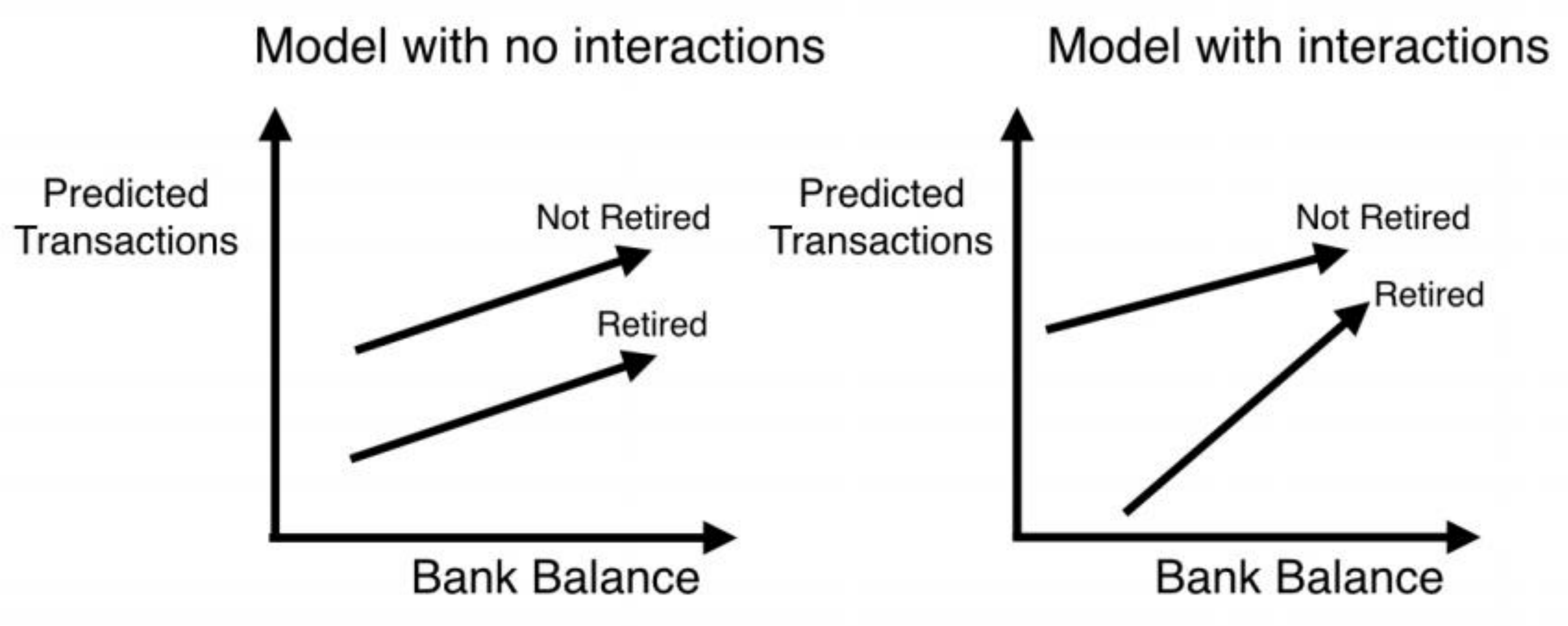


Introduction to DL Example




Model with no interactions
Predicted
Transactions
Not Retired
Retired
Bank Balance

Introduction to DL Example



Interactions

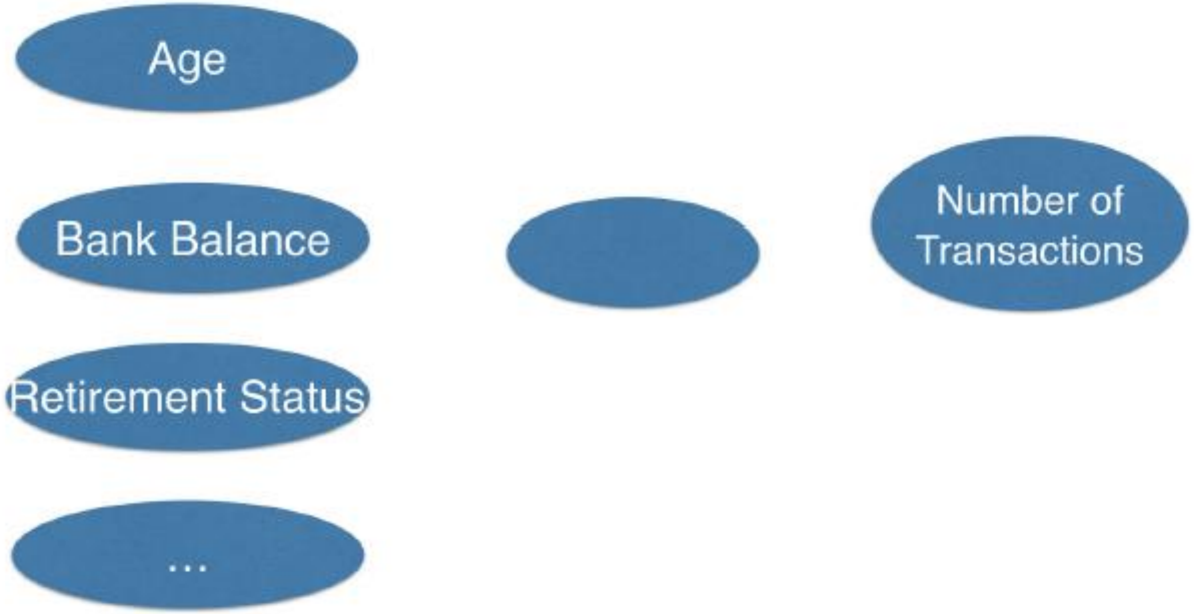
- Neural networks account for interactions really well
- Deep learning uses especially powerful neural networks
  - Text
  - Images
  - Videos
  - Audio
  - Source code

Build and tune deep learning models using keras

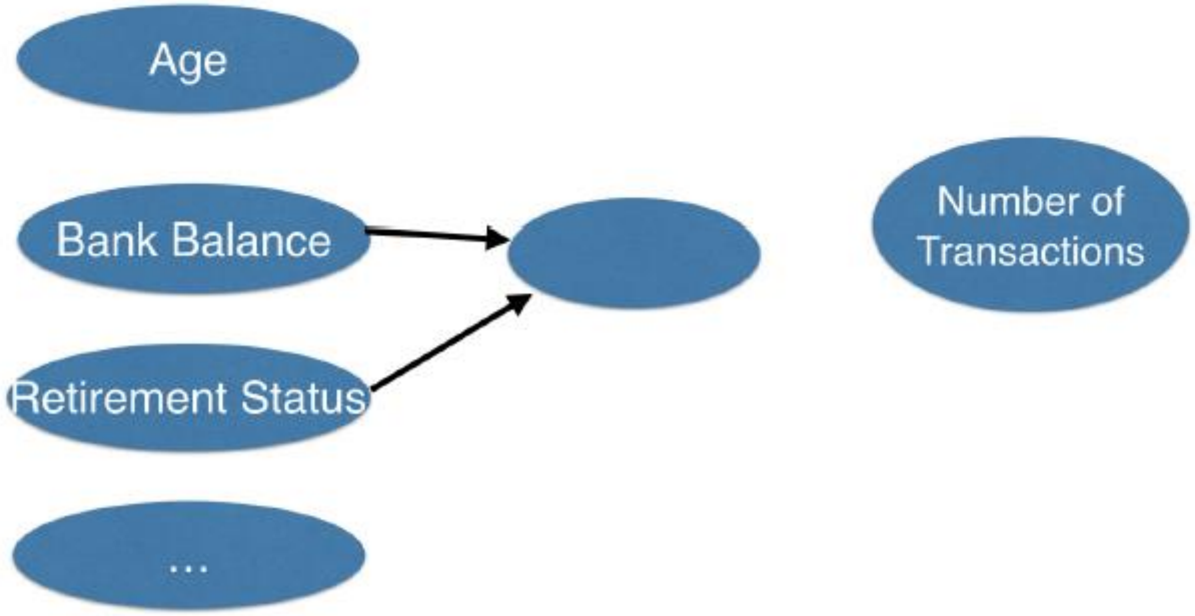
```
import numpy as np
from keras.layers import Dense
from keras.models import Sequential
predictors = np.loadtxt('predictors_data.csv', delimiter=',')
n_cols = predictors.shape[1]
model = Sequential()

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

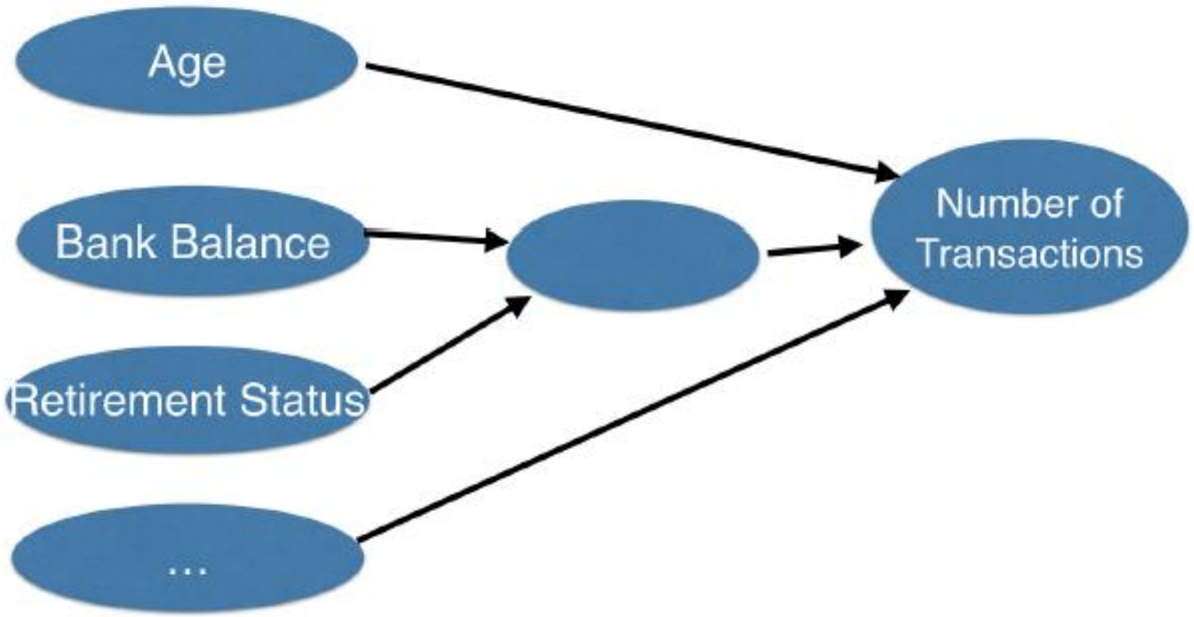
Deep learning models capture interactions



Deep learning models capture interactions



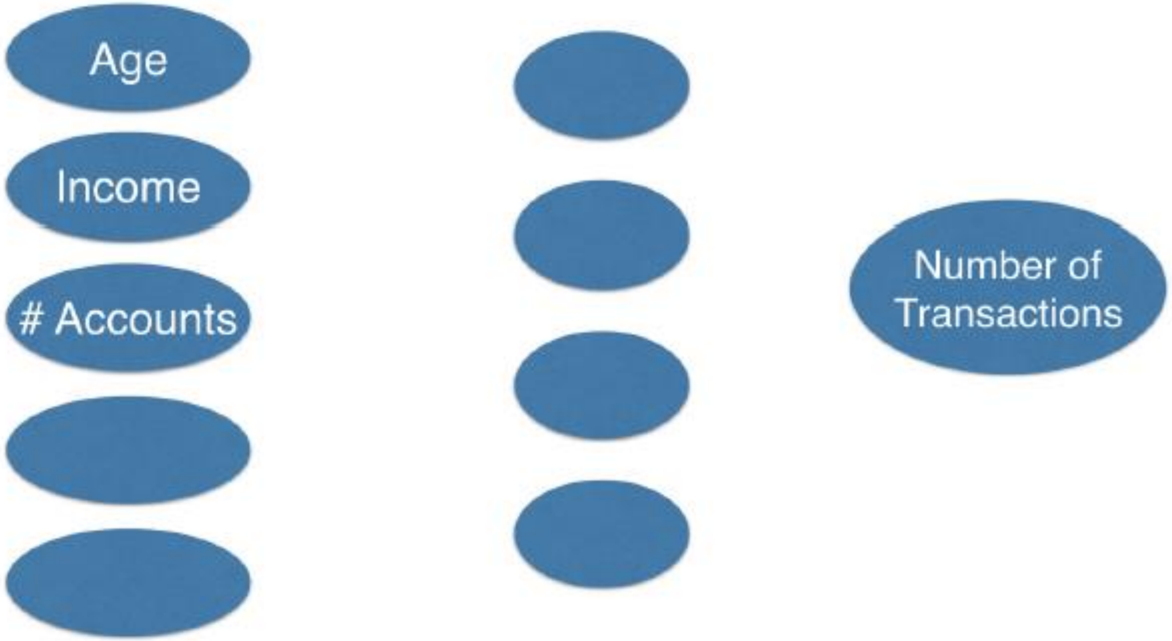
Deep learning models capture interactions



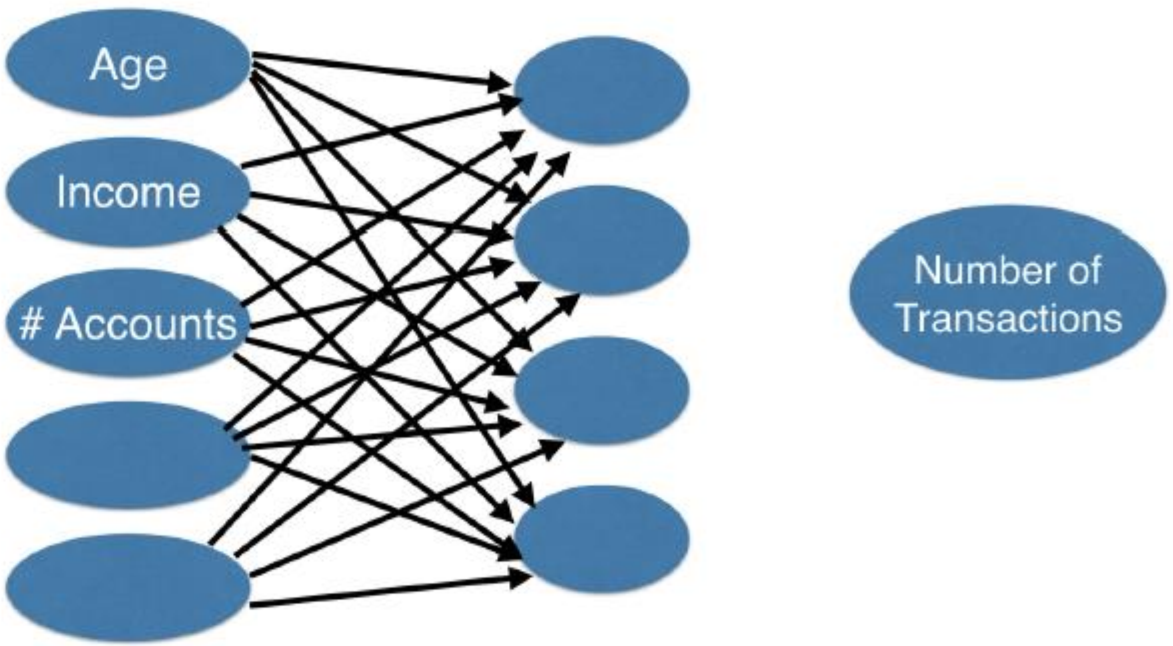
Interactions in neural network

Interactions in neural network

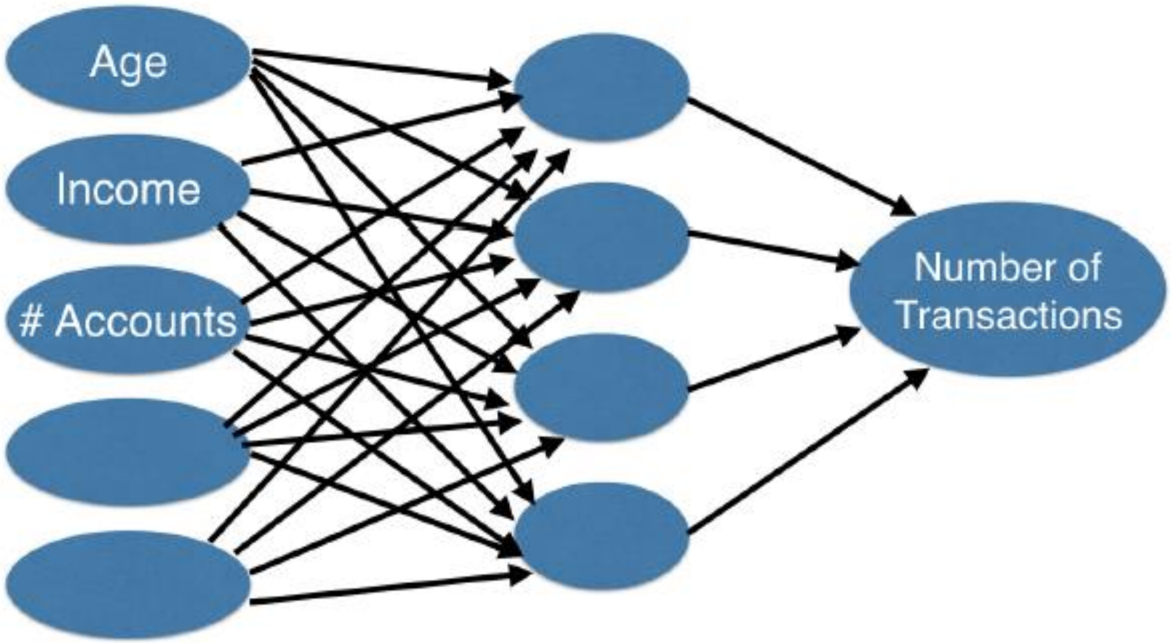
Interactions in neural network



Interactions in neural network



Interactions in neural network



OUTLINE

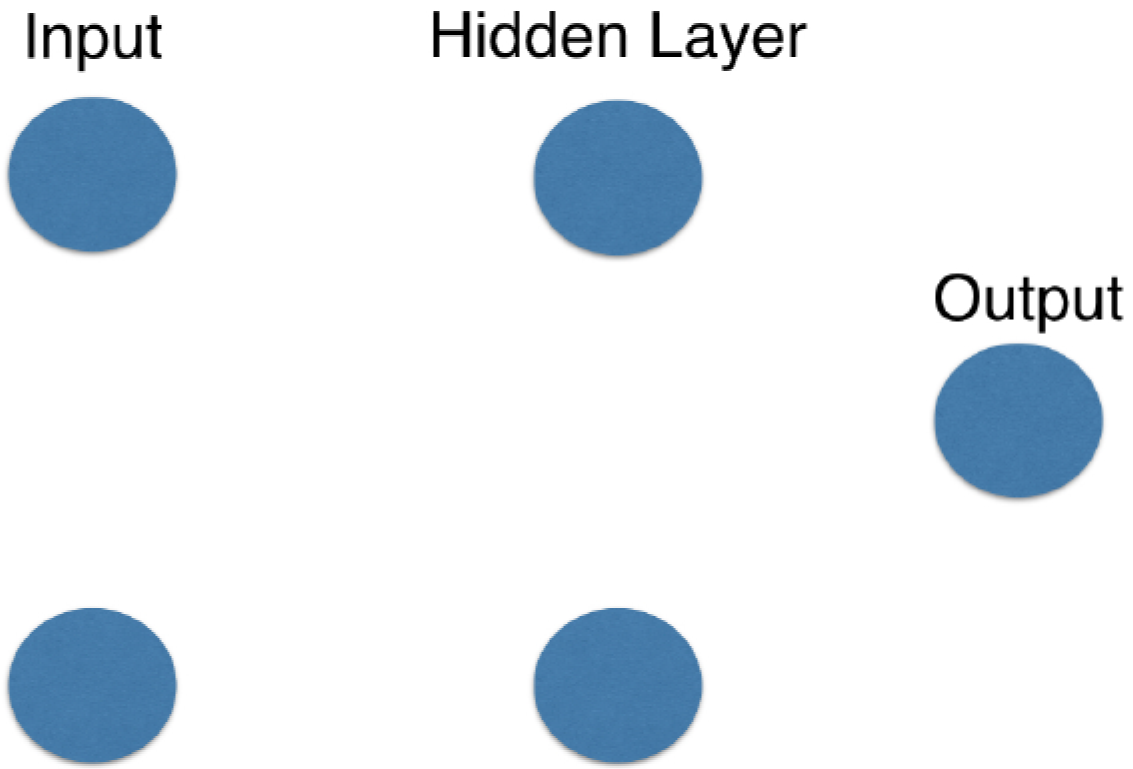
- Introduction to Deep Learning
- **Forward Propagation**
- Deeper Networks
- Deep Learning Models with Keras

Bank transactions example

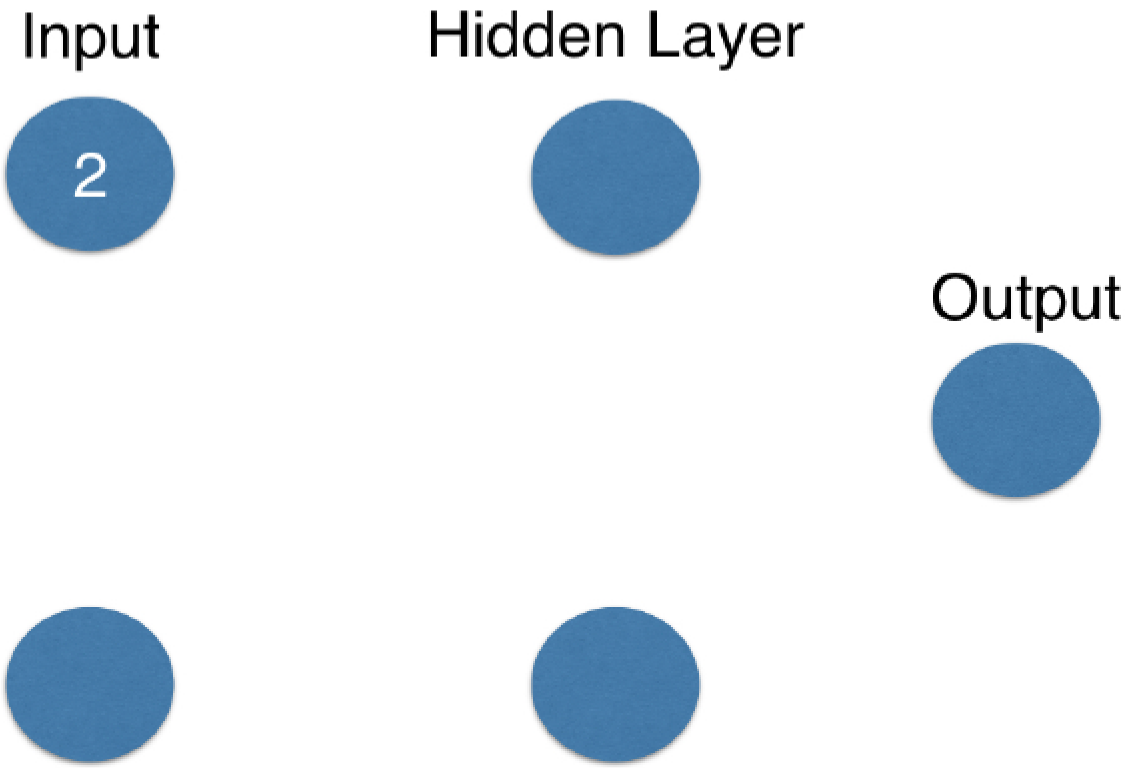
Make predictions based on:

- Number of children
- Number of existing accounts

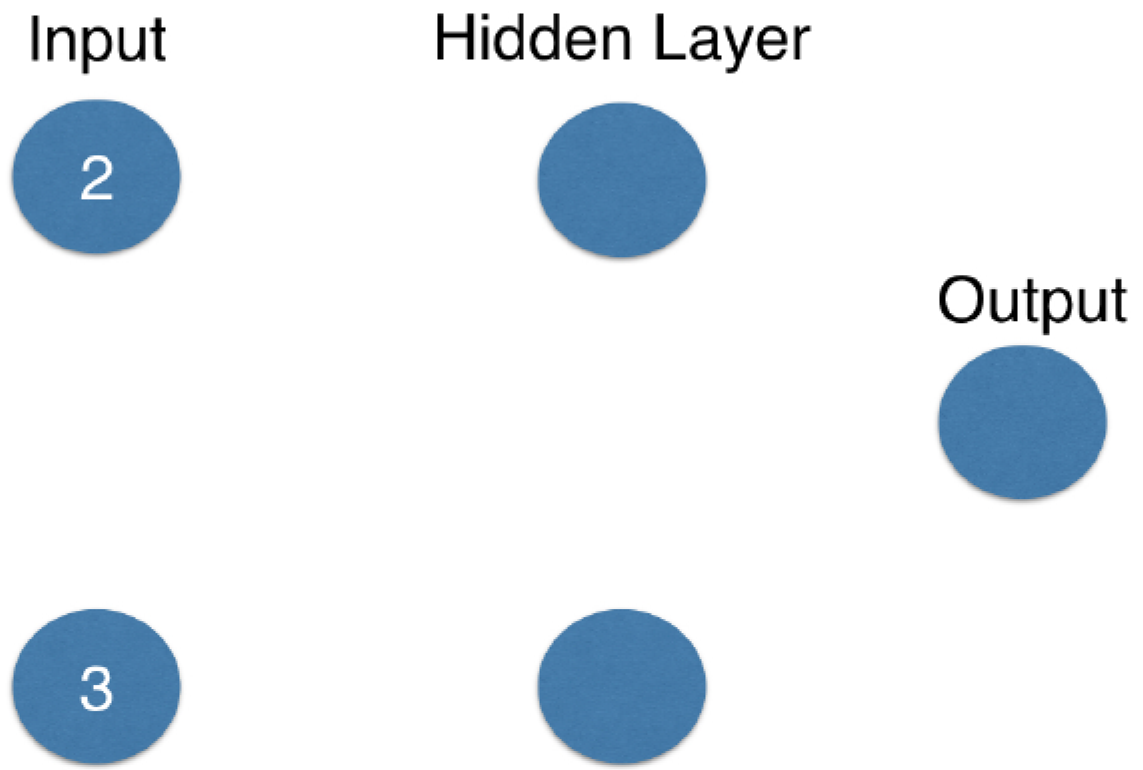
Forward propagation



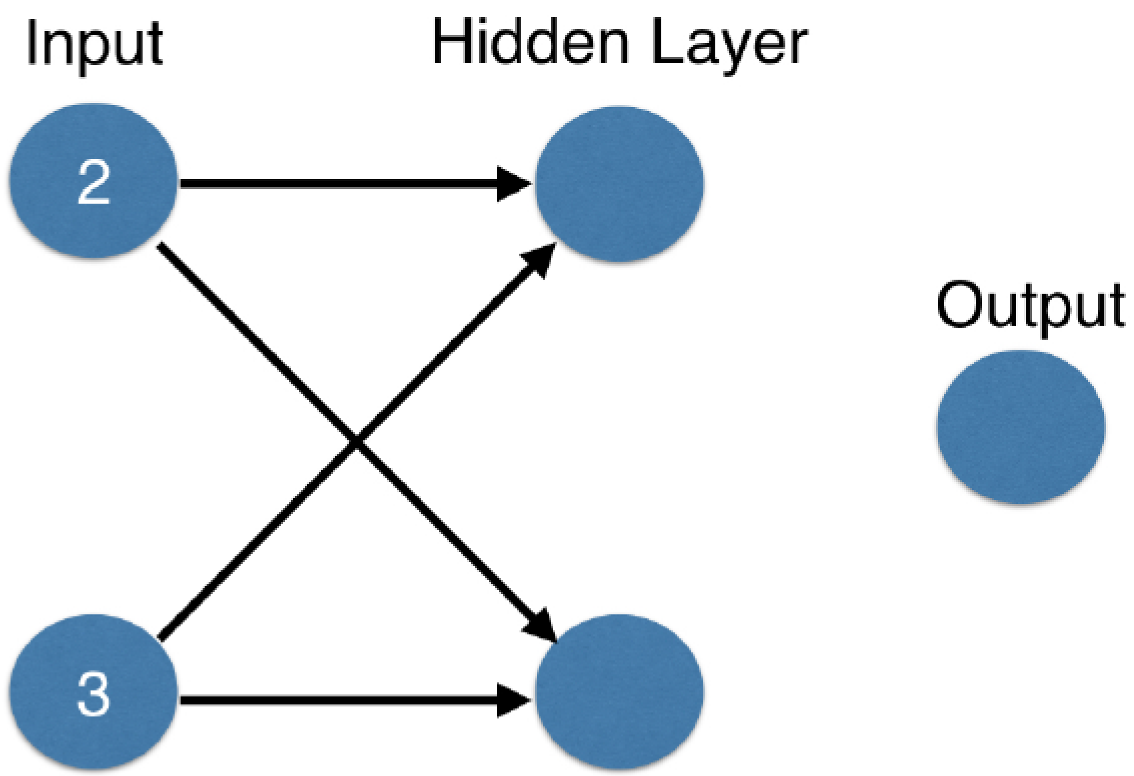
Forward propagation



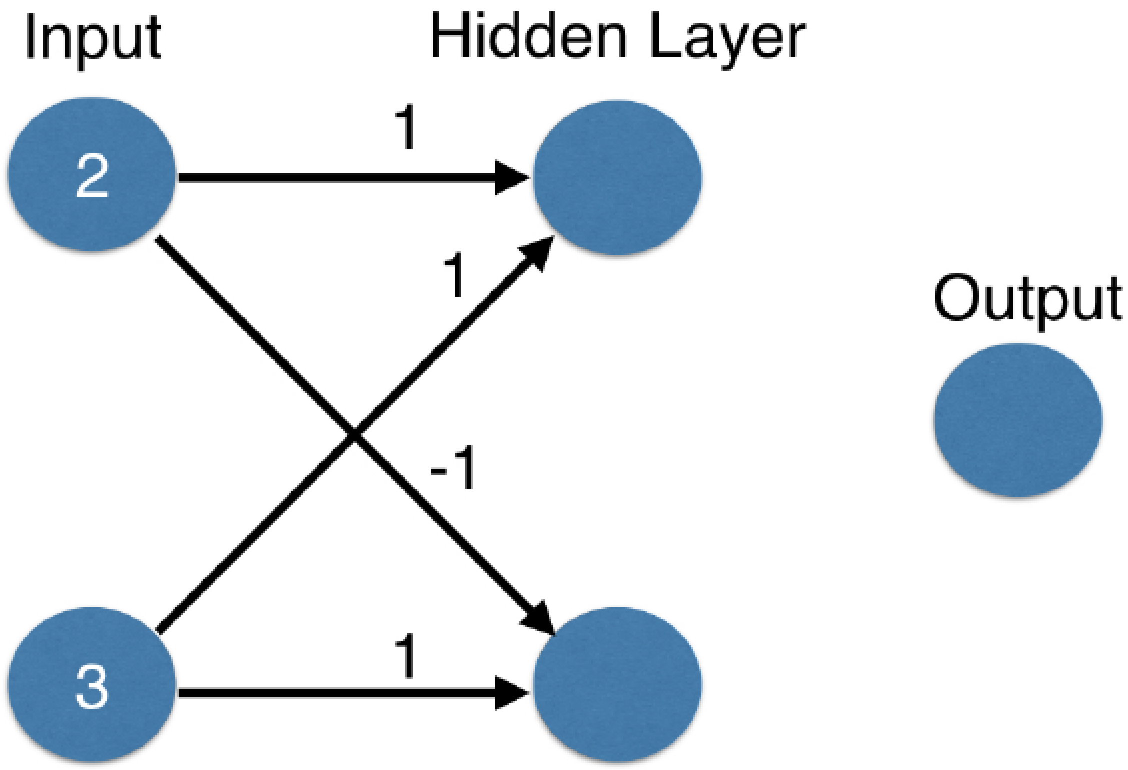
Forward propagation



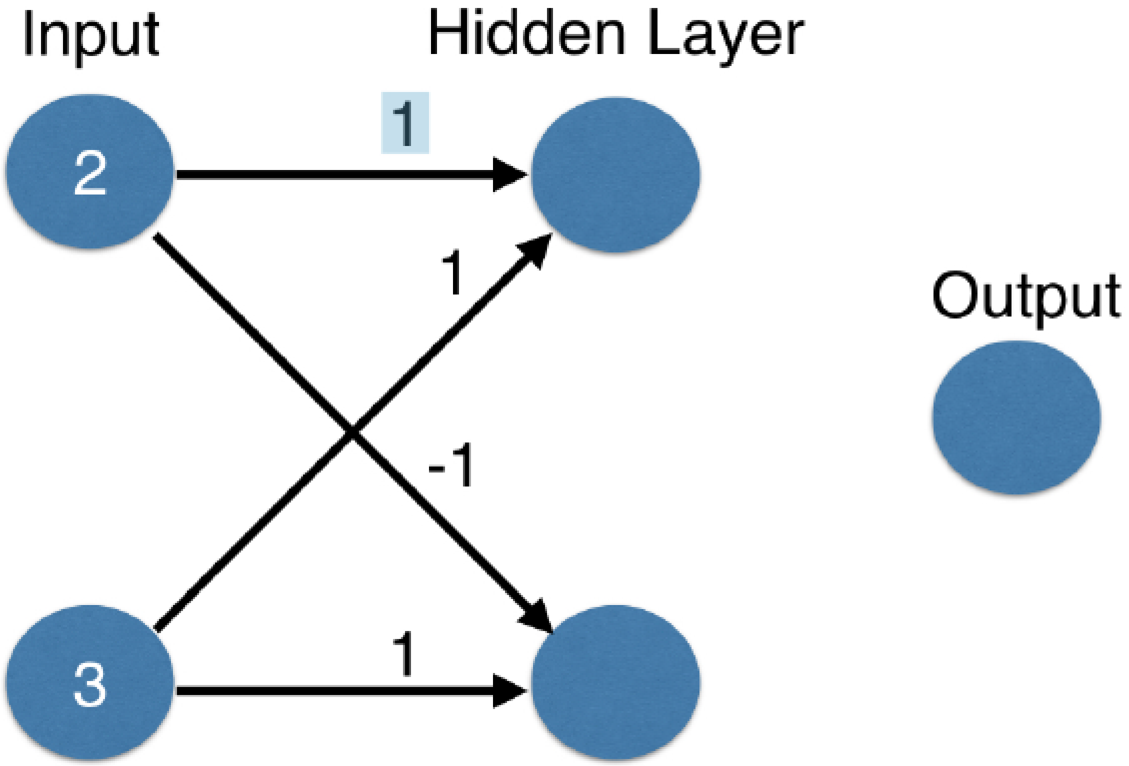
Forward propagation



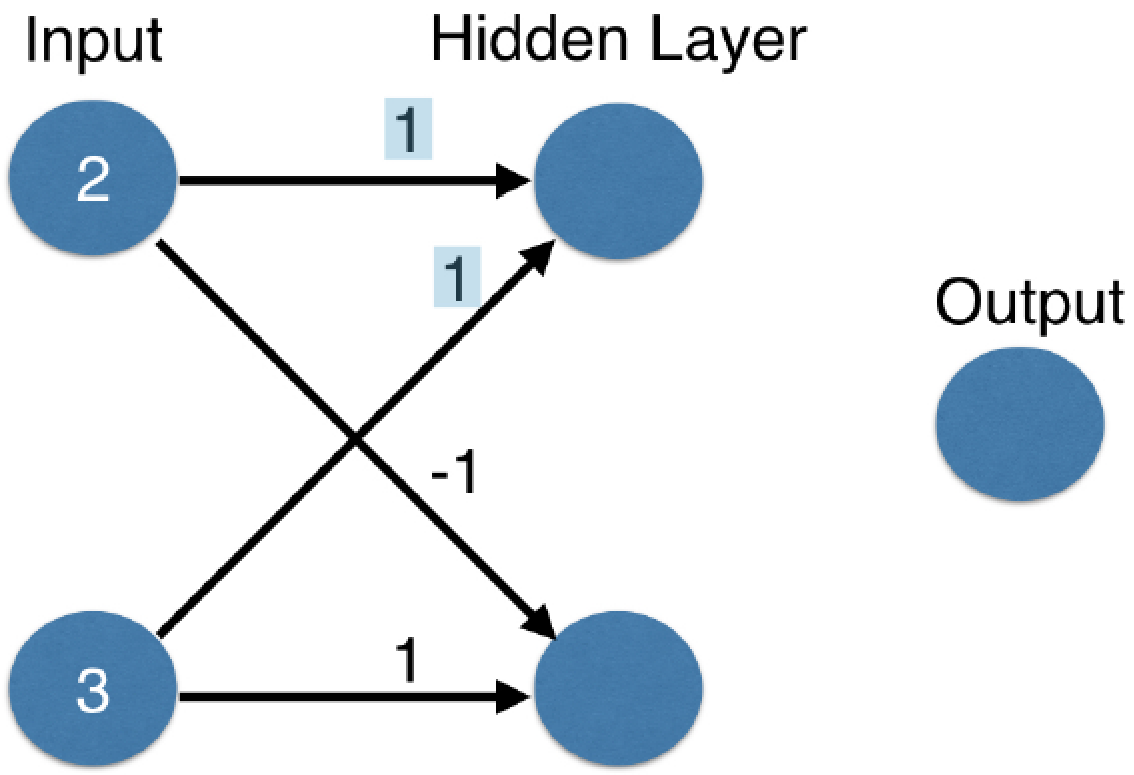
Forward propagation



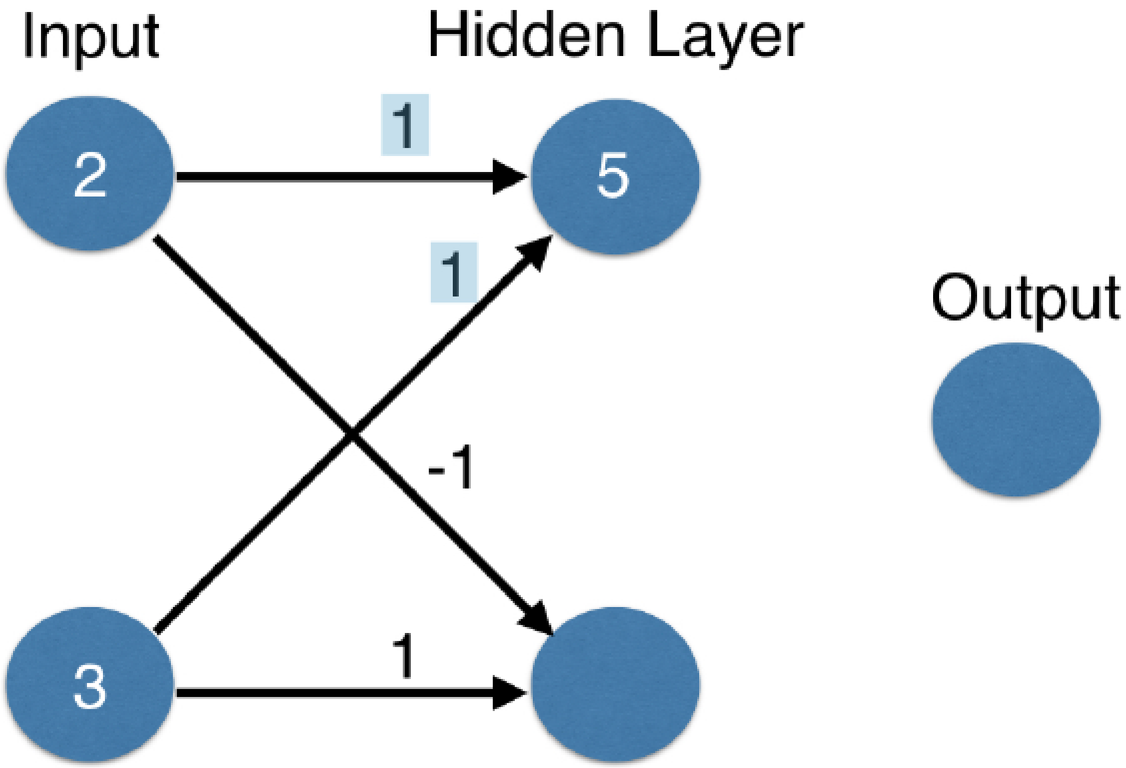
Forward propagation



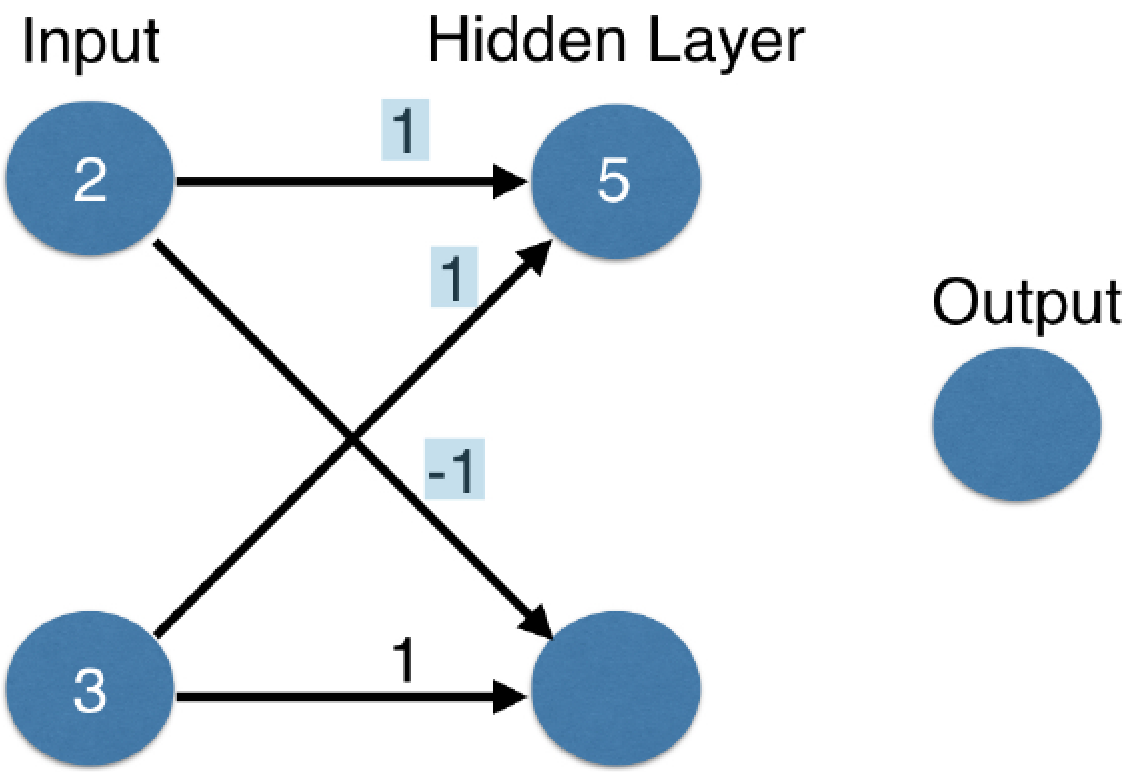
Forward propagation



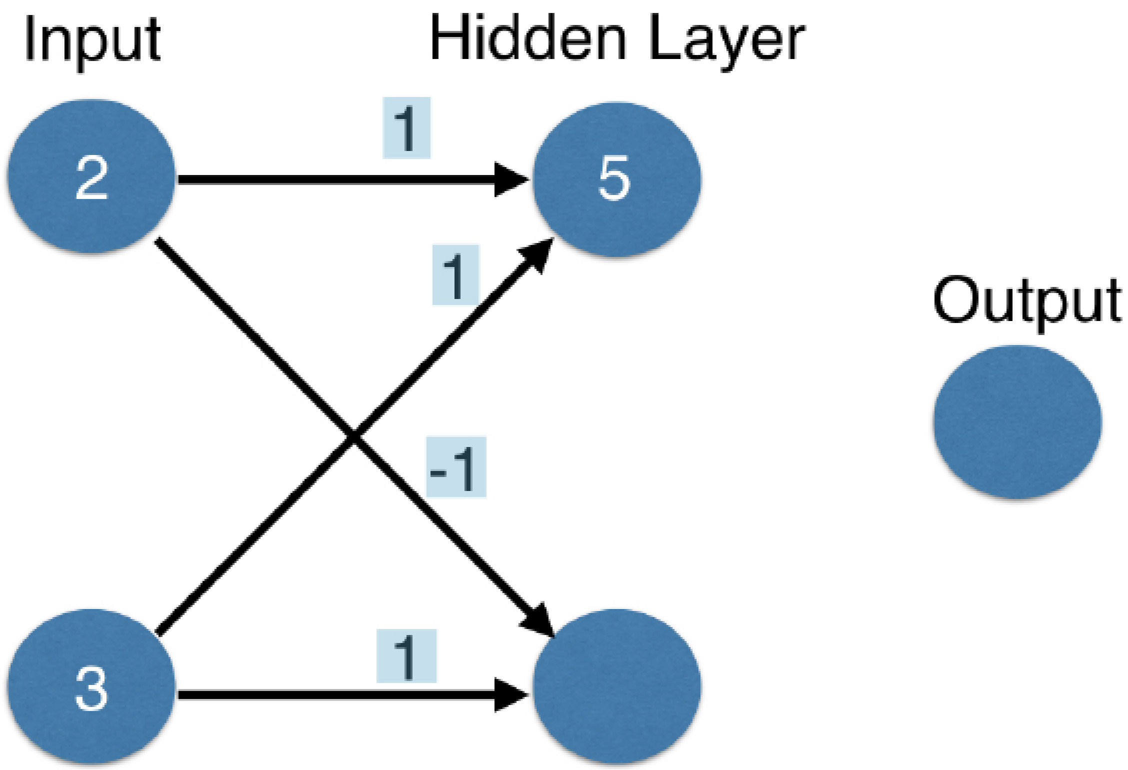
Forward propagation



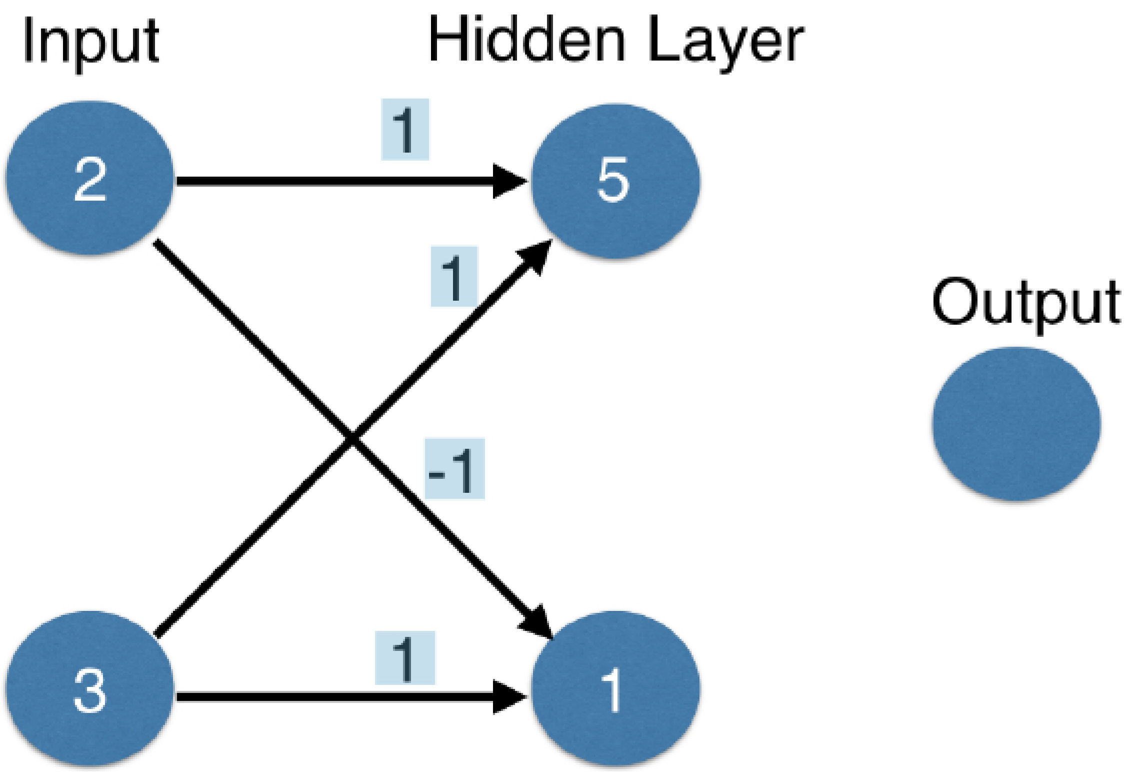
Forward propagation



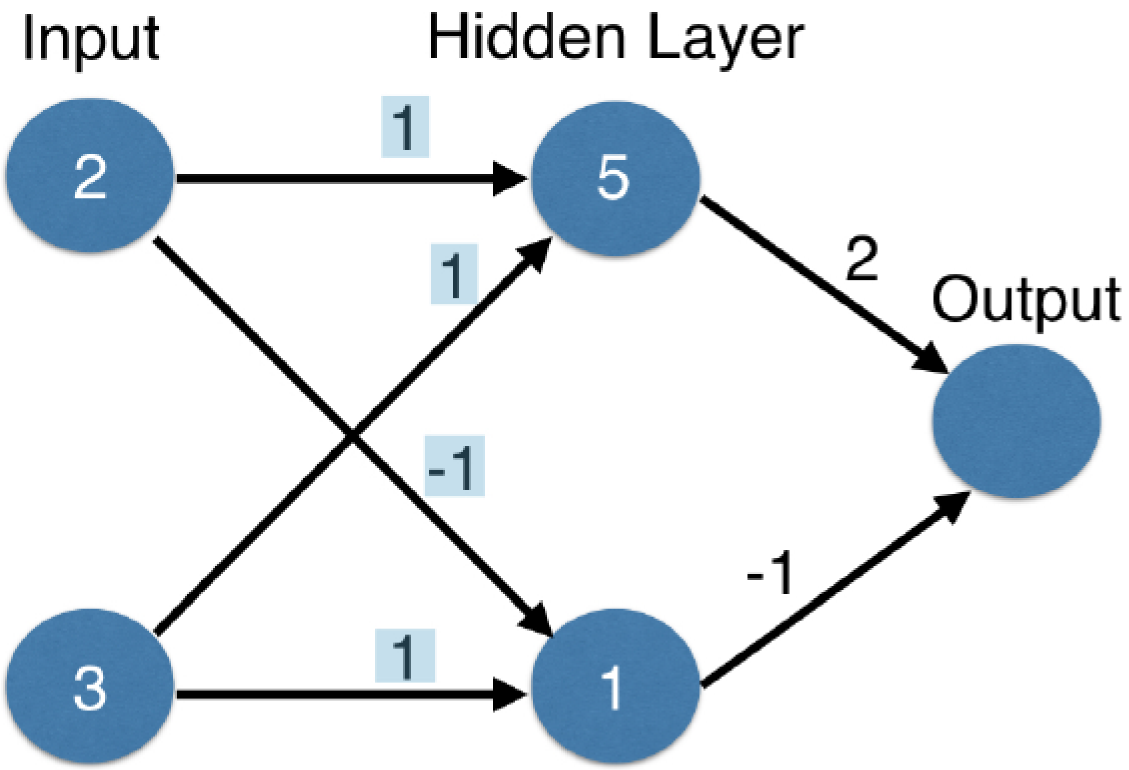
Forward propagation



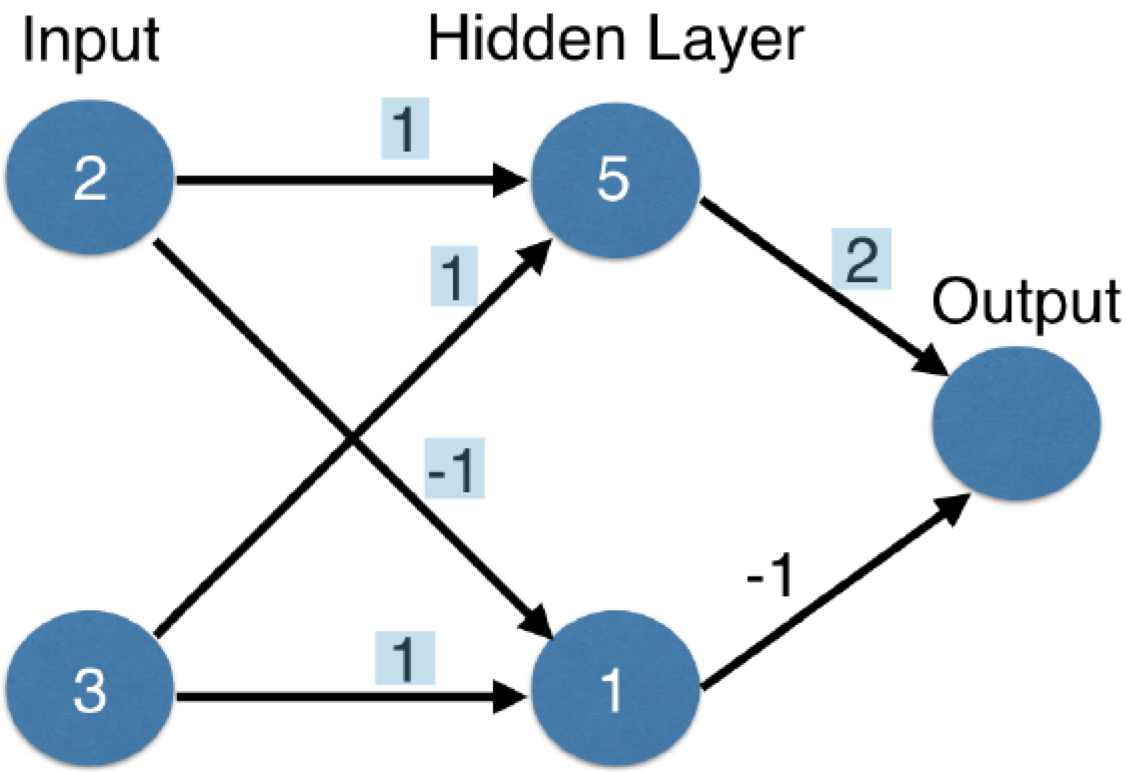
Forward propagation



Forward propagation

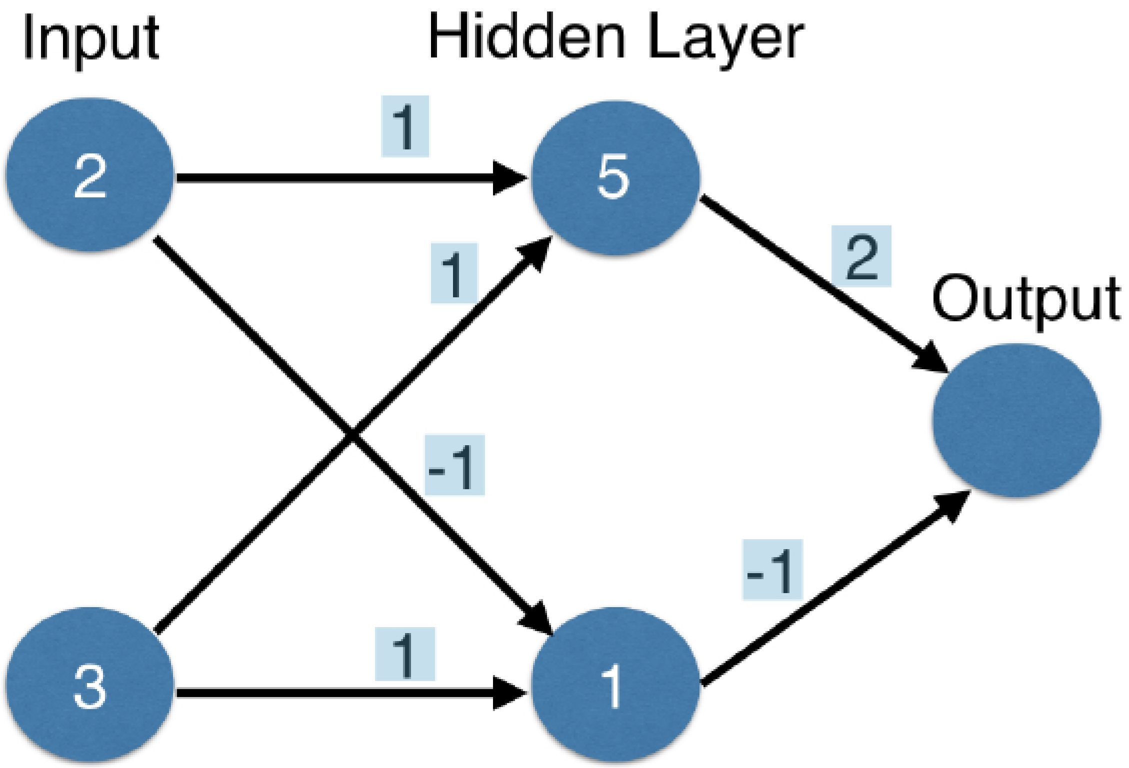


Forward propagation

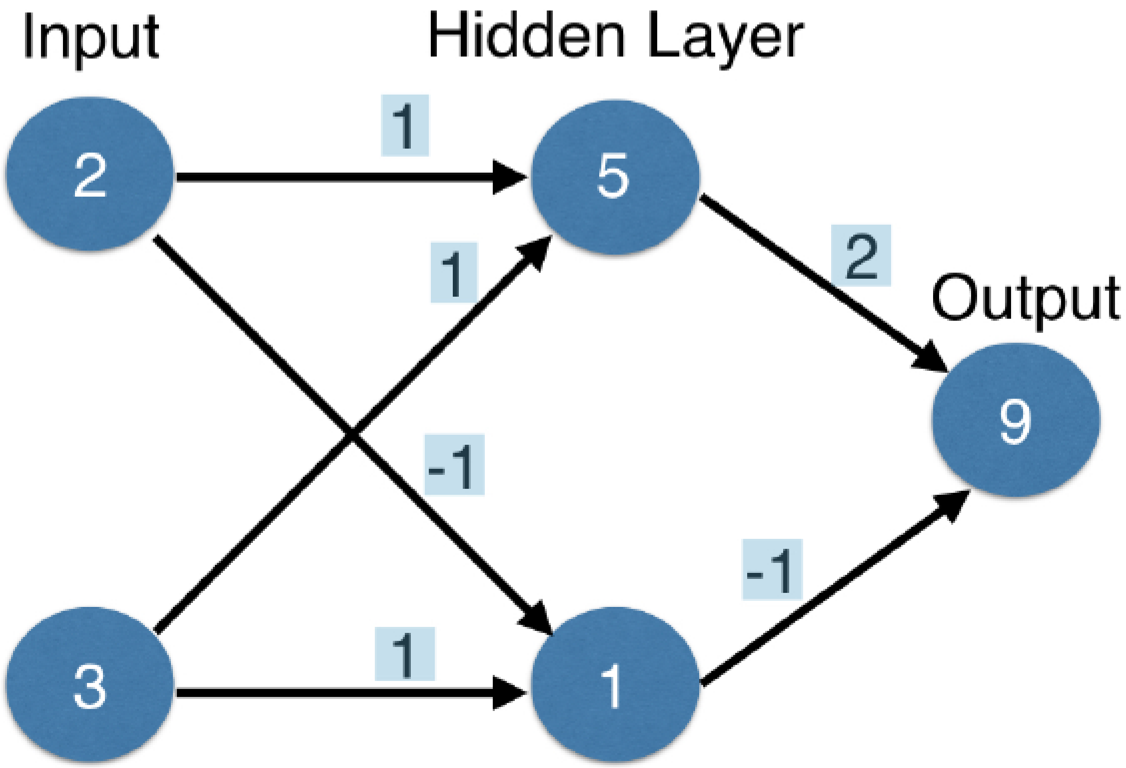


Doğuş Technology

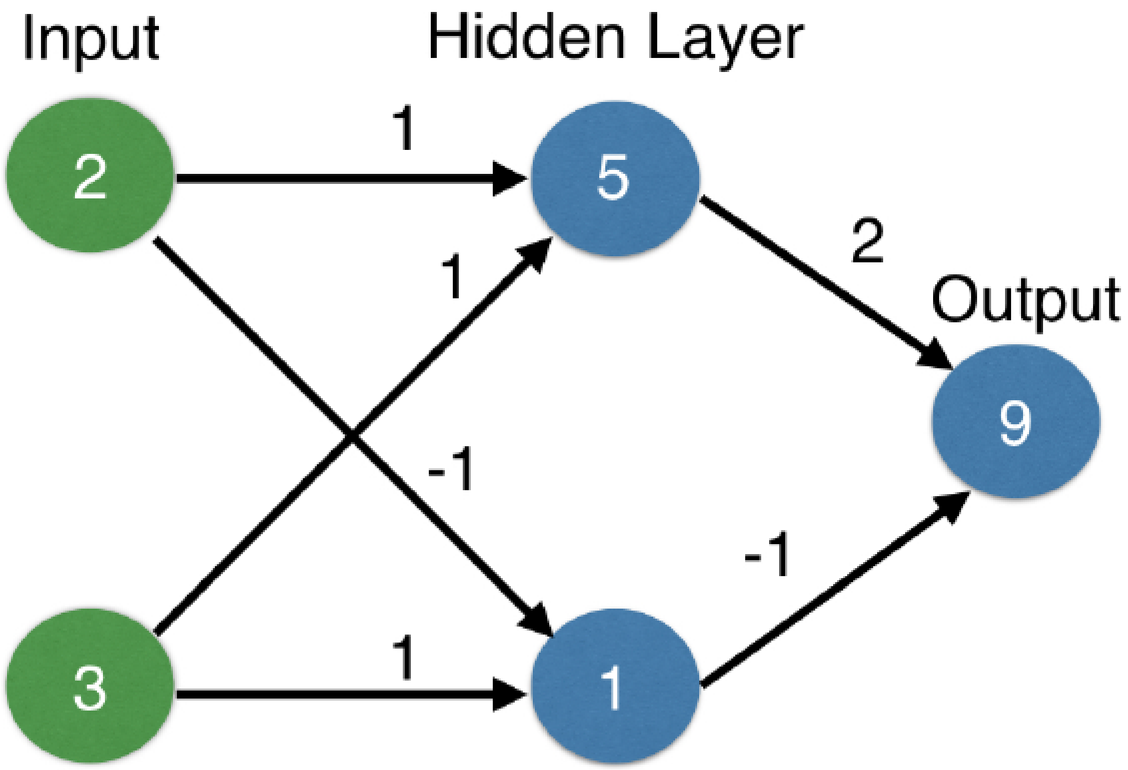
Forward propagation



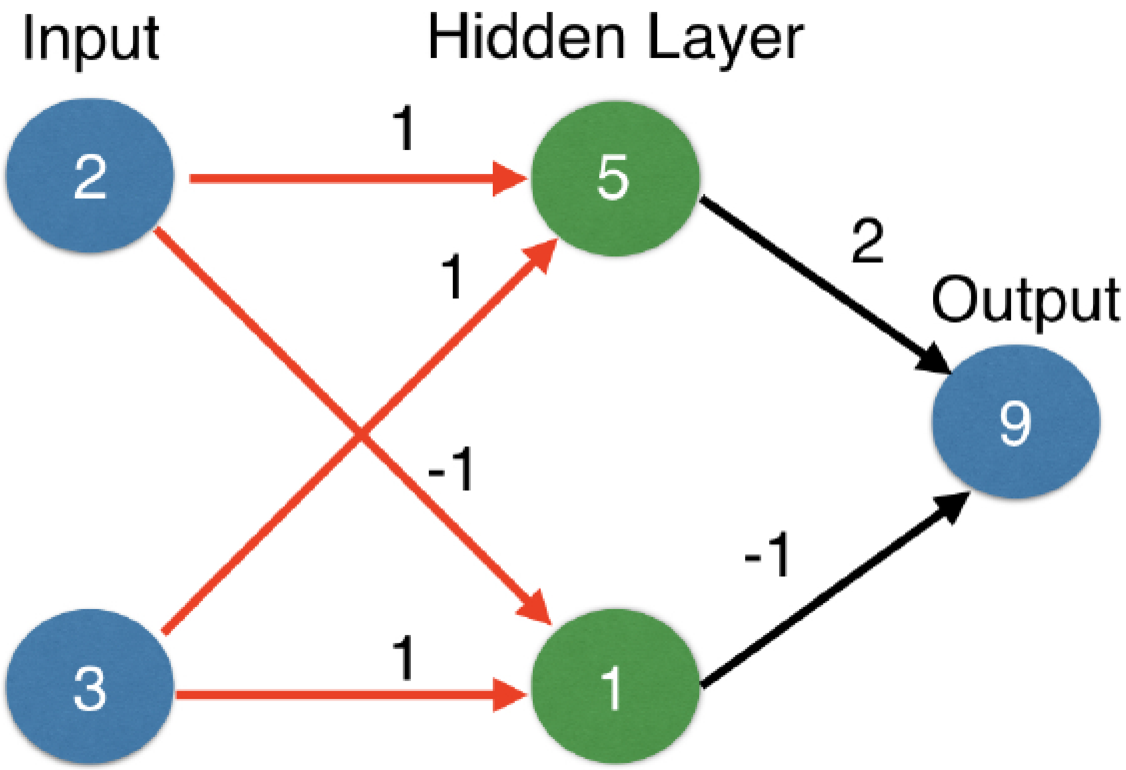
Forward propagation



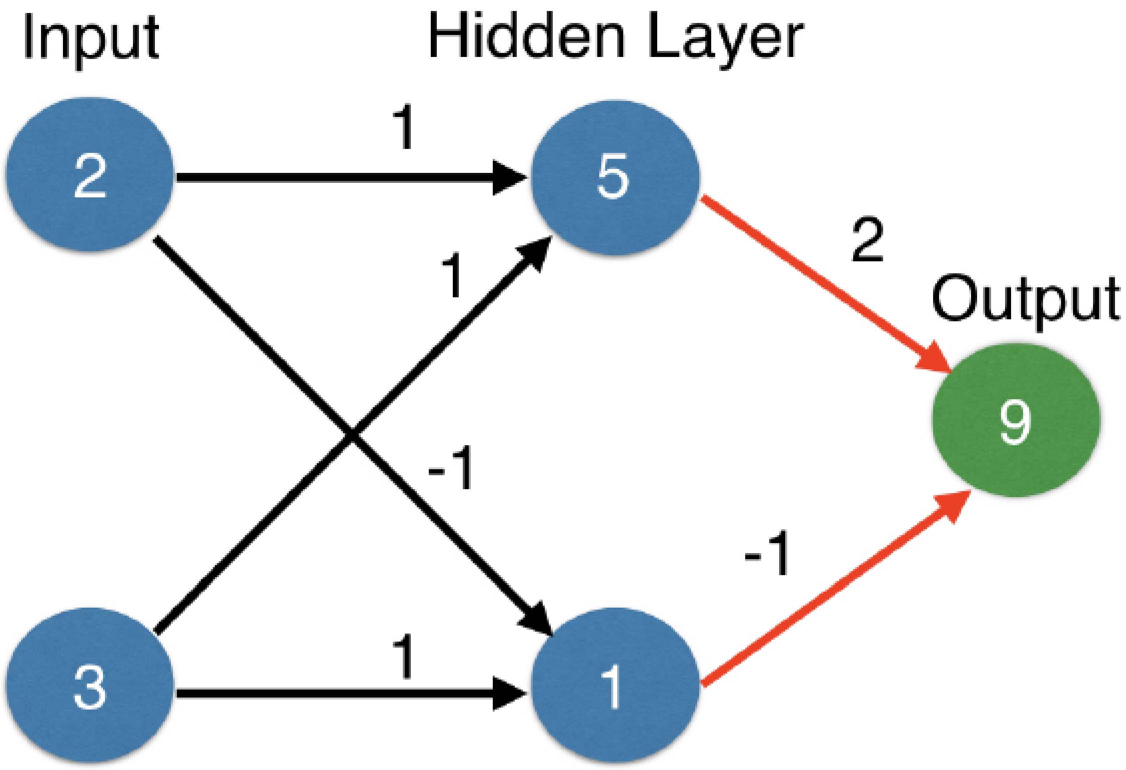
Forward propagation



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

```
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

```
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
```

OUTLINE

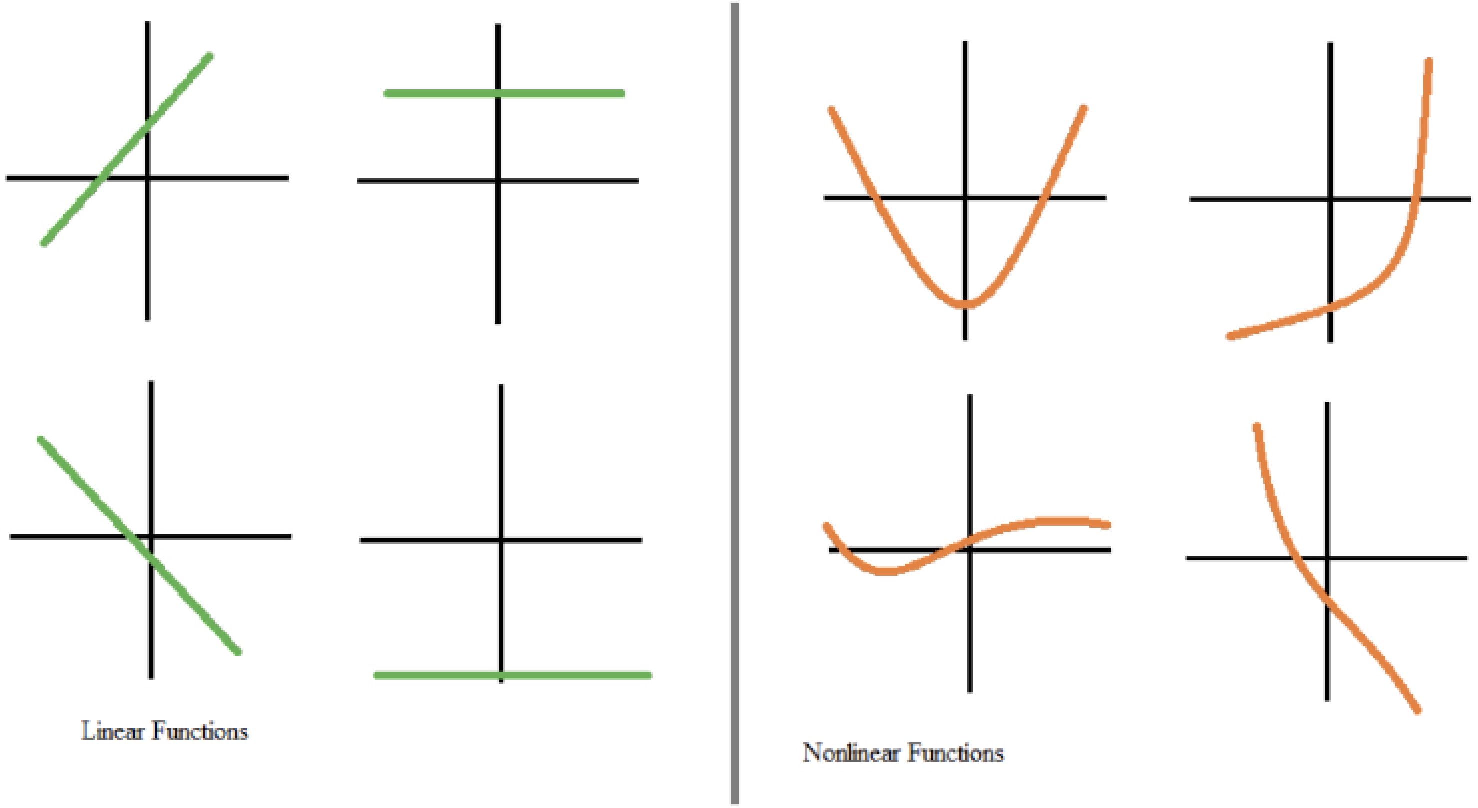
Introductio
n to Deep
Learning

Forward
Propagation

Deeper
Networks

Deep
Learning
Models
with Keras

Activation functions

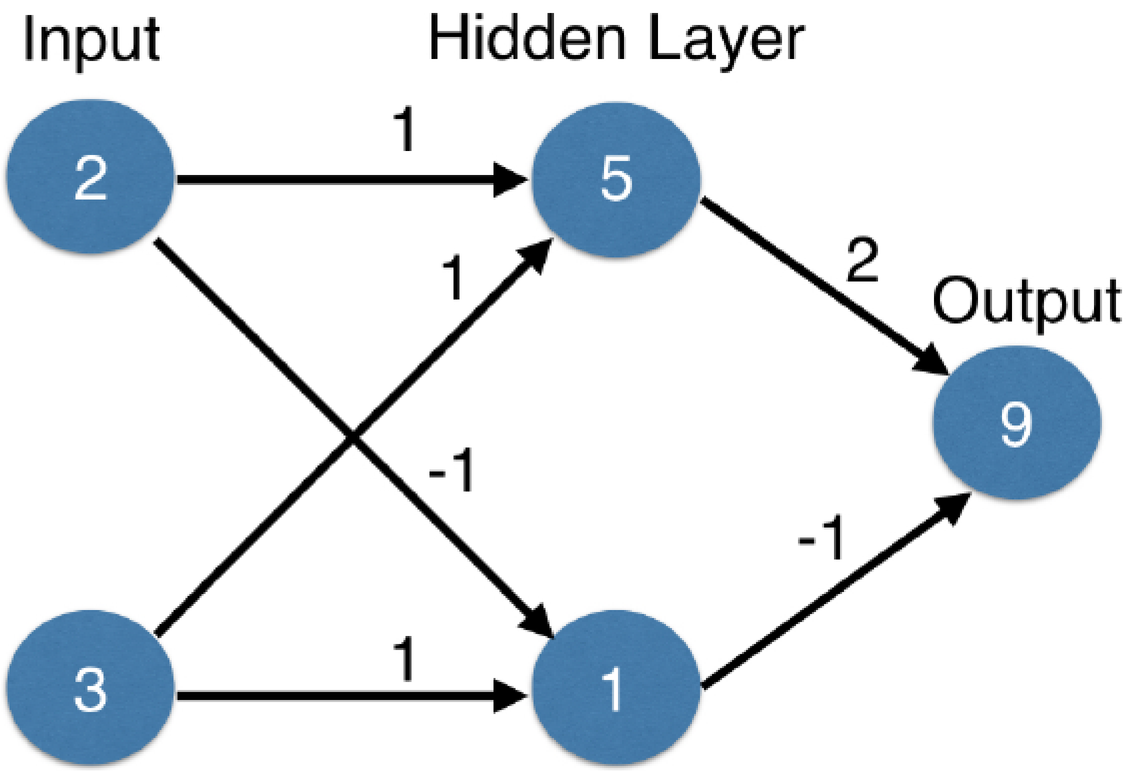


Doğuş Technology

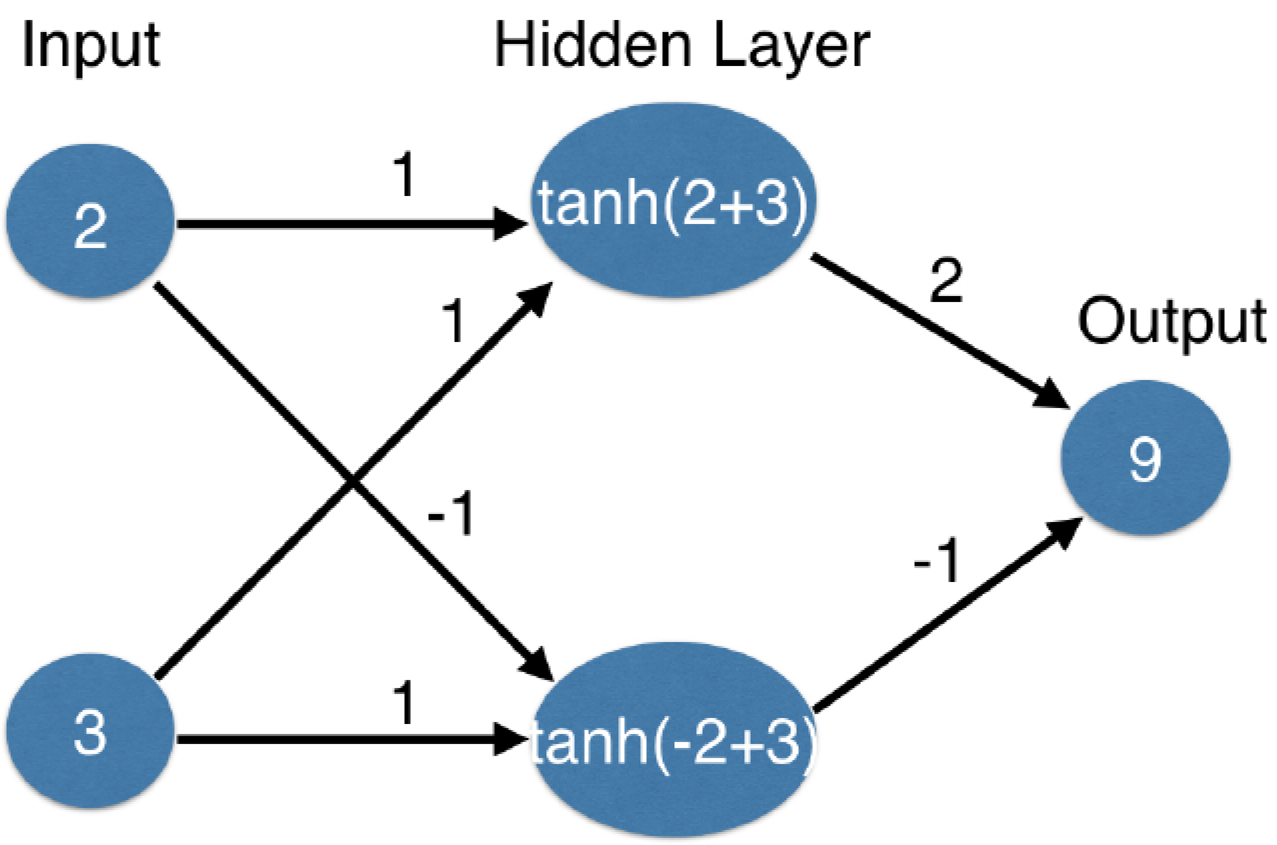
Activation functions

- Applied to node inputs to produce node output

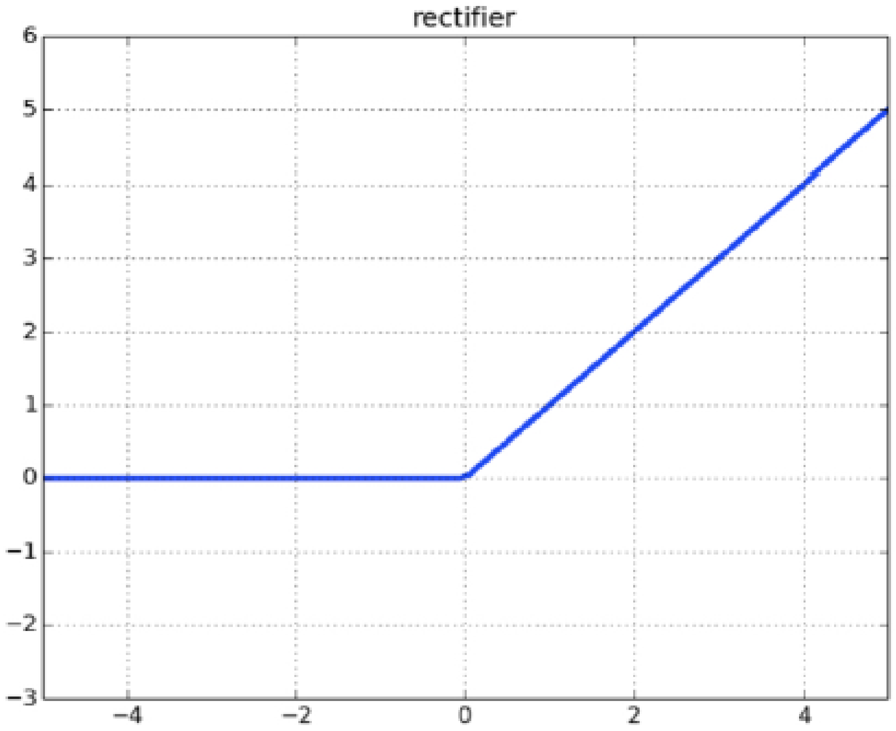
Improving our neural network



Activation functions



ReLU (Rectified Linear Activation)



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

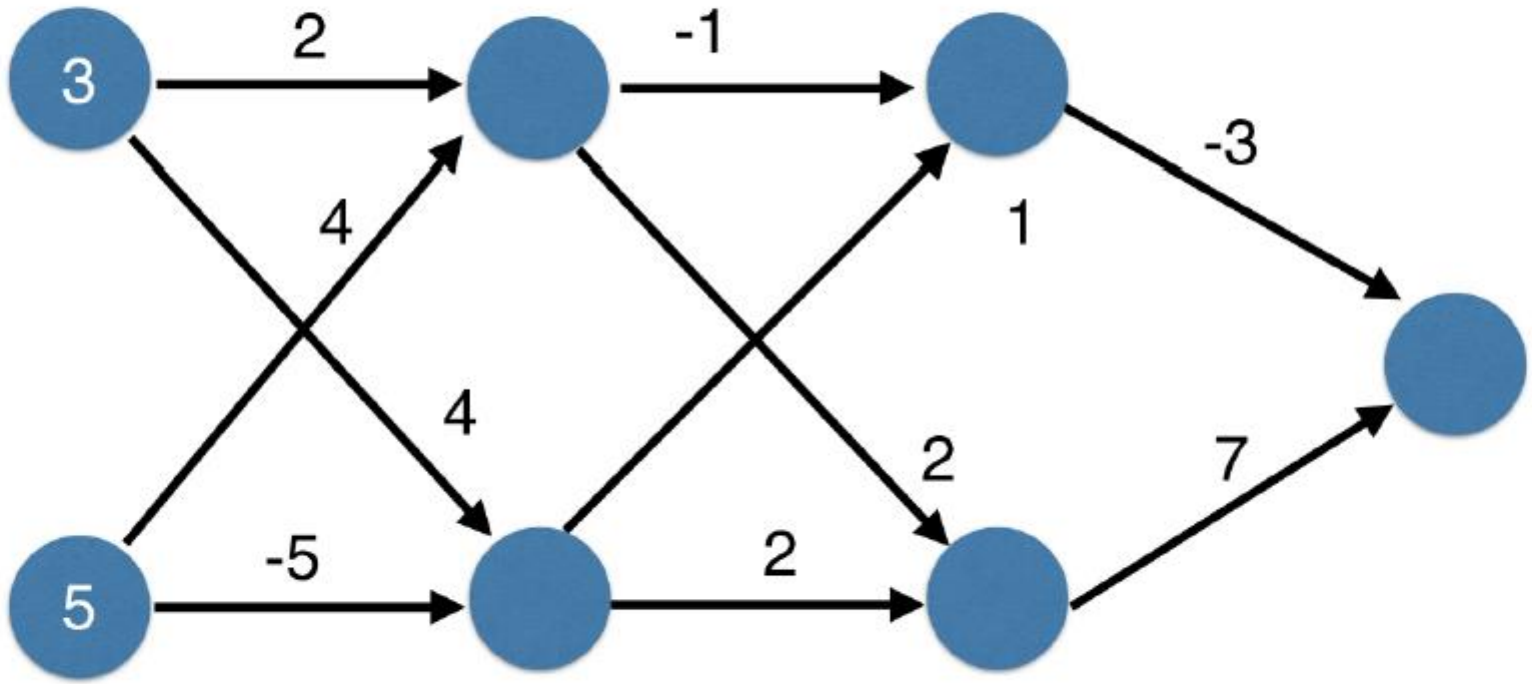
Activation functions

```python
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_output * weights['output']).sum()
```

```python
print(output)
```

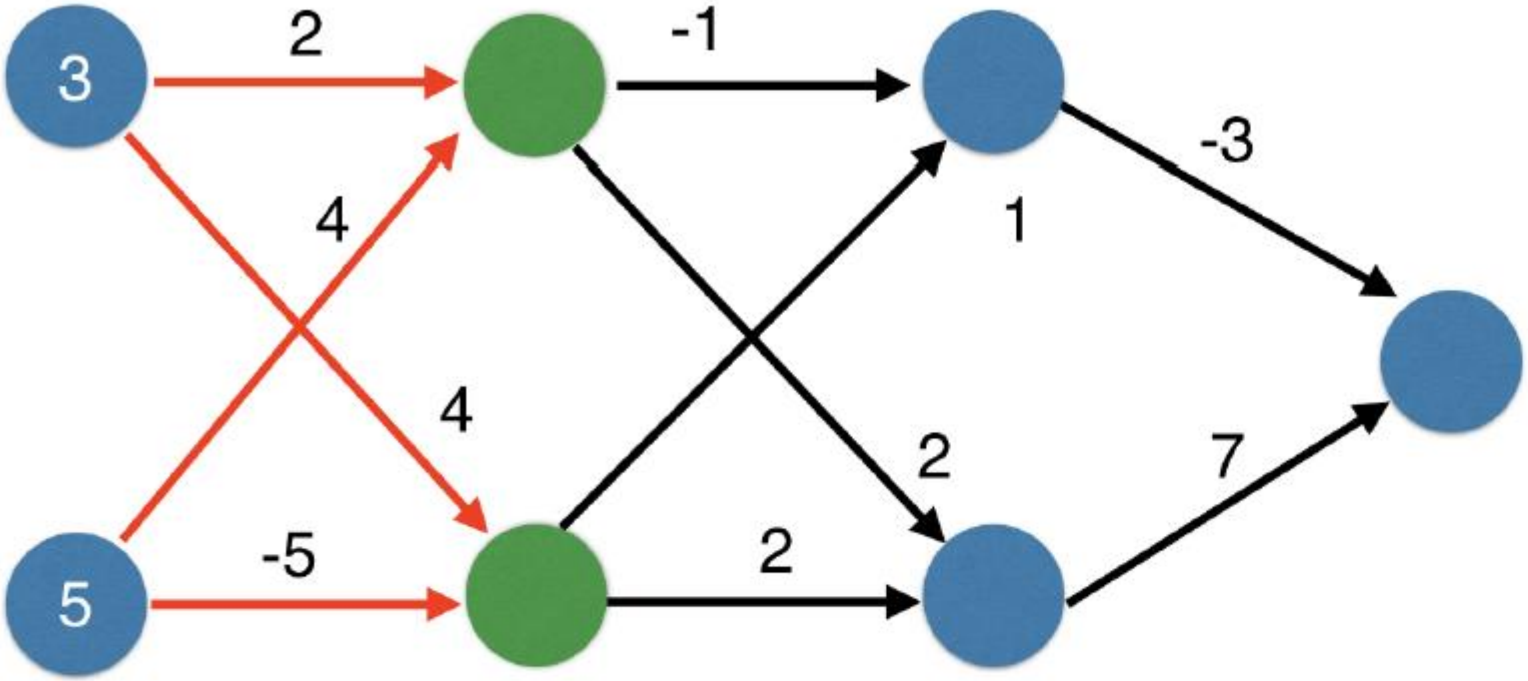
```
1.2382242525694254
```

Deeper Networks



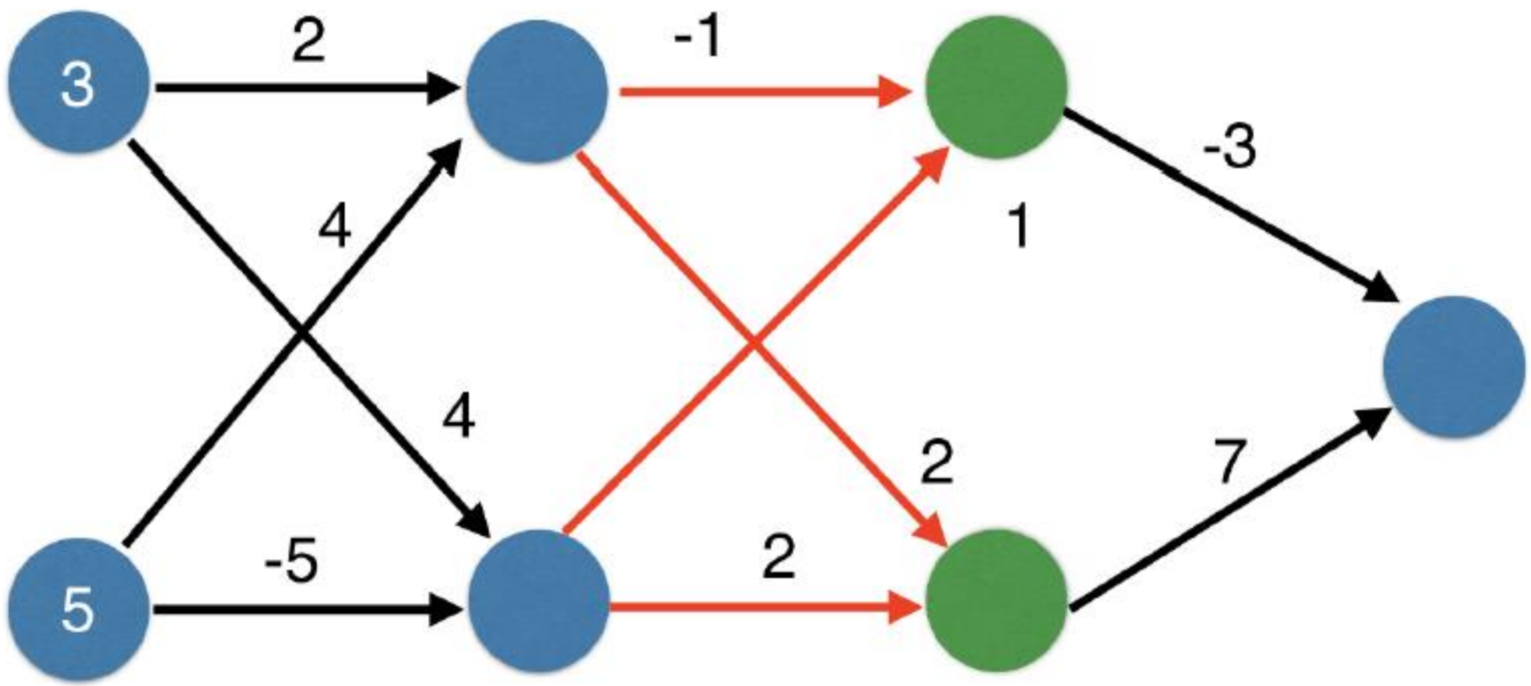
Calculate with ReLU Activation Function

Deeper Networks



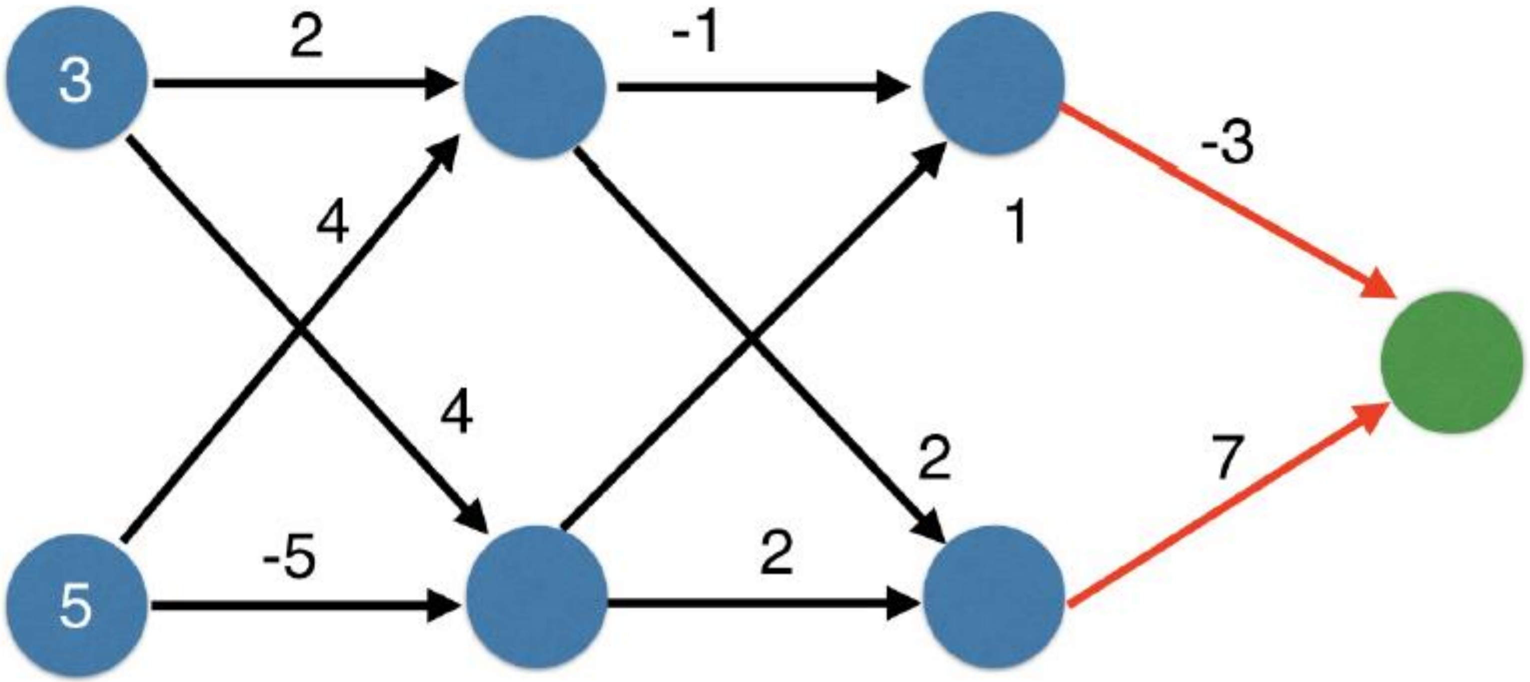
Calculate with ReLU Activation Function

Deeper Networks



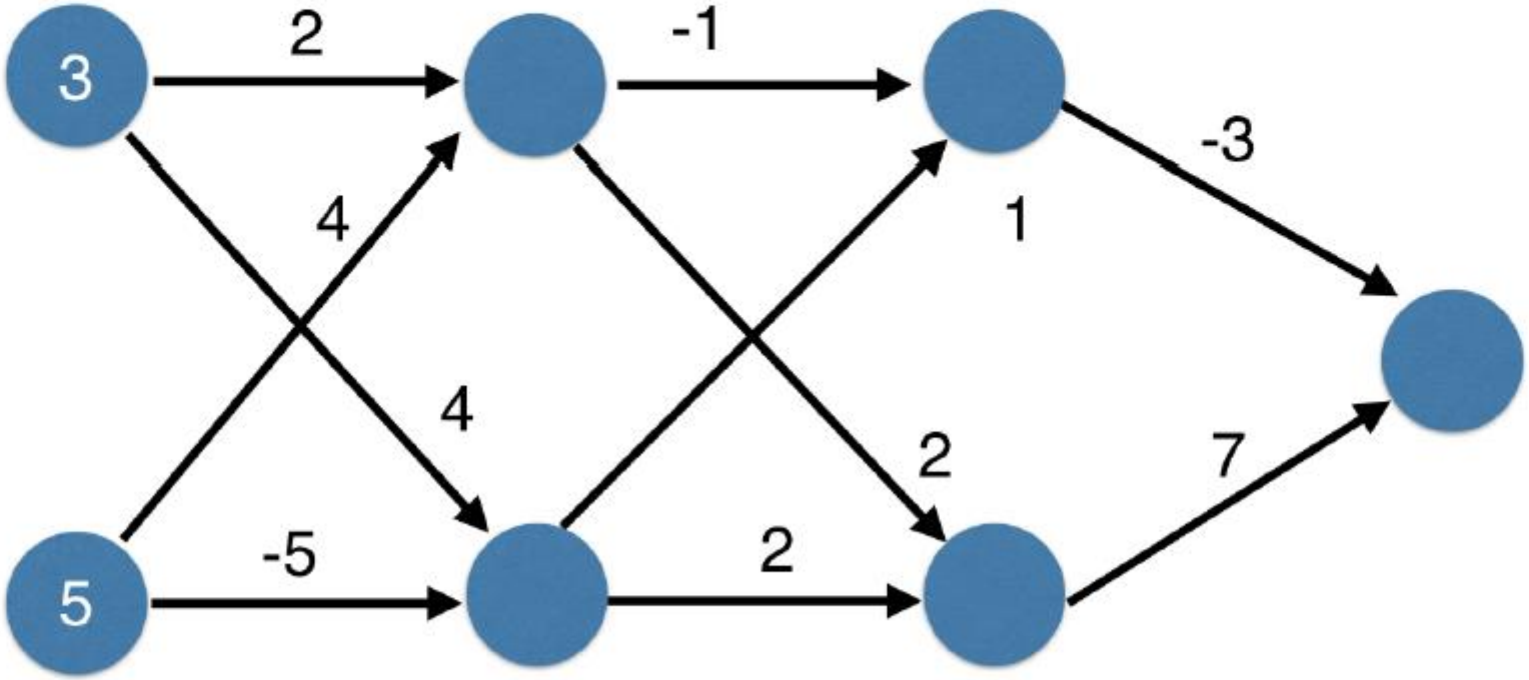
Calculate with ReLU Activation Function

Deeper Networks



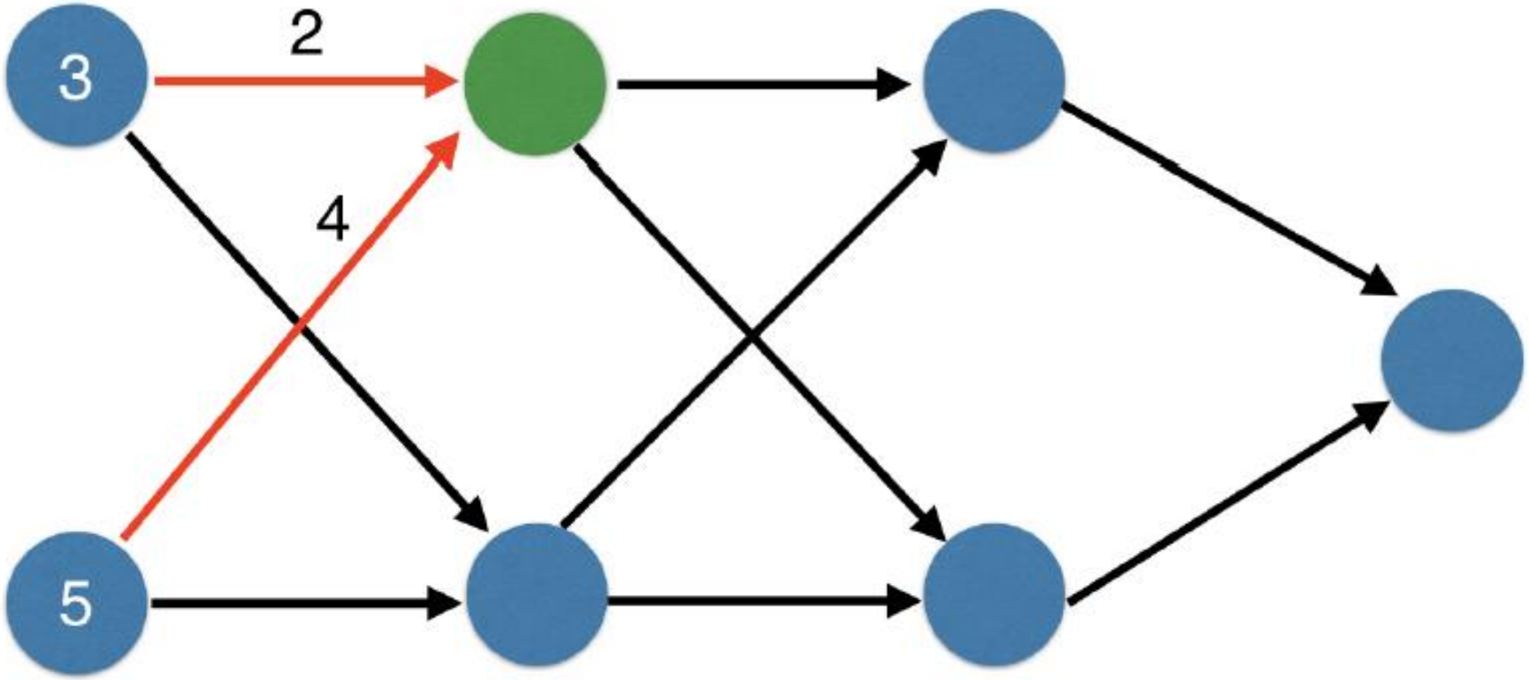
Calculate with ReLU Activation Function

Deeper Networks



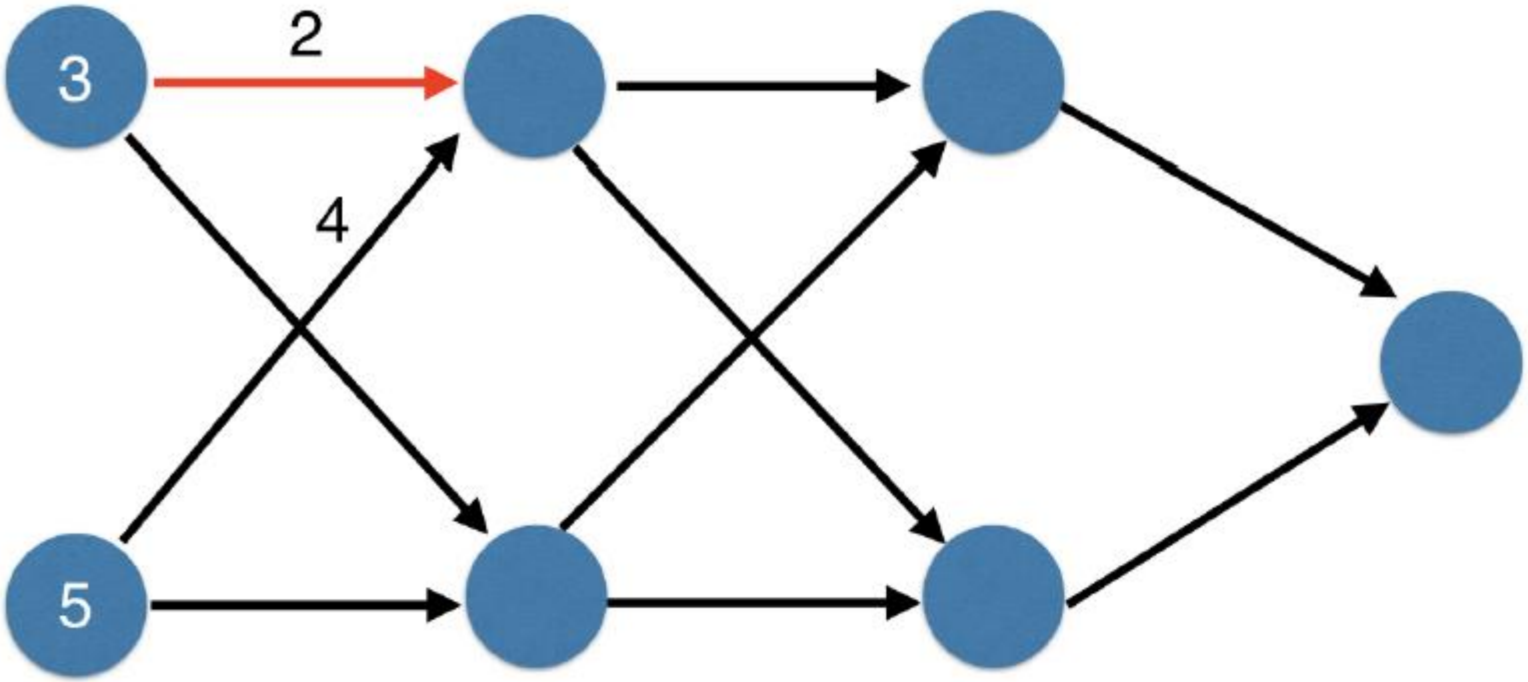
Calculate with ReLU Activation Function

Deeper Networks

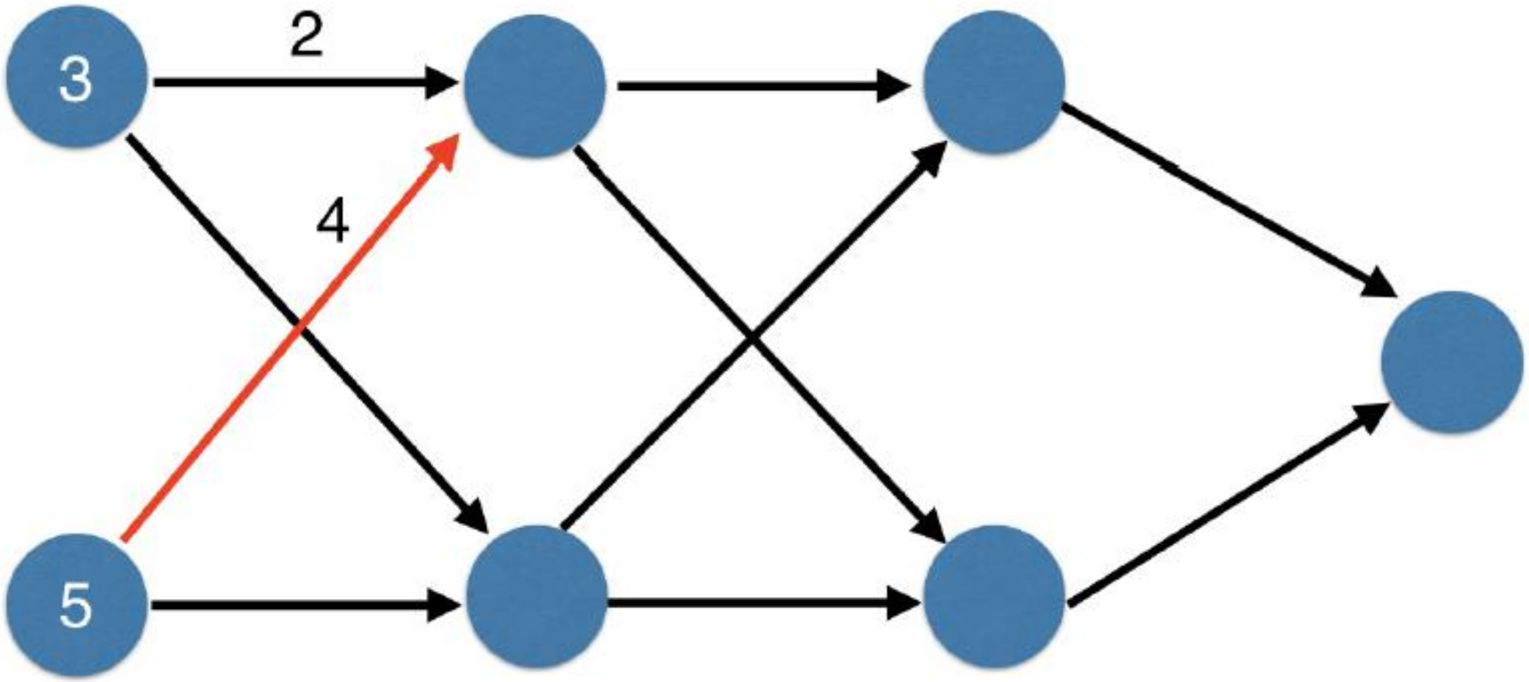


Calculate with ReLU Activation Function

Deeper Networks

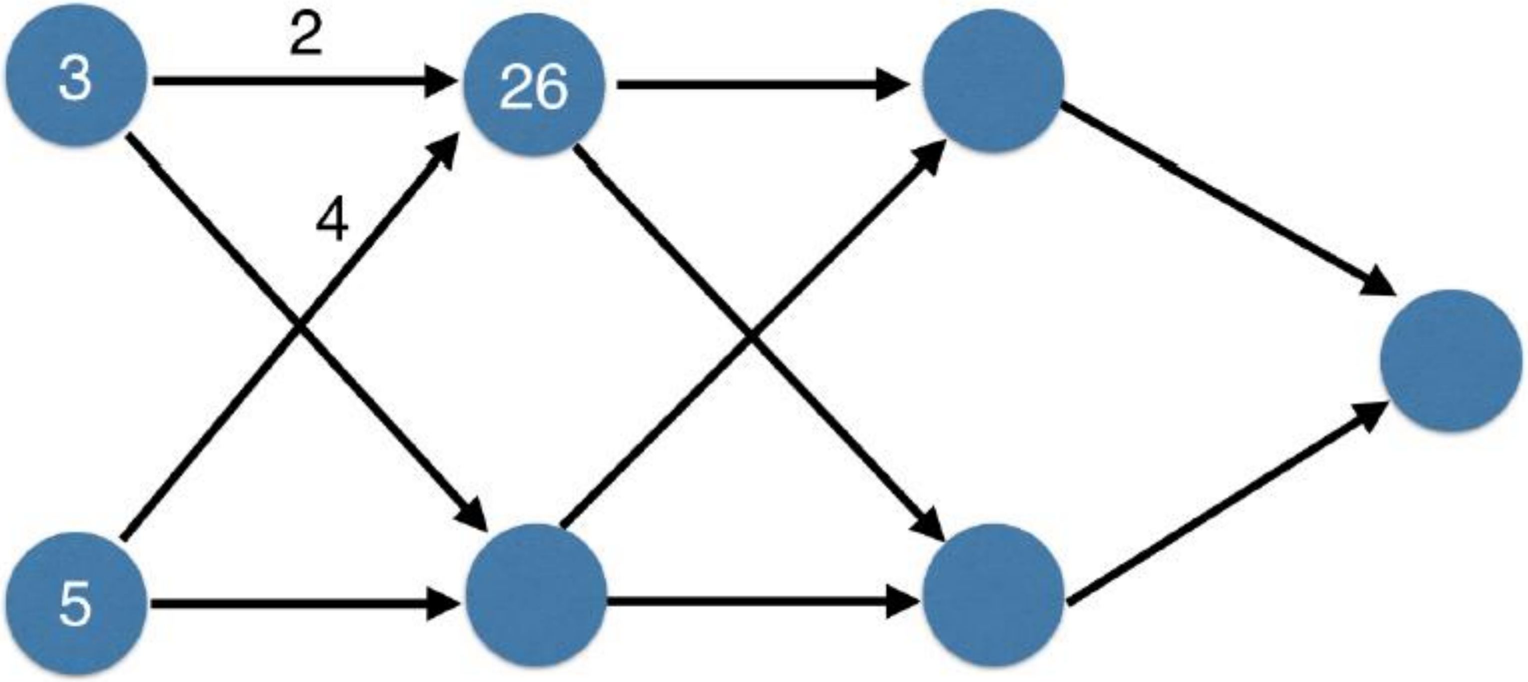


Calculate with ReLU Activation Function



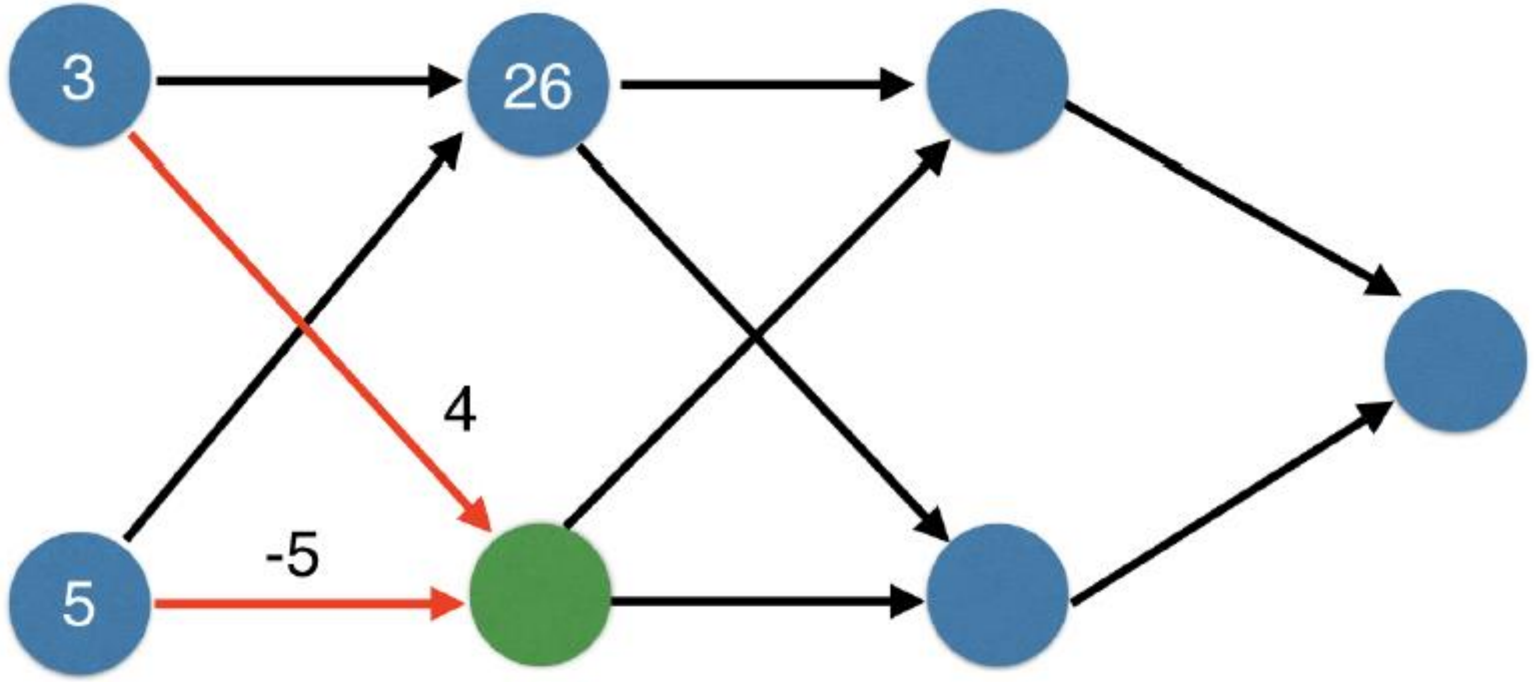
Calculate with ReLU Activation Function

Deeper Networks

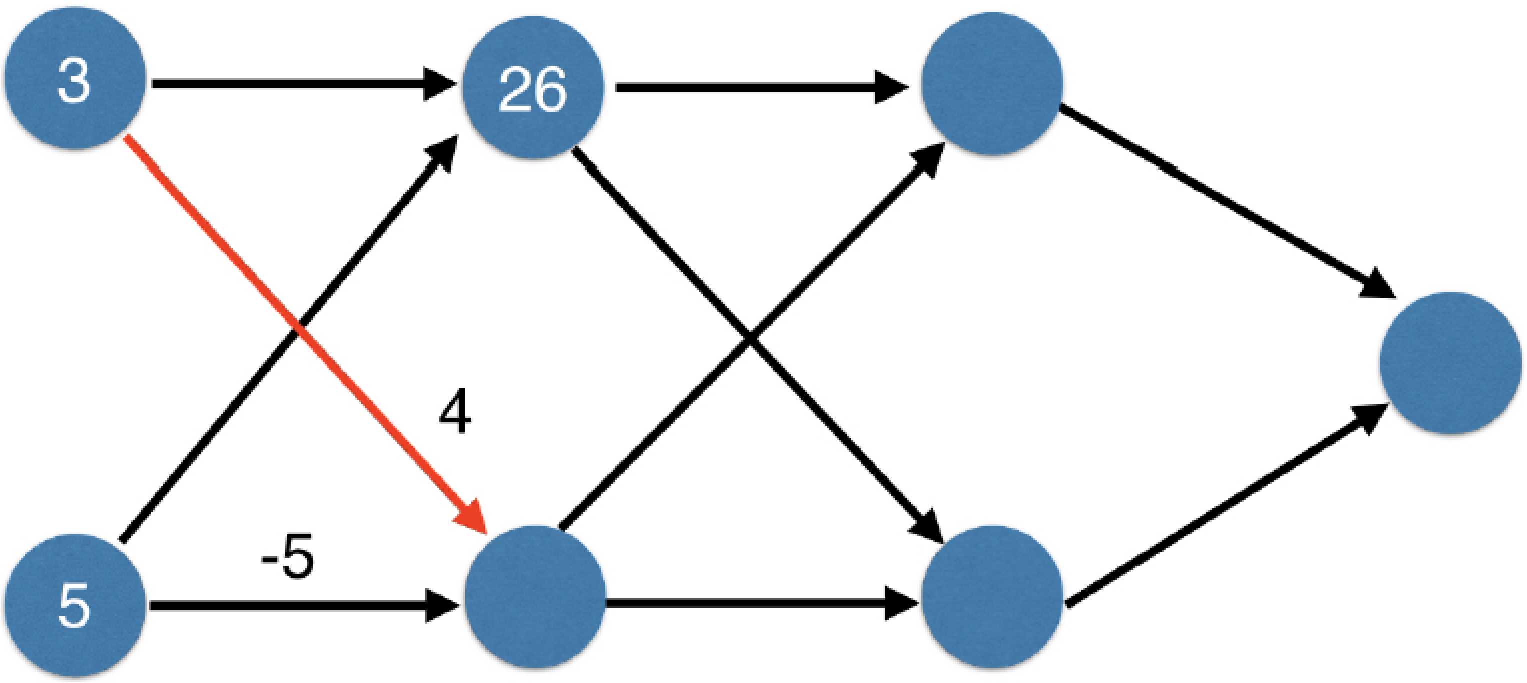


Calculate with ReLU Activation Function

Deeper Networks

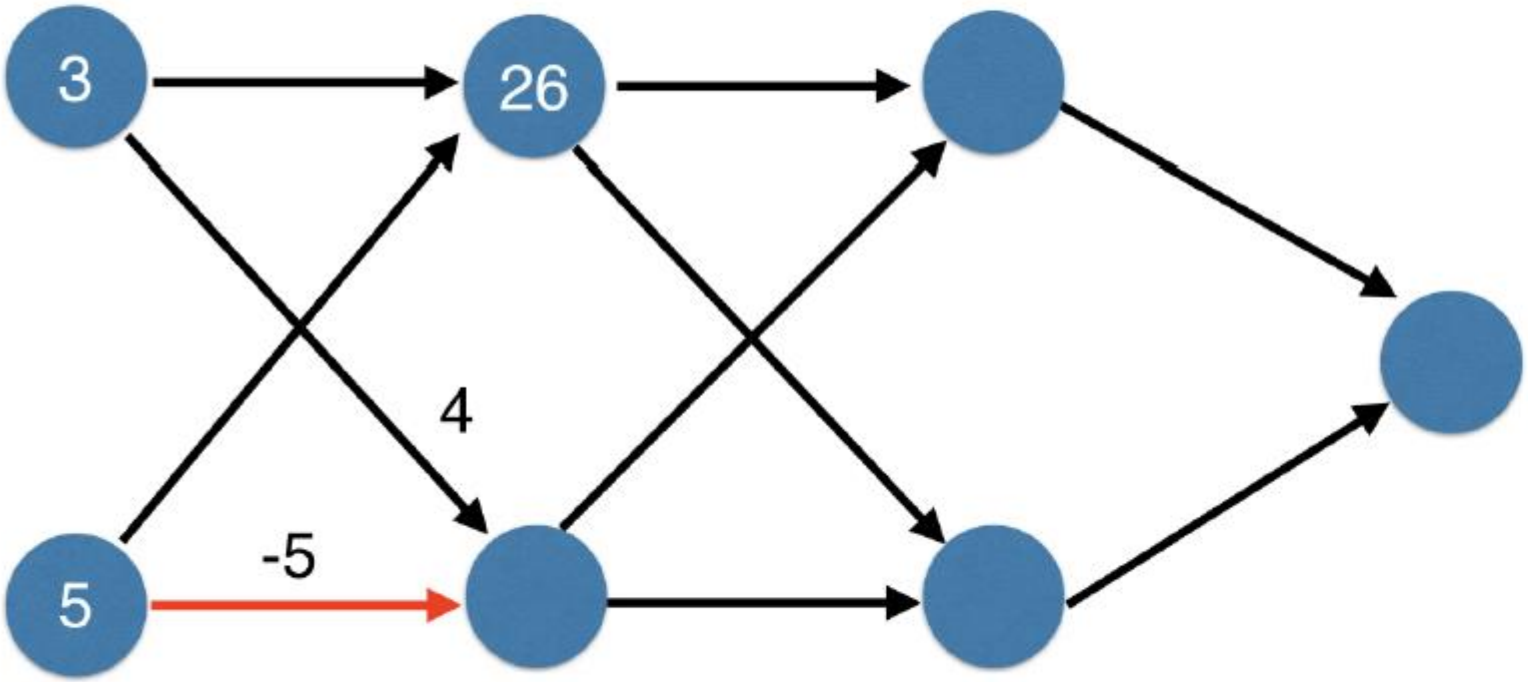


Calculate with ReLU Activation Function



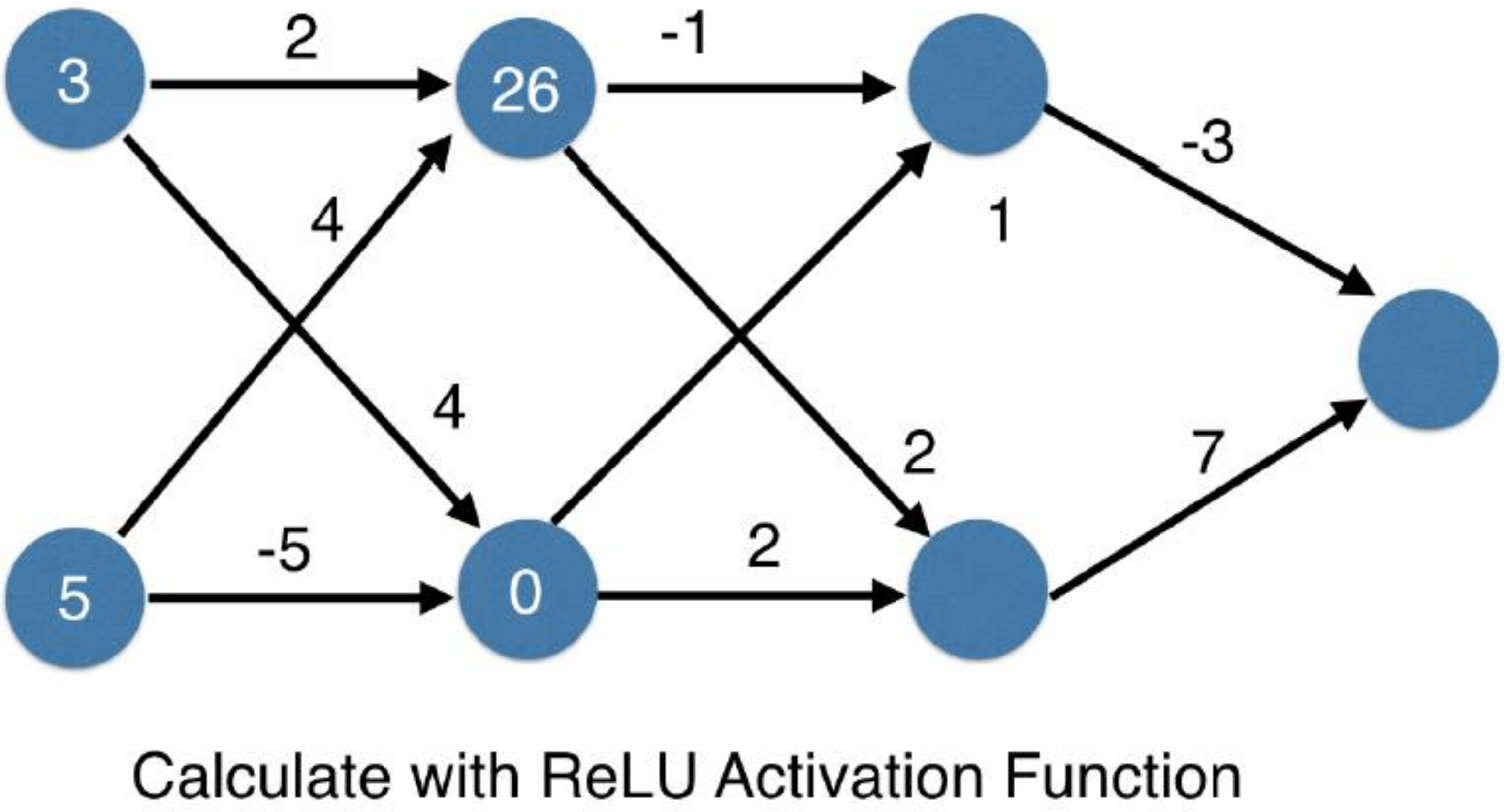
Calculate with ReLU Activation Function

Deeper Networks

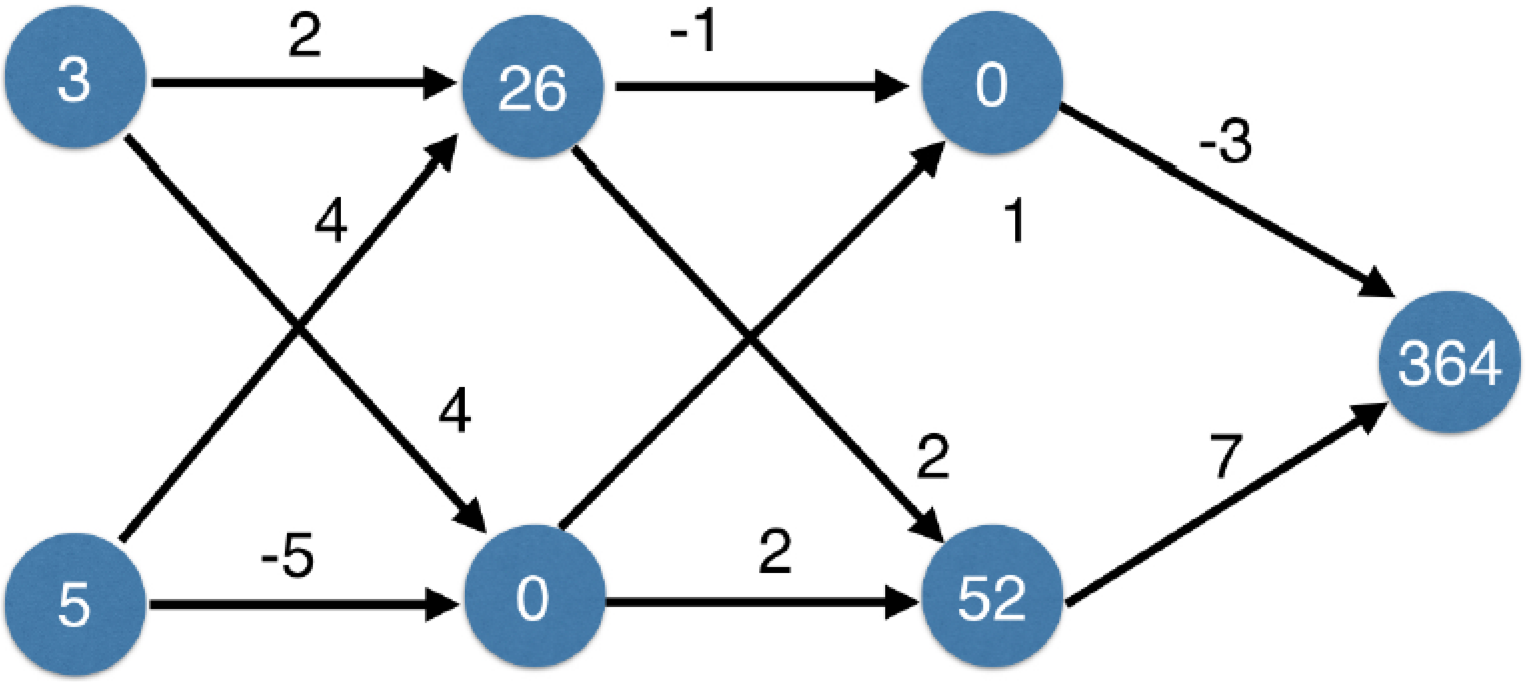


Calculate with ReLU Activation Function

Deeper Networks



Deeper Networks

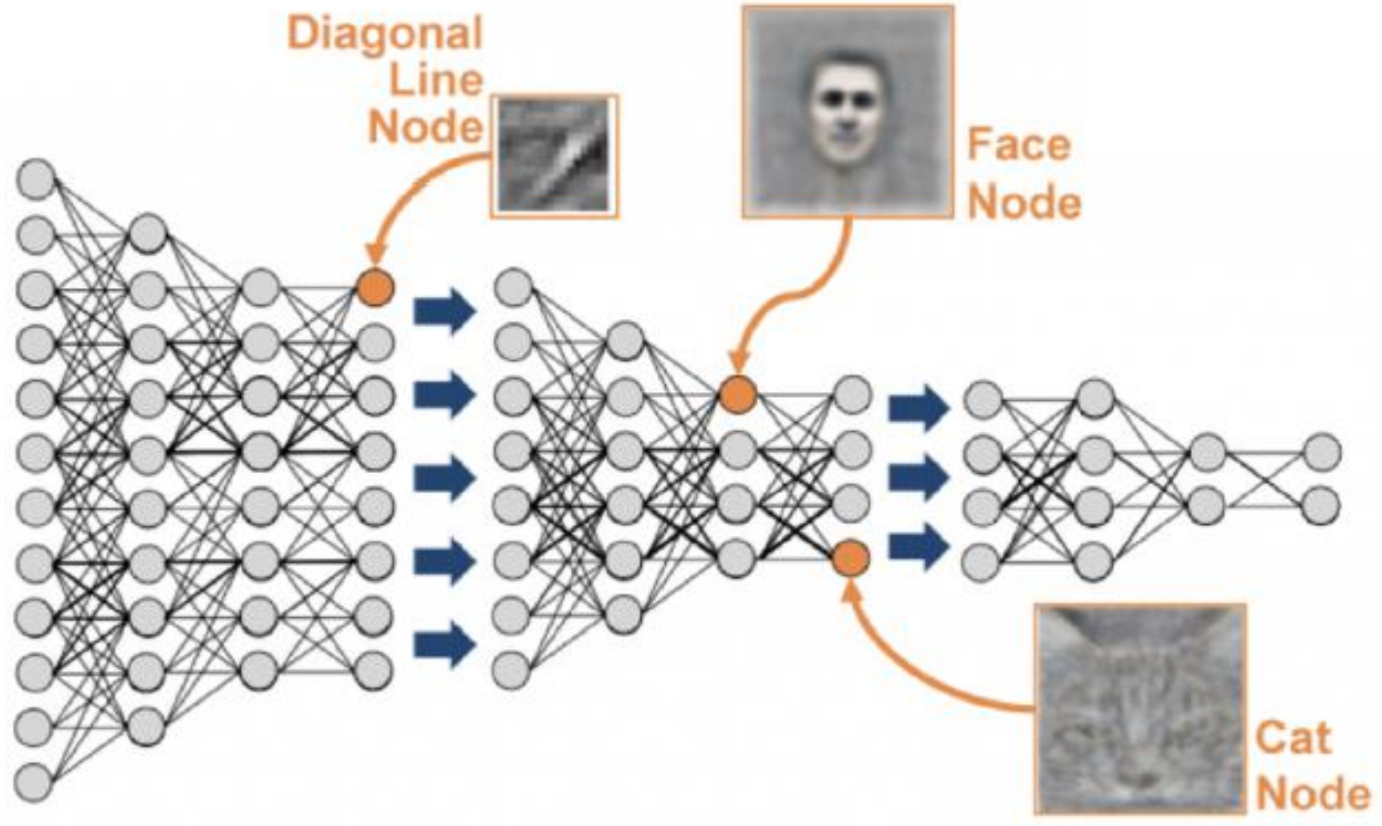


Calculate with ReLU Activation Function

Representation learning

- Deep networks internally build representations of patterns in 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

OUTLINE

Introductio n to Deep Learning

Forward Propagation

Deeper Networks

Deep Learning Models with Keras

Model building steps

- Specify Architecture
- Compile
- Fit
- Predict

Model specification

```
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))
```

Why you need to compile your model

- Specify the optimizer
    - Many options and mathematically complex
    - "Adam" is usually a good choice
- Loss function
    - "mean_squared_error" common for regression

Compiling a model

```
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))
model.compile(optimizer='adam', loss='mean_squared_error')
```

What is fitting a model

- Applying backpropagation and gradient descent with your data to update the weights
- Scaling data before fitting can ease optimization

Fitting a model

```
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))
model.compile(optimizer='adam', loss='mean_squared_error')
model.fit(predictors, target)
```

Classification models

- 'categorical_crossentropy' loss function
- Similar to log loss: Lower is better
- Add metrics = ['accuracy'] to compile step for easy-to understand diagnostics
- Output layer has separate node for each possible outcome,

and uses 'softmax' activation

Quick look at the data

shot_clock	dribbles	touch_time	shot_dis	close_def_dis	shot_result
10.8	2	1.9	7.7	1.3	1
3.4	0	0.8	28.2	6.1	0
0	3	2.7	10.1	0.9	0
10.3	2	1.9	17.2	3.4	0

Quick look at the data

shot_clock	dribbles	touch_time	shot_dis	close_def_dis	shot_result
10.8	2	1.9	7.7	1.3	1
3.4	0	0.8	28.2	6.1	0
0	3	2.7	10.1	0.9	0
10.3	2	1.9	17.2	3.4	0

Transforming to categorical

shot_result
1
0
0
0

Outcome 0	Outcome 1
0	1
1	0
1	0
1	0

Classification models

```
from keras.utils.np_utils import to_categorical

data = pd.read_csv('basketball_shot_log.csv')
predictors = data.drop(['shot_result'], axis=1).as_matrix()
target = to_categorical(data.shot_result)

model = Sequential()
model.add(Dense(100, activation='relu', input_shape = (n_cols,)))
model.add(Dense(100, activation='relu'))
model.add(Dense(100, activation='relu'))
model.add(Dense(2, activation='softmax'))
model.compile(optimizer='adam', loss='categorical_crossentropy',
              metrics=['accuracy'])
model.fit(predictors, target)
```

Classification models

```
Epoch 1/10
128069/128069 [==============================] - 4s - loss: 0.7706 - acc: 0.5759
Epoch 2/10
128069/128069 [==============================] - 5s - loss: 0.6656 - acc: 0.6003
Epoch 3/10
128069/128069 [==============================] - 6s - loss: 0.6611 - acc: 0.6094
Epoch 4/10
128069/128069 [==============================] - 7s - loss: 0.6584 - acc: 0.6106
Epoch 5/10
128069/128069 [==============================] - 7s - loss: 0.6561 - acc: 0.6150
Epoch 6/10
128069/128069 [==============================] - 9s - loss: 0.6553 - acc: 0.6158
Epoch 7/10
128069/128069 [==============================] - 9s - loss: 0.6543 - acc: 0.6162
Epoch 8/10
128069/128069 [==============================] - 9s - loss: 0.6538 - acc: 0.6158
Epoch 9/10
128069/128069 [==============================] - 10s - loss: 0.6535 - acc: 0.6157
Epoch 10/10
128069/128069 [==============================] - 10s - loss: 0.6531 - acc: 0.6166
```

Using models

- Save
- Reload
- Make predictions

Saving, reloading and using your Model

```python
from keras.models import load_model
model.save('model_file.h5')
my_model = load_model('my_model.h5')
predictions = my_model.predict(data_to_predict_with)
probability_true = predictions[:,1]
```

Verifying model structure

```
my_model.summary()
```

```
____________________________________________________________________________________________________
Layer (type)                     Output Shape          Param #     Connected to
====================================================================================================
dense_1 (Dense)                  (None, 100)           1100        dense_input_1[0][0]

____________________________________________________________________________________________________
dense_2 (Dense)                  (None, 100)           10100       dense_1[0][0]

____________________________________________________________________________________________________
dense_3 (Dense)                  (None, 100)           10100       dense_2[0][0]

____________________________________________________________________________________________________
dense_4 (Dense)                  (None, 2)             202         dense_3[0][0]
====================================================================================================
Total params: 21,502
Trainable params: 21,502
Non-trainable params: 0
```

Exercises!

THANK YOU!

habib.gultekin@ou.bau.edu.tr
habib.gultekin@d-teknoloji.com.tr
gultekinhabib@gmail.com
www.habibgultekin.com

Doğuş Technology