

Image Classification with Deep Neural Networks

Victoria Catterson, PhD
Data Scientist, BioSymetrics Inc.

What we will cover

1. Overview of neural networks
2. Introducing the dataset
 - Practical: download and inspect the data
3. How does a neural network learn?
 - Practical: training the network
4. How well has the network learned?
 - Practical: testing the network
5. Improving network performance
 - Practical: trials with different parameters

1. Overview of neural networks

What is a neural network?

- Based on structure of the brain
- Network of simple processing units
 - Performs calculation on input
 - Passes output to next neuron
- Learns by repeated exposure to stimuli
- Multiple names, eg:
 - Artificial neural network (ANN),
neural net, deep belief network



Why are we interested?

Speech
recognition

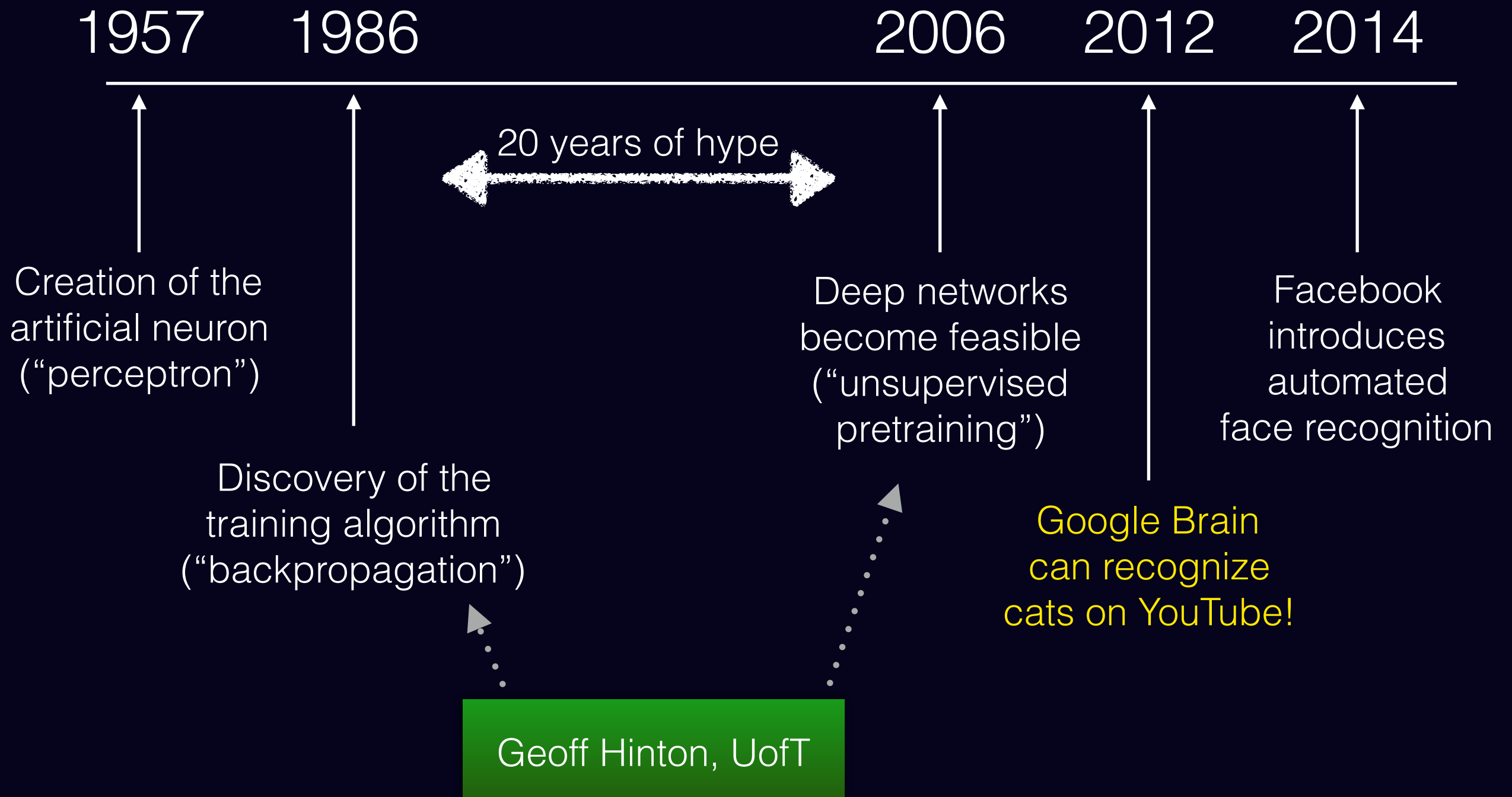
Currently the most
successful machine
learning technique

Recommender
systems

Game playing
e.g. Go

Image
recognition

Timeline



Today

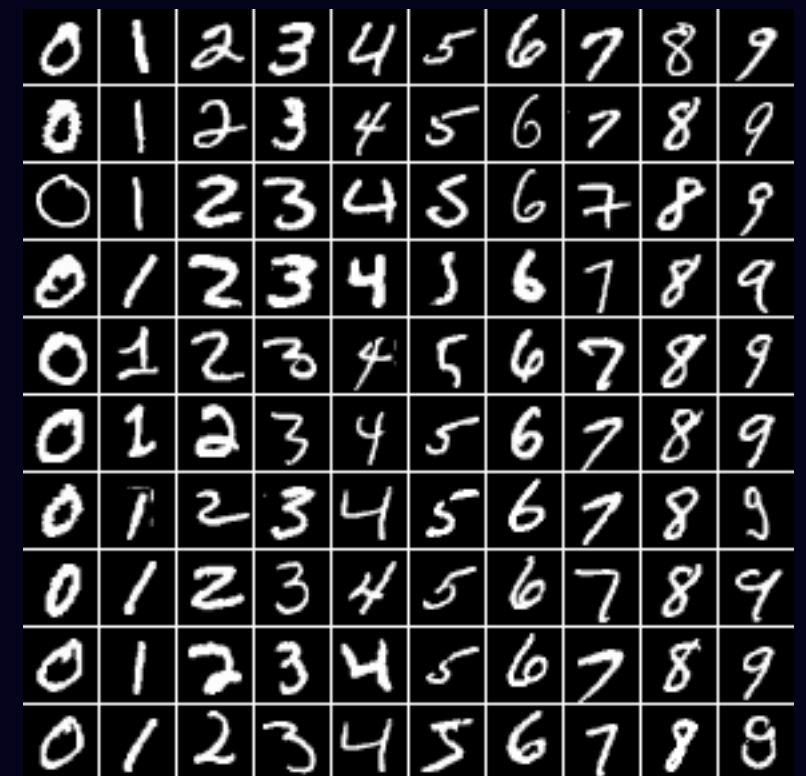
Teach a neural network to
recognize hand-written digits

Same techniques used for
all image recognition,
e.g. cats vs. dogs

2. Introducing the dataset

MNIST database

- 70,000 examples of handwritten digits, 0 – 9
- All images are 28 x 28 pixels and greyscale
- Different styles of writing
- Standard dataset for testing image recognition algorithms
- More details here: <http://yann.lecun.com/exdb/mnist/>



Using MNIST

- Images are split into training and test sets
 - 60,000 for training, 10,000 for testing
- Images are “labeled”
 - We know what digit each image contains
 - Makes it a “supervised learning” task
- Because it’s a standard set, easy to dl and use

Practical: Download and Inspect the Data

Start Python (Anaconda or Terminal + python3)

```
stripe testing $ python3
Python 3.5.1 (default, Dec 27 2015, 18:15:38)
[GCC 4.2.1 Compatible Apple LLVM 7.0.0 (clang-700.1.76)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Start Python (Anaconda or Terminal + python3)

```
stripe testing $ python3
Python 3.5.1 (default, Dec 27 2015, 18:15:38)
[GCC 4.2.1 Compatible Apple LLVM 7.0.0 (clang-700.1.76)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> from keras.datasets import mnist
Using TensorFlow backend.
>>>
```

Start Python (Anaconda or Terminal + python3)

```
stripe testing $ python3
Python 3.5.1 (default, Dec 27 2015, 18:15:38)
[GCC 4.2.1 Compatible Apple LLVM 7.0.0 (clang-700.1.76)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> from keras.datasets import mnist
Using TensorFlow backend.
>>> (x_train, y_train), (x_test, y_test) = mnist.load_data()
>>>
```

Start Python (Anaconda or Terminal + python3)

```
stripe testing $ python3
Python 3.5.1 (default, Dec 27 2015, 18:15:38)
[GCC 4.2.1 Compatible Apple LLVM 7.0.0 (clang-700.1.76)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> from keras.datasets import mnist
Using TensorFlow backend.
>>> (x_train, y_train), (x_test, y_test) = mnist.load_data()
>>> x_train.shape
(60000, 28, 28)
>>>
```

Why are there
60,000 items in
the array?

Why is each item
a 28x28 array?

x_train is the set of 60,000 training
images (28x28 pixels each)

Start Python (Anaconda or Terminal + python3)

```
stripe testing $ python3
Python 3.5.1 (default, Dec 27 2015, 18:15:38)
[GCC 4.2.1 Compatible Apple LLVM 7.0.0 (clang-700.1.76)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> from keras.datasets import mnist
Using TensorFlow backend.
>>> (x_train, y_train), (x_test, y_test) = mnist.load_data()
>>> x_train.shape
(60000, 28, 28)
>>> y_train.shape
(60000,)
>>>
```

A 1 dimensional array,
60,000 items long

What is this
for?

y_train is the list of labels
for the images in x_train

Start Python (Anaconda or Terminal + python3)

```
stripe testing $ python3
Python 3.5.1 (default, Dec 27 2015, 18:15:38)
[GCC 4.2.1 Compatible Apple LLVM 7.0.0 (clang-700.1.76)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> from keras.datasets import mnist
Using TensorFlow backend.
>>> (x_train, y_train), (x_test, y_test) = mnist.load_data()
>>> x_train.shape
(60000, 28, 28)
>>> y_train.shape
(60000,)
>>> x_train[0]
>>>
```

What does
this give?

How about
y_train[0]?

Start Python (Anaconda or Terminal + python3)

```
stripe testing $ python3
Python 3.5.1 (default, Dec 27 2015, 18:15:38)
[GCC 4.2.1 Compatible Apple LLVM 7.0.0 (clang-700.1.76)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> from keras.datasets import mnist
Using TensorFlow backend.
>>> (x_train, y_train), (x_test, y_test) = mnist.load_data()
>>> x_train.shape
(60000, 28, 28)
>>> y_train.shape
(60000,)
>>> x_train[0]
>>>
```

- Explore further:
 - x_train[1]
 - x_test.shape
 - x_test[0] and y_test[0]

Questions?

Format of the images

- As humans, we look at images in 2D
 - Relative spatial information is important
- But the NN doesn't care about the pixel order
 - It learns the importance of relative locations
- It's easier to work with a 1D array of pixels
 - $28 \times 28 = 784$ pixels
- We will reshape the dataset to $60,000 \times 784$

Colour of the images

- The images start as greyscale
 - Pixel values between 0 and 255
- ANNs work best with numbers near zero
 - We'll see why later
- We will rescale the data to values 0–1

- Use an editor to open `mnist_prog1.py`

- Use an editor to open mnist_prog1.py
- Quit the Python interpreter if it's still running

```
>>> quit()  
stripe testing $
```

- Use an editor to open mnist_prog1.py
- Quit the Python interpreter if it's still running

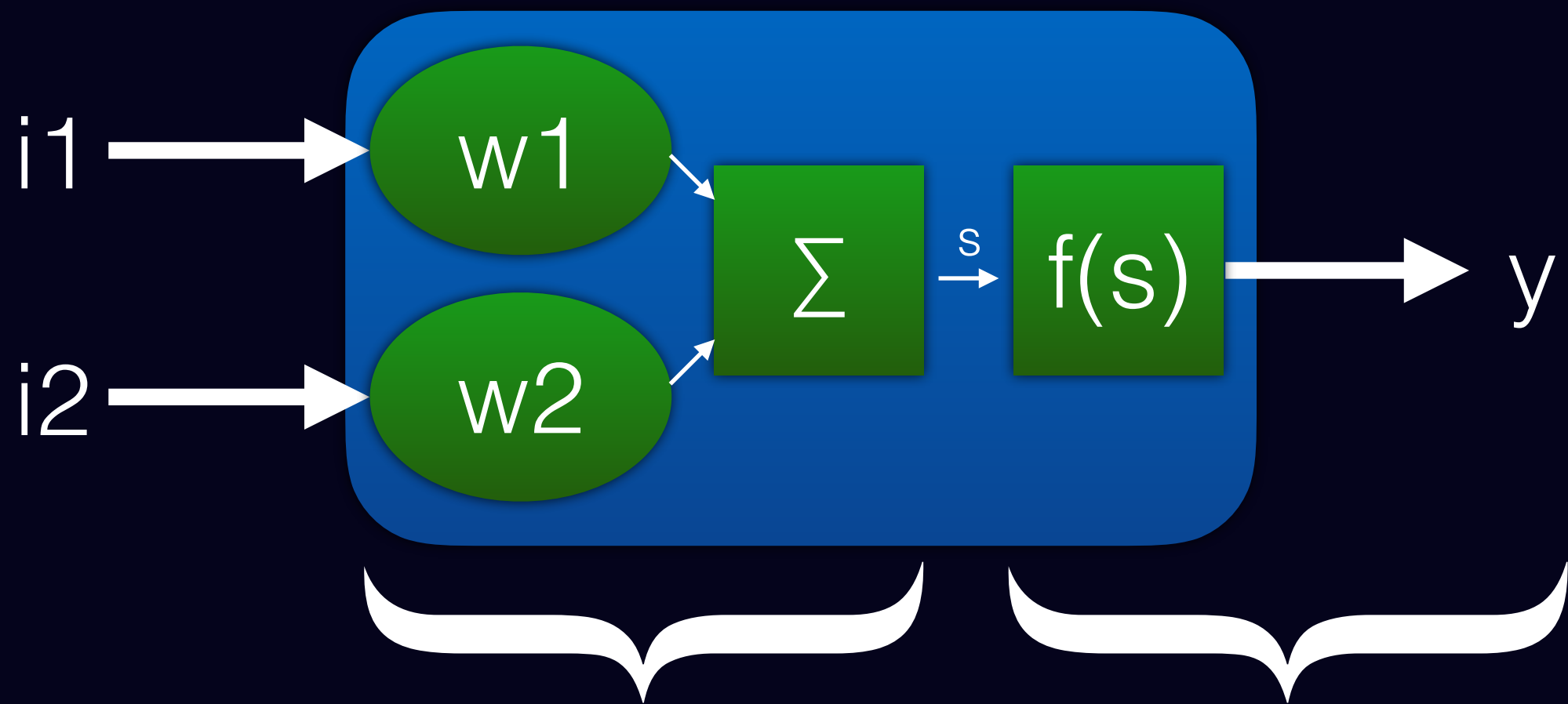
```
>>> quit()  
stripe testing $
```

- Now run the program!

```
stripe testing $ python3 mnist_prog1.py  
Using TensorFlow backend.  
The data starts with shape (60000, 28, 28) and (60000,)  
The data becomes shaped as (60000, 784) and (60000,)  
The max value in the training set is 255  
After scaling, the max value in the training set is 1.0  
stripe testing $
```

3. How does a neural network learn?

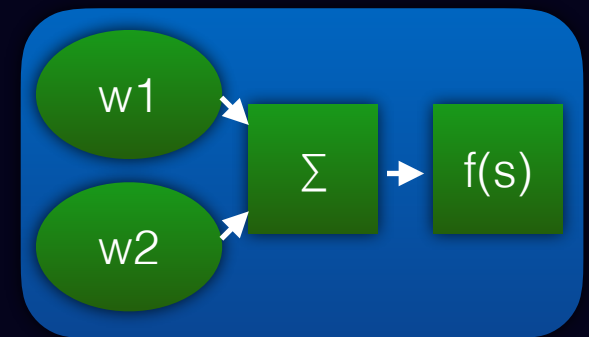
A single neuron



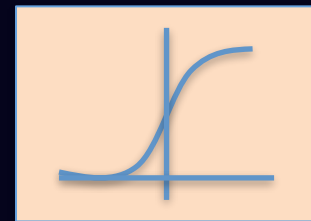
weighted sum
 $s = i_1 * w_1 + i_2 * w_2$

activation function
 $y = f(i_1 * w_1 + i_2 * w_2)$

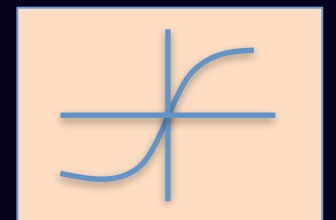
Typical activation functions



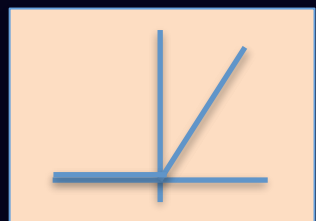
- Sigmoid: $f(s) = 1/(1 + e^s)$



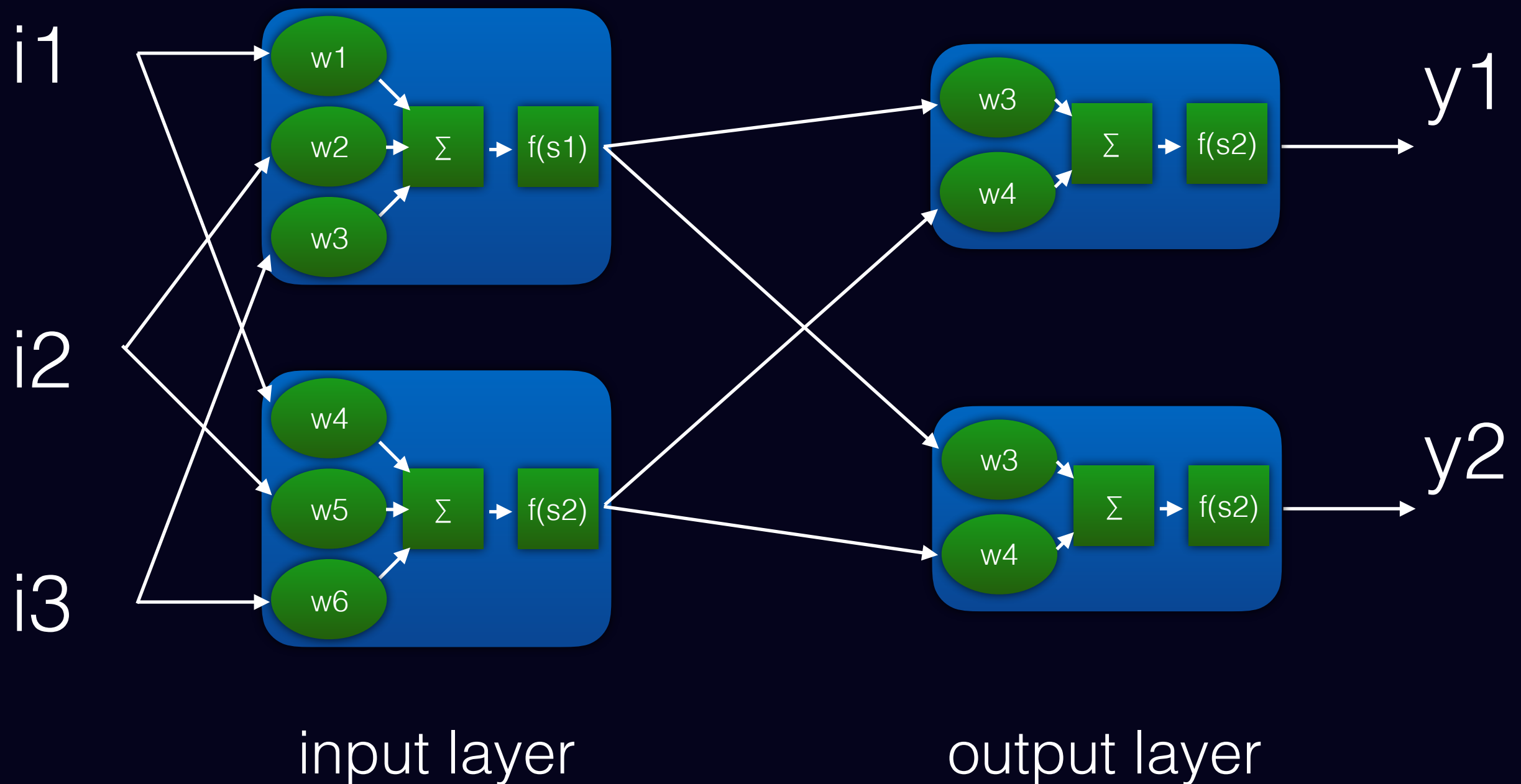
- Hyperbolic tangent (tanh): $f(s) = \tanh(s)$



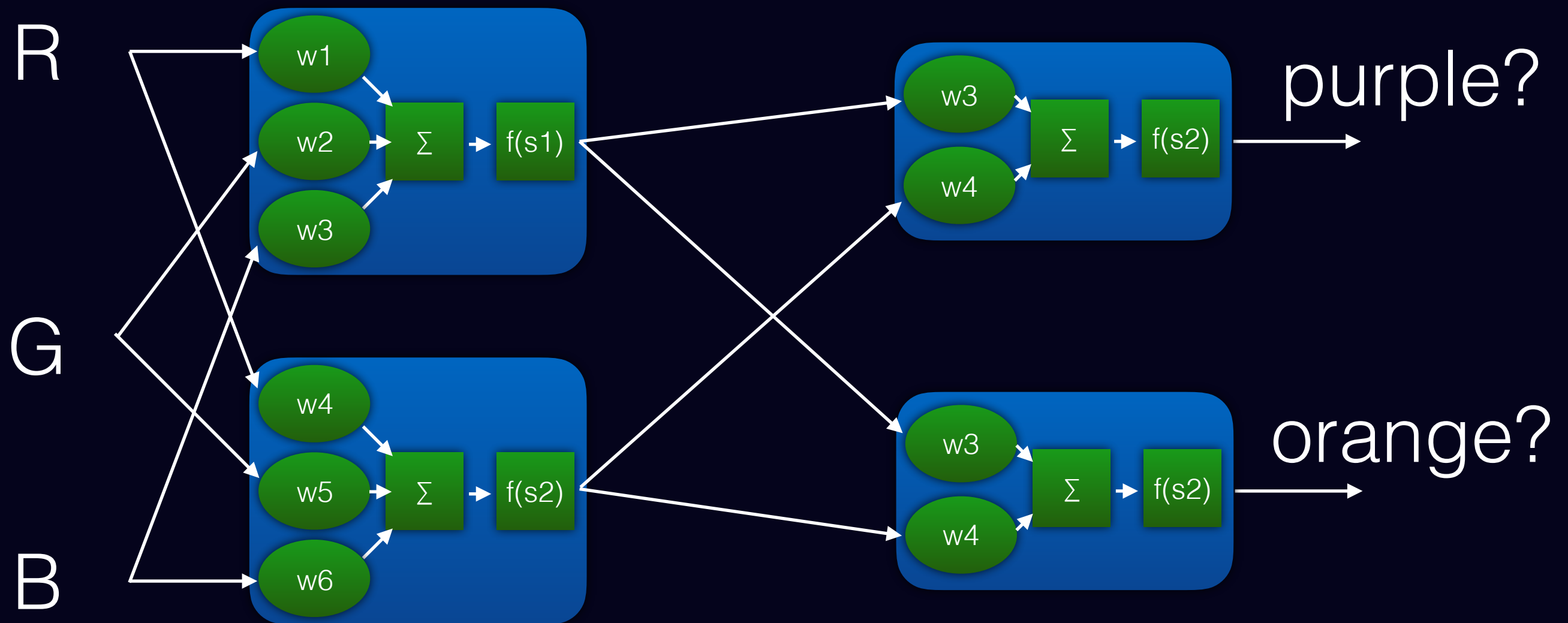
- Rectified Linear Unit (ReLU): $f(s) = \max(0, s)$



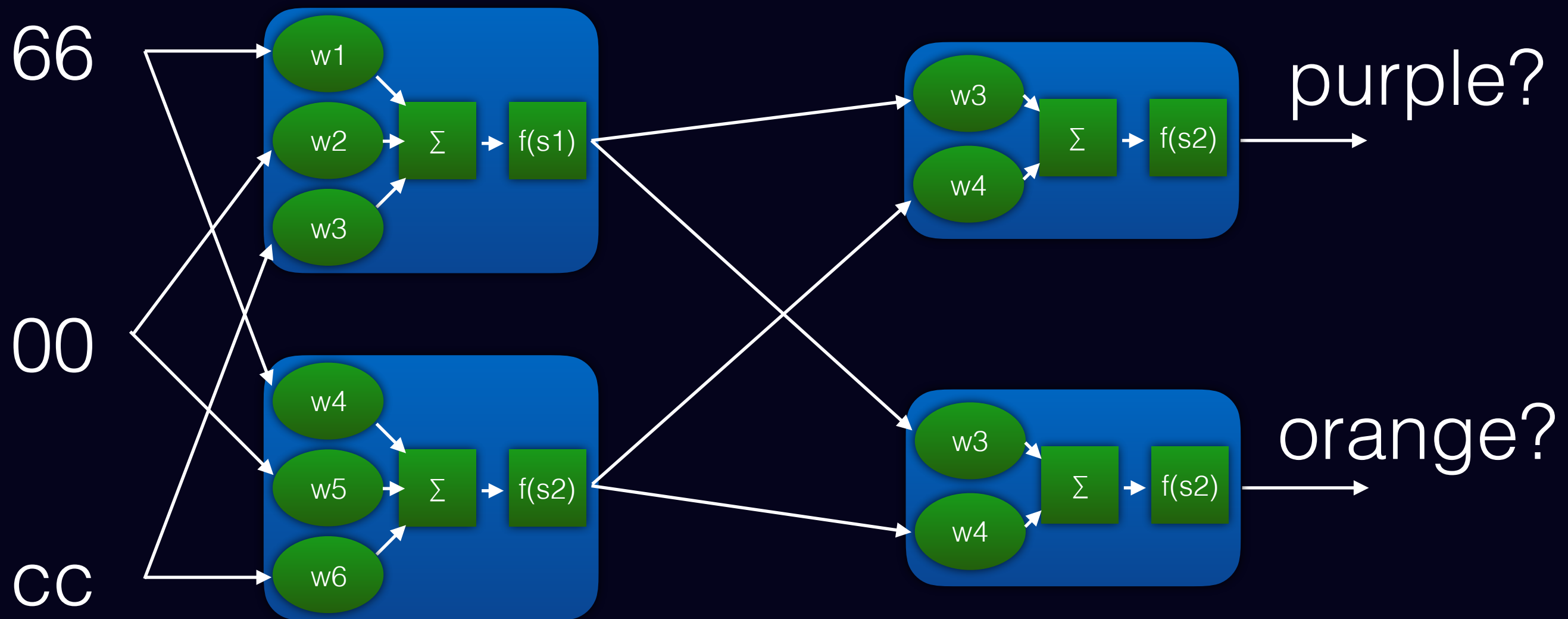
A network of neurons



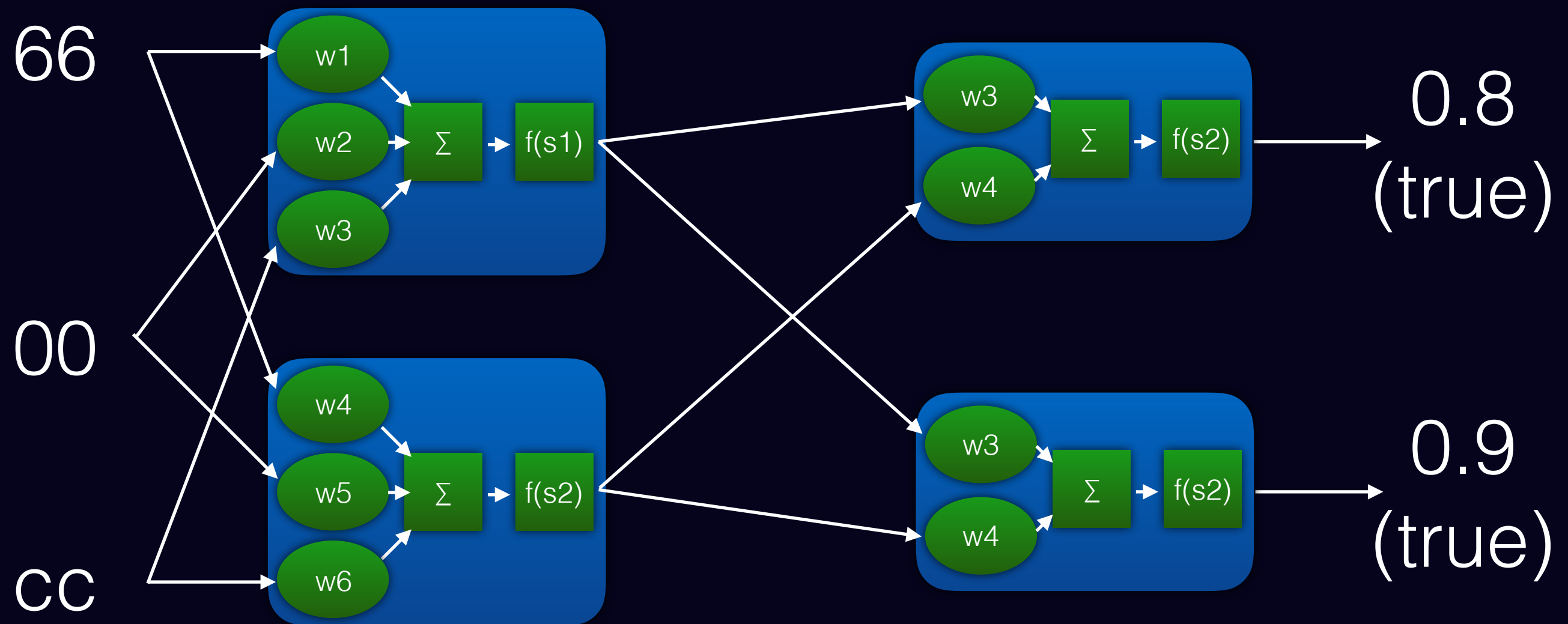
What do we call a colour?



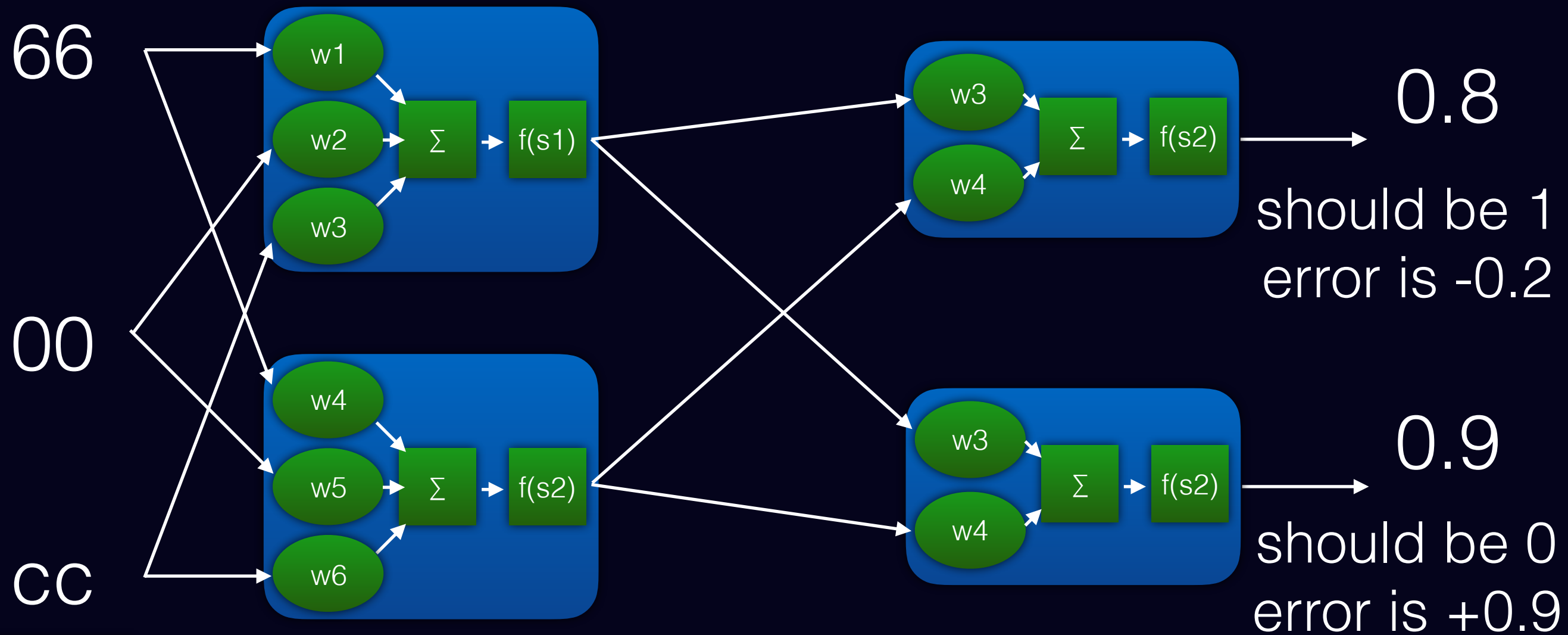
Show an example...



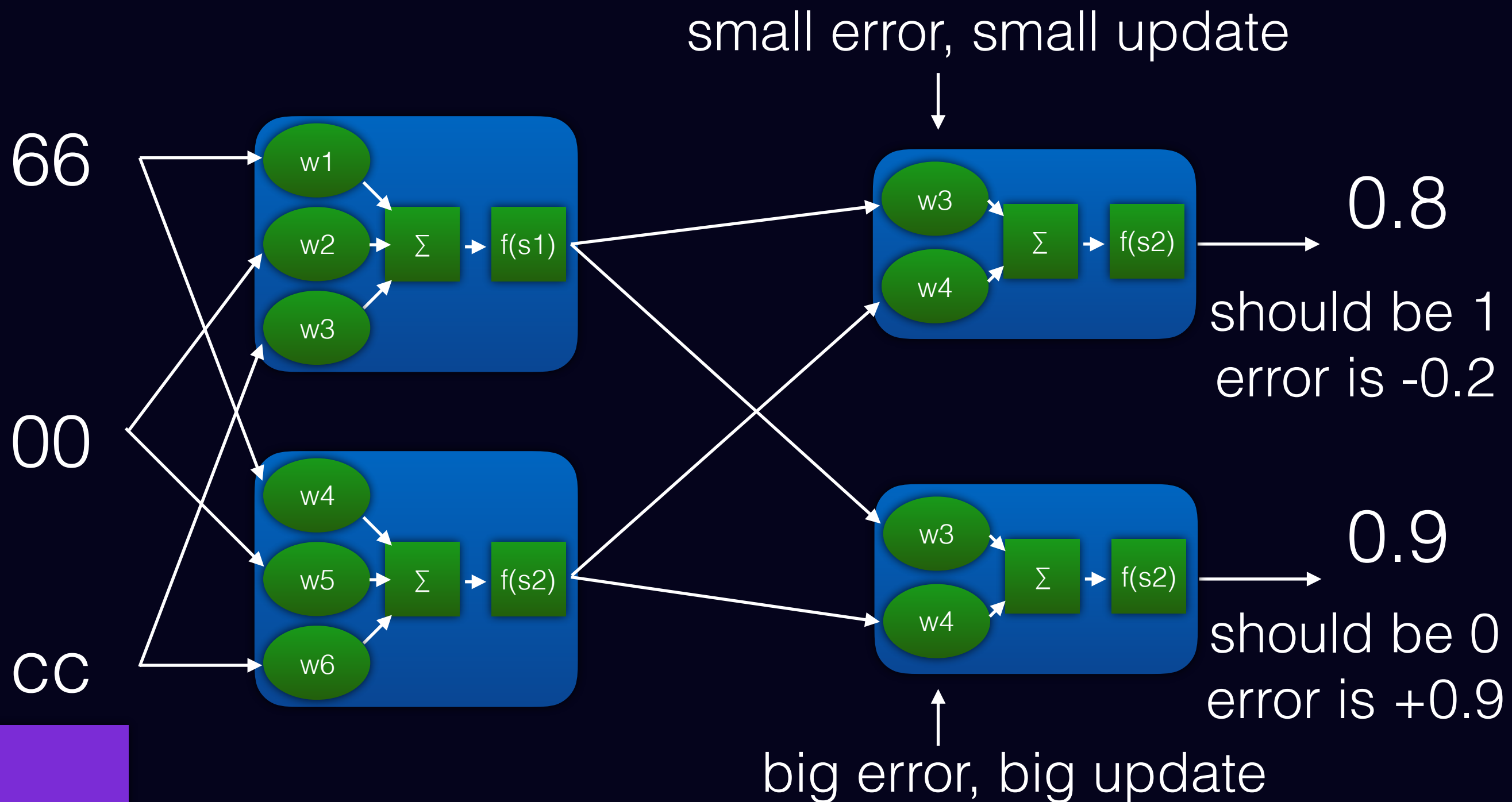
First guess is random...



Errors are calculated...

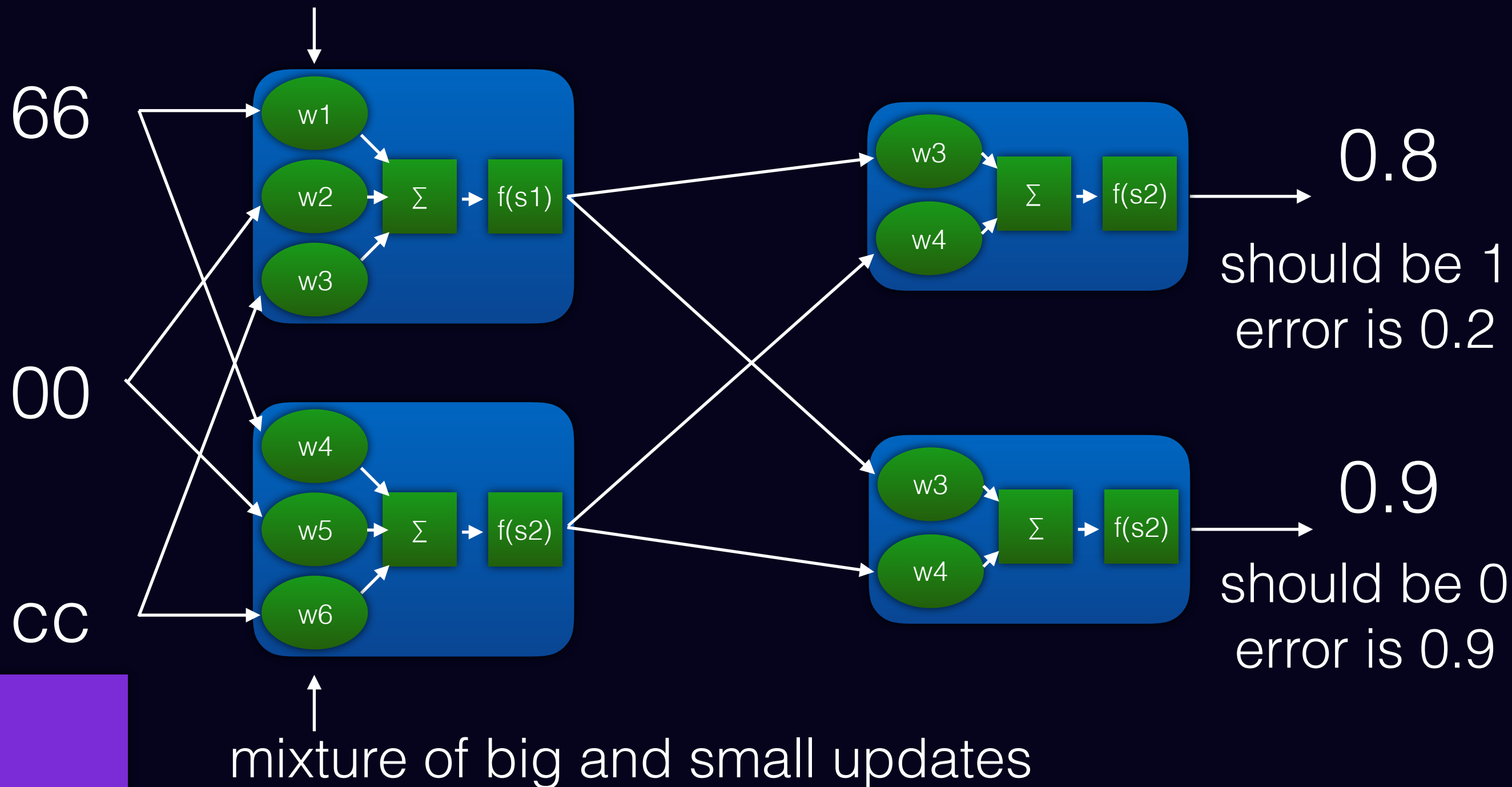


Feedback updates weights

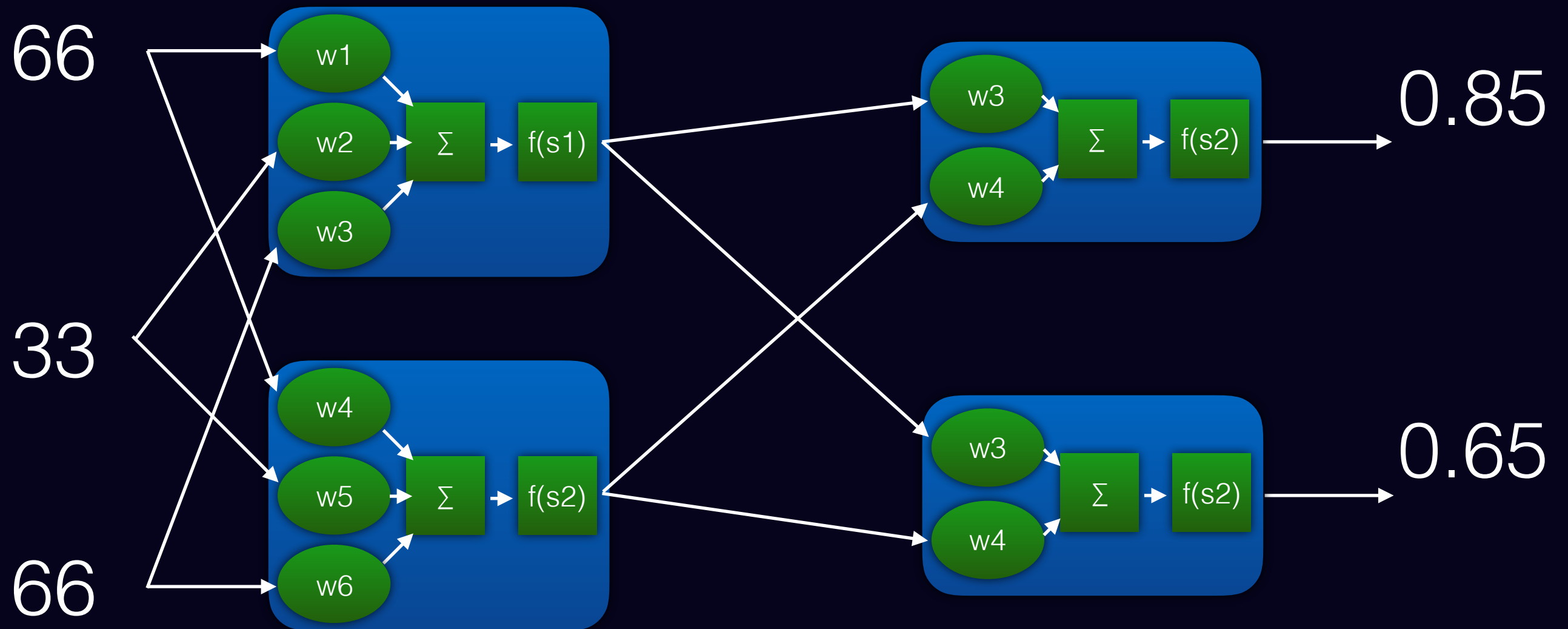


Feedback updates weights

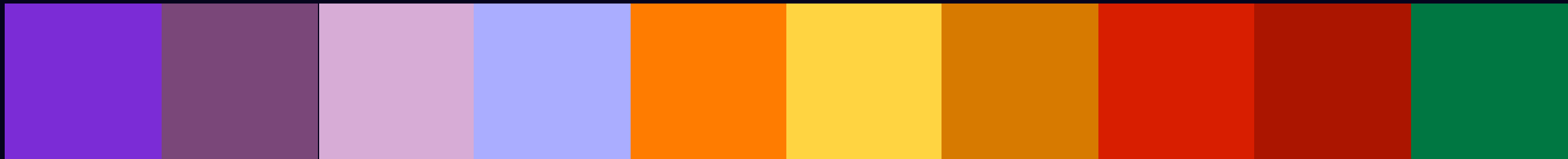
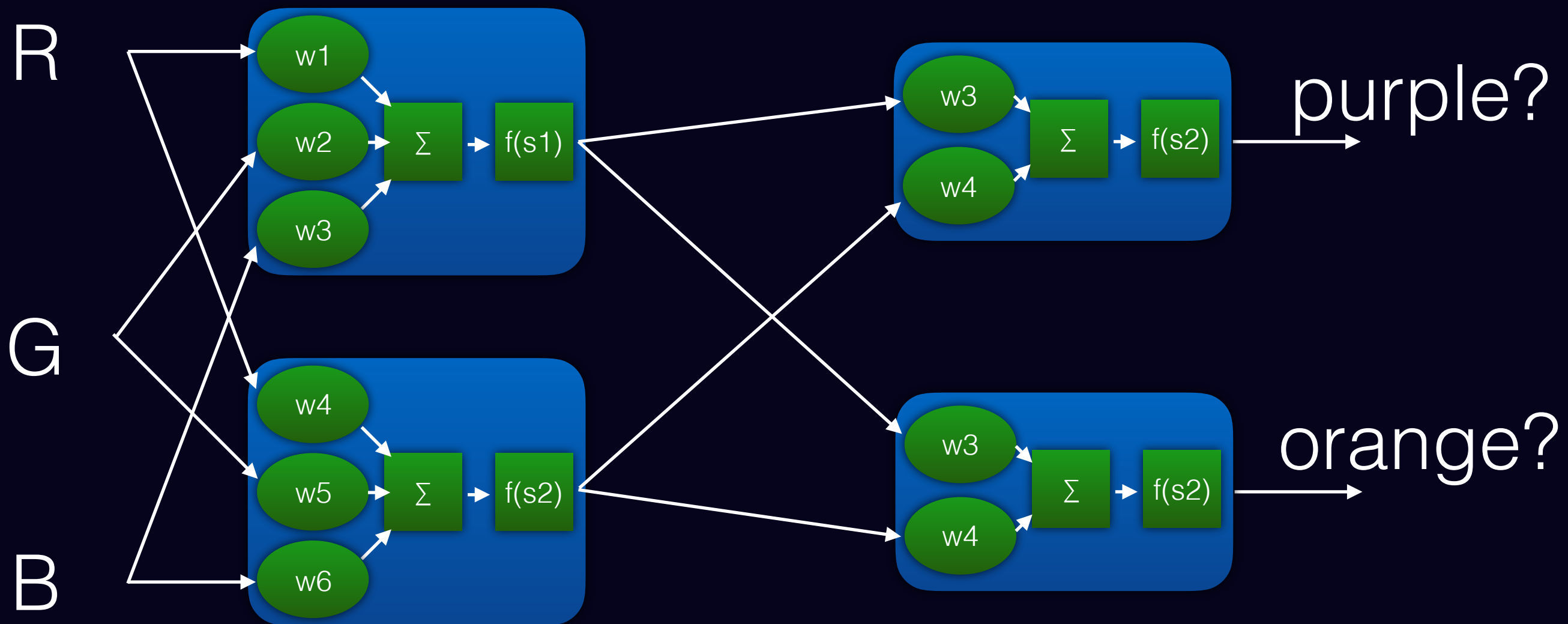
mixture of big and small updates



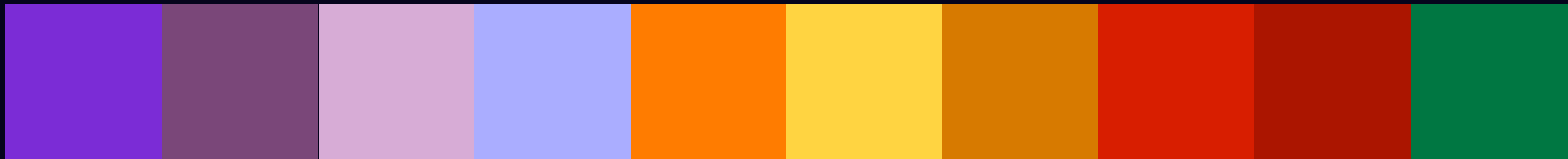
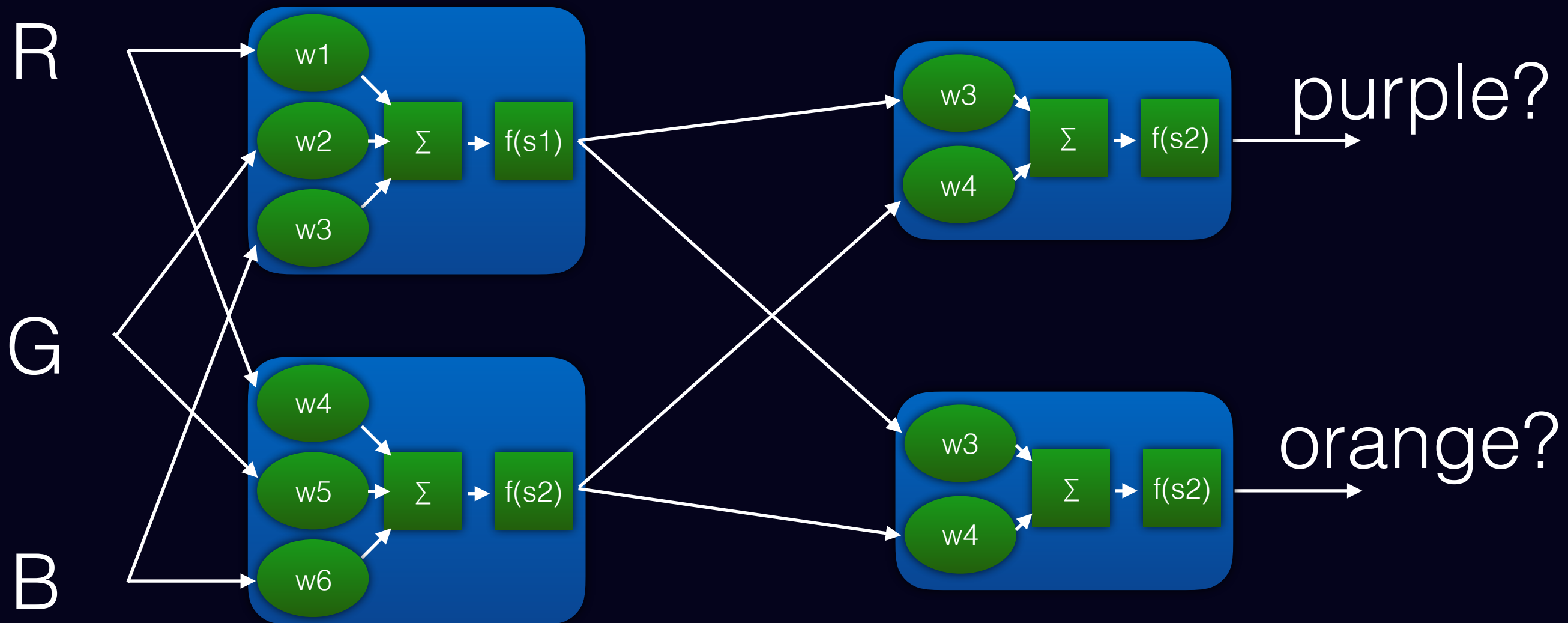
More likely to get it right next time



Show every example in the training set (one epoch)



Repeat training until weights stop changing



Digit recognition

- One training epoch: all 60,000 images
- Learning: weights on every neuron updated
- Number of inputs? 784 (one per pixel)
- Number of outputs? 10 (one per digit: 0–9)

Practical: Training the Network

- Use an editor to open mnist_prog2.py

```
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense
from keras.optimizers import RMSprop
```

- First thing to notice: more imports than last time
 - These are used for configuring the network
- Lines 7 to 22 are the same as mnist_prog1.py...

- Lines 25&26: one final step of data preparation!

```
# convert class vectors to binary class matrices  
y_train = keras.utils.to_categorical(y_train, 10)  
y_test = keras.utils.to_categorical(y_test, 10)
```

What did
y_train look
like before?

We want 10
outputs: one for
each digit

to_categorical converts a number
into an array of binary values

- Lines 29 to 32 set up the network architecture

```
model = Sequential()  
model.add(Dense(16, activation='relu', input_shape=(784,)))  
model.add(Dense(16, activation='relu'))  
model.add(Dense(10, activation='softmax'))
```

Why is the
input_shape
784?

What does
'relu' mean?

'softmax' is used when there is a
probability distribution over a set of
values

In this case, the probability of the
image being one of the 10 digits
sums to 100%

How many
neurons are in
the first layer?

How many layers
are there?

Why does the
last layer have
10 neurons?

One for each of the digits

Training the network

- The final lines:
 - Compile the network
 - Fit the network on the training data
 - Evaluate its accuracy
- Now let's run it!

```
stripe testing $ python3 mnist_prog2.py
```

4. How well has the network learned?

Network Testing

- Let's look again at line 48:

```
score = model.evaluate(x_test, y_test, verbose=0)  
print('Test accuracy:', score[1])
```

- We “fit” (train) the network using the training set
- But evaluate its performance using the test set
 - Accuracy is the percentage of right answers
- The network never sees the test set during training
 - Makes it a fair test & prevents “overfitting”

Practical: Testing the Network

- Use an editor to open mnist_prog3.py
- (It's the same except at the end)

```
y_pred = model.predict(x_test)
print(y_test[0])
print(y_pred[0])
print("Actual value:", y_test[0].argmax())
print("Predicted:", y_pred[0].argmax())
```

What do you think
model.predict(x_test)
will do?

What is y_test[0]?
(What was it before?)

We've converted the digit label into
a set of 10 binary values

argmax() finds the index of the
highest value in an array

- Now let's run it!

```
stripe testing $ python3 mnist_prog3.py
```

5. Improving performance

Accuracy is pretty good!

- But can we improve it?
- Play around with different network parameters:
 - Number of neurons in the layers?
 - Number of layers?
 - Activation function?
 - Number of epochs?
 - Others?
- Keras documentation: <https://keras.io/activations/>

Networks with more than 3 layers
are called Deep Neural Networks

Does a Deep Neural Network
perform better?

Network architecture	Activation functions	Epochs	Other comments	Accuracy
(16, 16, 10)	ReLU, ReLU, softmax	4	Scaled inputs	93%

Conclusions

Today we covered

- What is a neural network?
 - And why are people interested right now?
- How to build, train, and test a neural network
 - And compare performance of different nets
- How does the network actually learn?
- What are some of the network parameters?
 - And how important are layers/neurons/AFs?

Next steps

- MNIST is just one dataset
 - Although very foundational
 - See Keras docs for more complex datasets
- Today we used a Multi-Layer Perceptron (MLP)
 - Many other types such as recurrent NNs, convolutional NNs
 - See Keras docs for different layer types

Thank you!