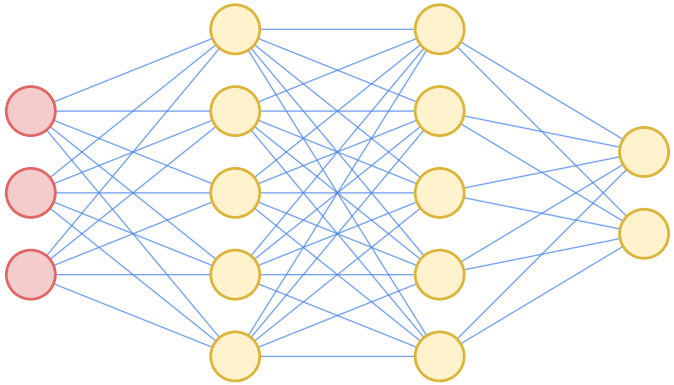
UNIT 02 · CHAPTER 1

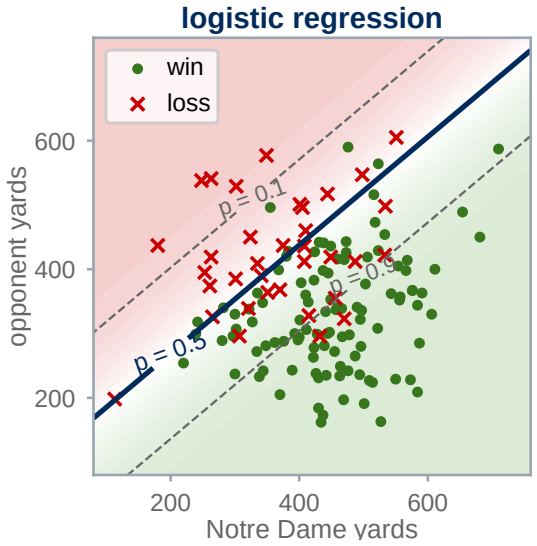
Feed-Forward Networks



CSE 30124 · Introduction to Artificial Intelligence
William Theisen · University of Notre Dame

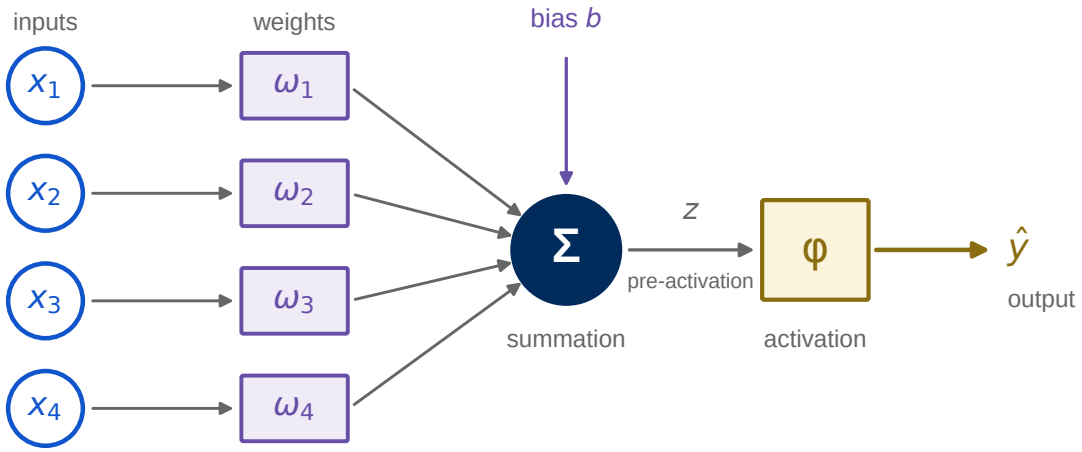
Predicting Game results

In **Model Evaluation** we asked "Did we win?" and a **logistic regression** answered with a straight line through the two teams' yards, getting **83.7%** of unseen games right.



Review: Perceptrons

Really, a **logistic regression** is the same thing as a **perceptron**, just with a _____ **activation function**. Generally, we can call this **`unit`** an _____



Deep learning is all about how we can use _____

Artificial Neurons

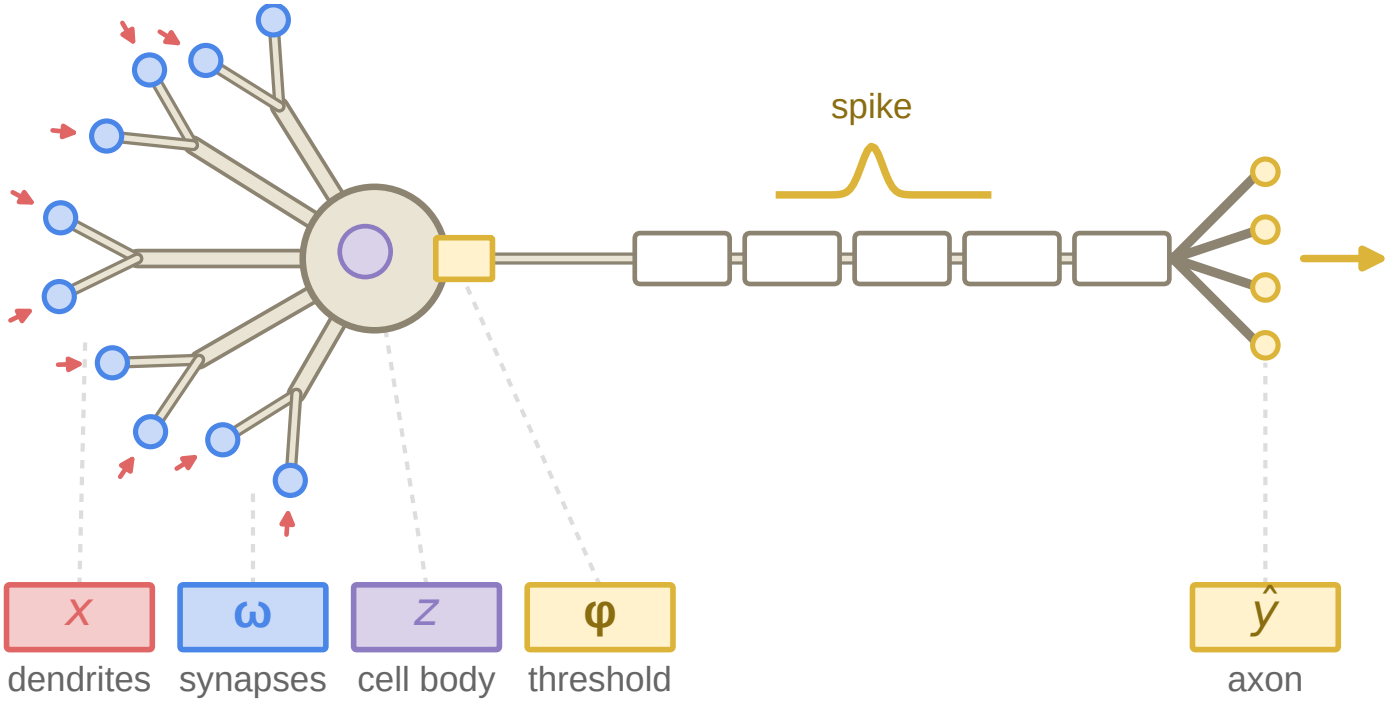
- _____ **(X):** The information the neuron is receiving
- _____ **(W):** defines how important a given input is for the neuron
- _____ **(b or ω_0):** A small adjustment to the weighted sum of the inputs
  - Think of this as shifting the neurons threshold to make it harder or easier to activate

- **Summation (Σ):** The weighted sum of the inputs and bias
  - represents the neurons "total input", also known as the **pre-activation**

The summation **transforms multiple inputs into one number** that summarizes the information that the neuron cares about.

- _____ **(ϕ):** A function applied to the pre-activation of the neuron
- _____ **($\hat{y}$):** The final resulting value from the activation function

Real Neurons

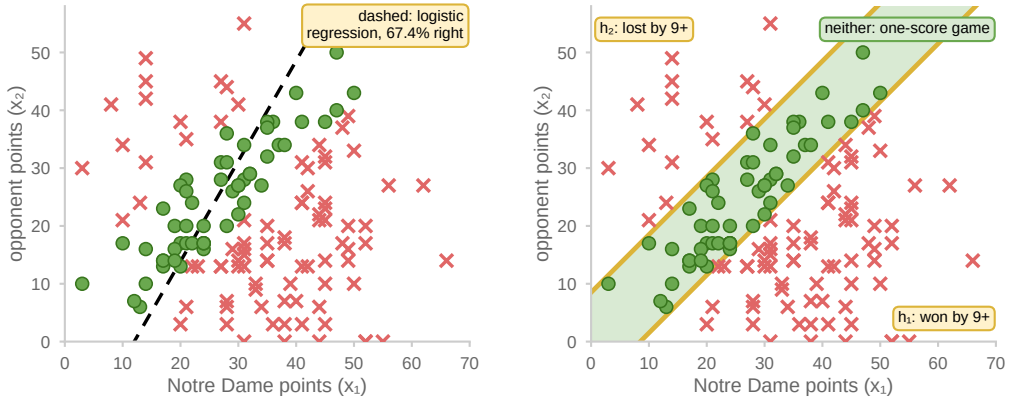


The **artificial neuron** is a (very) loose copy of a real one. A real neuron collects signals from other neurons on its **dendrites**, each synapse makes a signal count for more or less, and the neuron **fires a spike down its ____ only when the total crosses a threshold**.

Our brains have about **86** billion neurons and something like **100** trillion synapses. Our football network has **3** neurons.

The XOR Problem

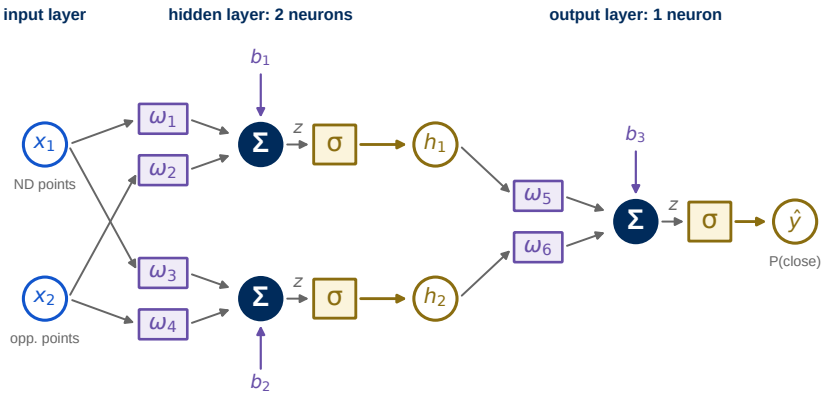
What if instead of asking **win** or **loss**, we wanted to **decide whether a game was close or not (within 8 points)**



Asking if a game was close or not results in data that is very similar to the **XOR problem** we saw with **perceptrons**, where our **data is not** _____

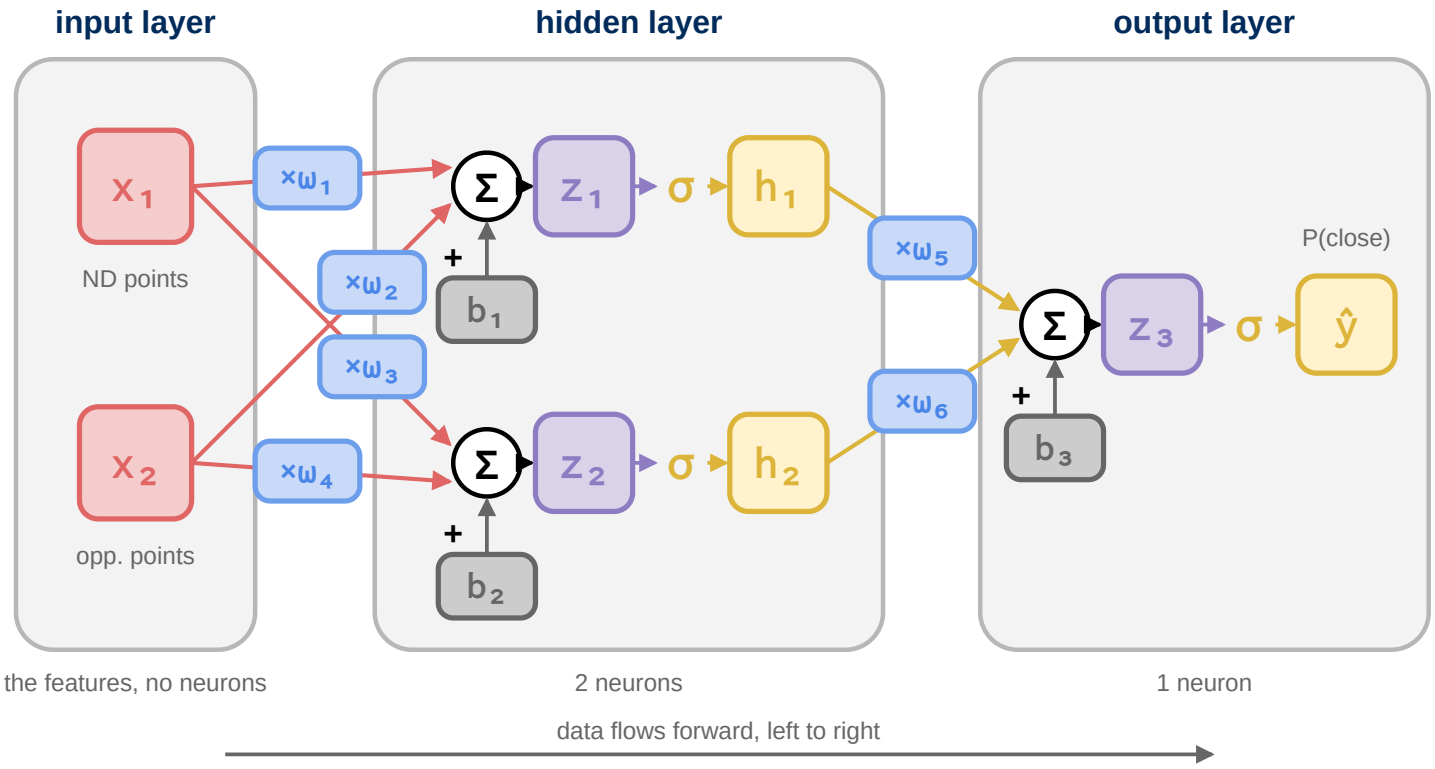
The XOR Problem

A **neuron** is capable of drawing a single line, but our data requires two lines to separate. We can have two **neurons** each create one of the two lines and use a _____ to **combine their output!**



Stacking **neurons** like this, so some take the outputs of others as their inputs, gives a **multi-layer perceptron (MLP)**, or a **feed-forward neural network (FFN)**

Neural Networks: Diagram

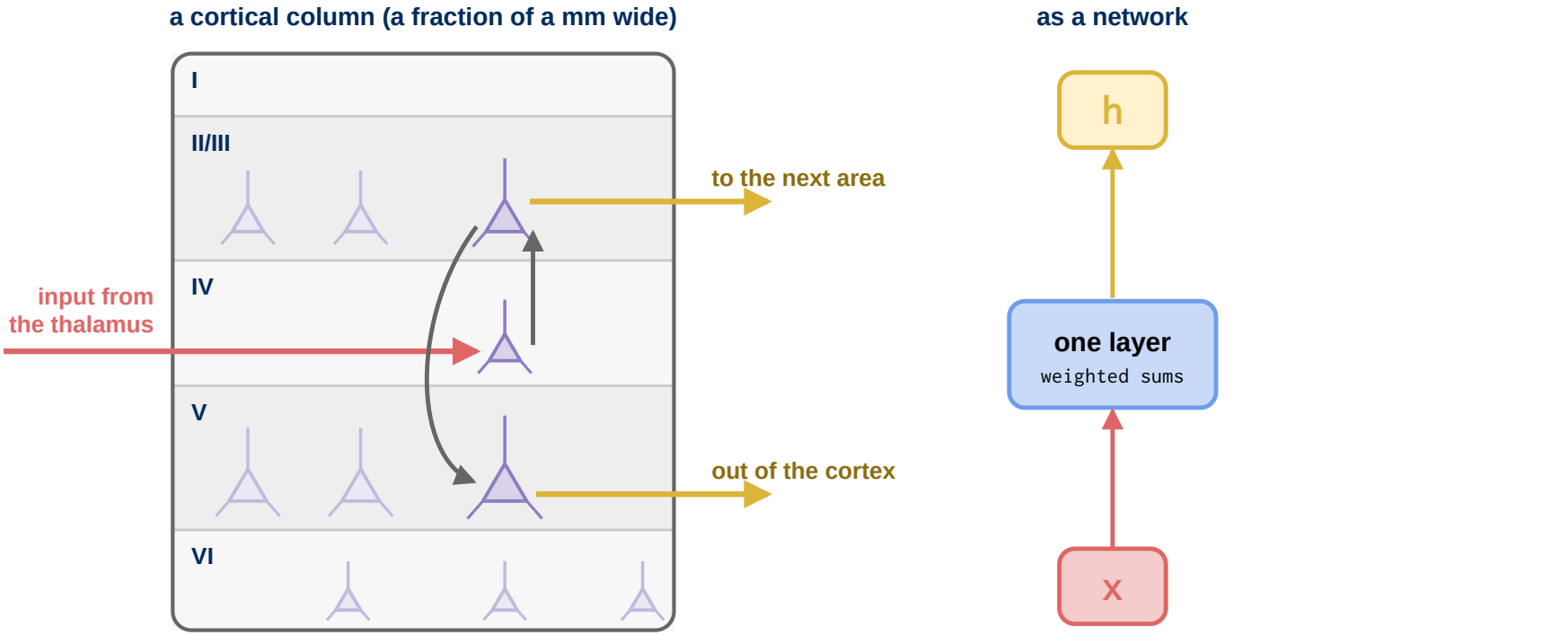


Combining our three **artificial neurons** gives us a simple **FFN**

The organization of neurons in a neural network is called its _____.

The Cortical Column

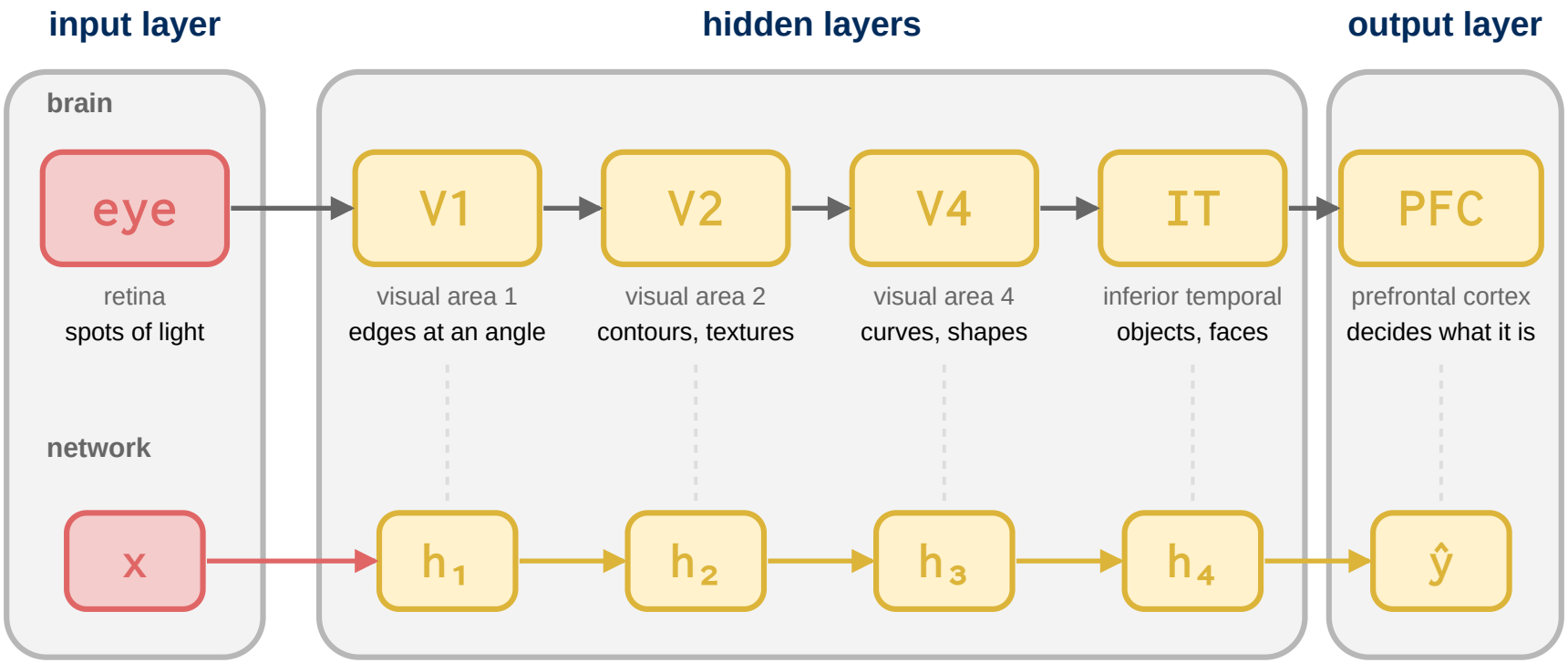
In **1957** Vernon Mountcastle found that neurons stacked straight down through a cat's cortex all responded to **the same patch of skin**. The cortex is built out of these _____, each a fraction of a millimeter wide.



Douglas and Martin (**1989**) proposed that columns all over the cortex run **the** _____ _____, a bit like a network built by stacking the same kind of layer.

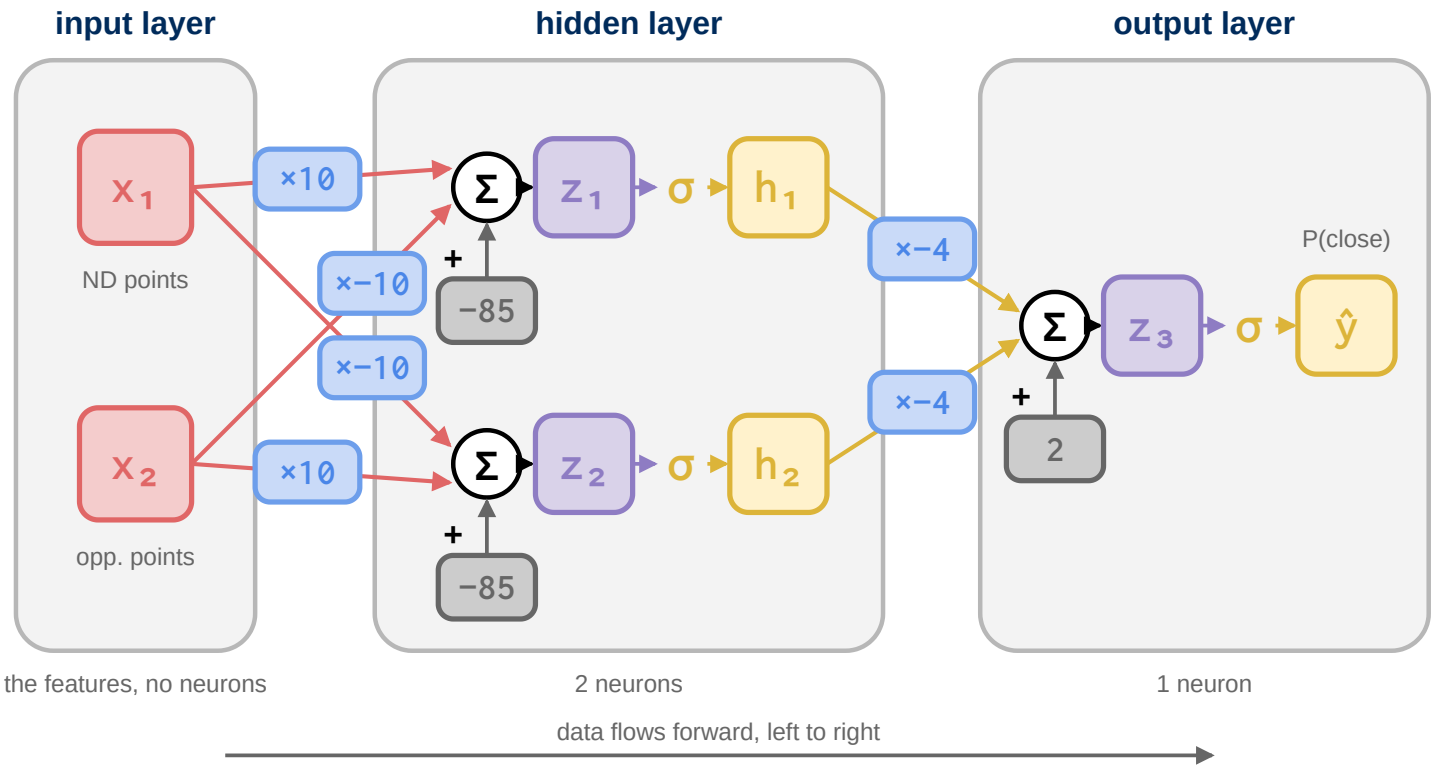
Depth in the Brain

Our network has an input layer, an output layer and **one layer** of neurons in between. Vision in the brain has the same shape, only _____: a chain of areas, each finding **more complex patterns** than the one before.



Felleman and Van Essen (**1991**) counted **32** visual areas in the monkey brain, arranged in about **10** levels.

Network Diagram with Weights



[0.1] two neurons look at the score

$$h_1 = \sigma(10x_1 - 10x_2 - 85) \qquad h_2 = \sigma(10x_2 - 10x_1 - 85)$$

[0.2] a third combines them

$$\hat{y} = \sigma(2 - 4h_1 - 4h_2)$$

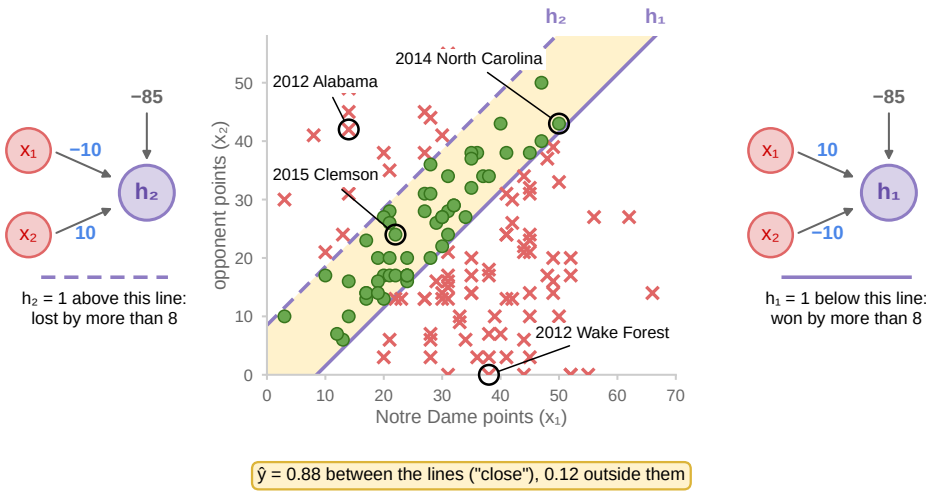
Producing Output from a Neural Network

Producing output with a **neural network** is called "performing the _____"

game	score	h_1	h_2	$\hat{y}$	close?
2014 North Carolina	50–43	0	0	0.88	yes
2015 Clemson	22–24	0	0	0.88	yes
2012 Wake Forest	38–0	1	0	0.12	no
2012 Alabama (title game)	14–42	0	1	0.12	no

Decision Regions

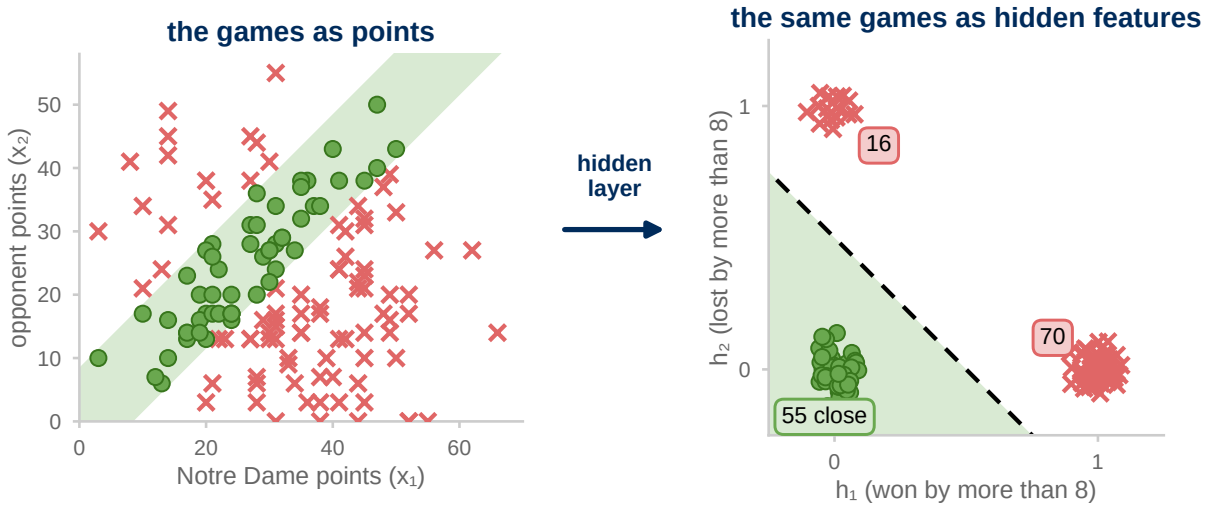
Our first two **neurons** each draw a _____, our third **neuron** combines the two boundaries to create a **decision region**



On all **141** games it gets every one right (**100%**), while no straight line could do better than **88.7%**.

Hidden Layer Features

Layers that don't produce the final output are called _____.



These are what transform our data into _____ points! They create new features for our data that our third **neuron** can separate.

Hidden Features in the Brain

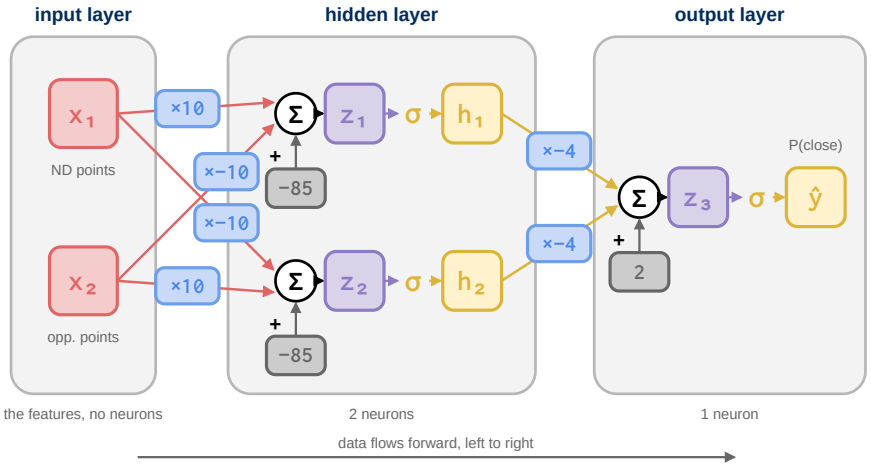
Our hidden layer turned each game into new features, **won by more than 8** and **lost by more than 8**. Do the hidden layers of a bigger network find features like the brain's? Yamins et al. (**2014**) trained a network to recognize objects in photos, then compared its hidden layers with neurons recorded in monkeys looking at the same photos.

- Its **middle hidden layers** matched ___, the area that responds to curves and shapes
- Its **last hidden layer** matched _, the area that responds to whole objects and faces
- Networks that were **better at recognizing objects** matched the brain better

The networks were never shown any brain data. Learning to do the same job, they ended up with **hidden features** a lot like the brain's.

Condensing the Hidden Layer

We can compactly represent the operations of the hidden layer as a single matrix multiplication.



X inputs, one row per game
<4, 2>

	x_1	x_2
2014 North Carolina	50	43
2015 Clemson	22	24
2012 Wake Forest	38	0
2012 Alabama	14	42

W_1 weights, one row per neuron
<2, 2>

	x_1	x_2
neuron 1	10	−10
neuron 2	−10	10

b_1 biases
<2>

−85
−85

Weights as Matrices

four games times both hidden neurons' weights, plus the bias (added to every row)

$$Z_1 = XW_1^\top + b_1 = \begin{bmatrix} 50 & 43 \\ 22 & 24 \\ 38 & 0 \\ 14 & 42 \end{bmatrix} \begin{bmatrix} 10 & -10 \\ -10 & 10 \end{bmatrix} + \begin{bmatrix} -85 & -85 \end{bmatrix}$$

each row is one game, each column one hidden neuron

$$= \begin{bmatrix} 70 & -70 \\ -20 & 20 \\ 380 & -380 \\ -280 & 280 \end{bmatrix} + \begin{bmatrix} -85 & -85 \end{bmatrix} = \begin{bmatrix} -15 & -155 \\ -105 & -65 \\ 295 & -465 \\ -365 & 195 \end{bmatrix}$$

the sigmoid of every cell, the A1 the code prints

$$A_1 = \sigma(Z_1) = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$$

NOTE: This example is a little goofy because the result of the transpose is the same matrix

Forward Pass of a Neural Network

```python
import torch

X = torch.tensor([[50., 43.], [22., 24.], [38., 0.], [14., 42.]])

W1 = torch.tensor([[10., -10.], [-10., 10.]])
b1 = torch.tensor([-85., -85.])

A1 = torch.sigmoid(X @ W1.T + b1)  # b1 is added to every row

print(X.shape, W1.shape, b1.shape)
print(A1.round(decimals=3))
```

OUTPUT

```
torch.Size([4, 2]) t
tensor([[0., 0.],
        [0., 0.],
        [1., 0.],
        [0., 1.]])
```

similar to `sklearn` for machine learning, `PyTorch` is the library we will use to implement all of our **neural networks**.

The `@` symbol is how we multiply matrices and `.T` gives us the transpose of a matrix

A Forward Pass in the Brain

Thorpe, Fize and Marlot (**1996**) flashed photos for just **20** ms and asked people **"is there an animal in it?"**

Their brain signals for "animal" and "no animal" had already split apart about **150** ms after the photo appeared.

Neurons are slow (milliseconds per step), so **150** ms is only enough time for the signal to go up the visual hierarchy about _____. A first look at a scene seems to work **much like a** _____: no loops, no second guesses.

Activation Functions

Critically, **activation functions** introduce _____ into the model. This is what allows **neural networks** to learn complex patterns (beyond just linear relationships).

two layers with no activation function

$$Z_2 = (XW_1^T + b_1)W_2^T + b_2$$

multiply out

$$= X(W_2W_1)^T + (b_1W_2^T + b_2)$$

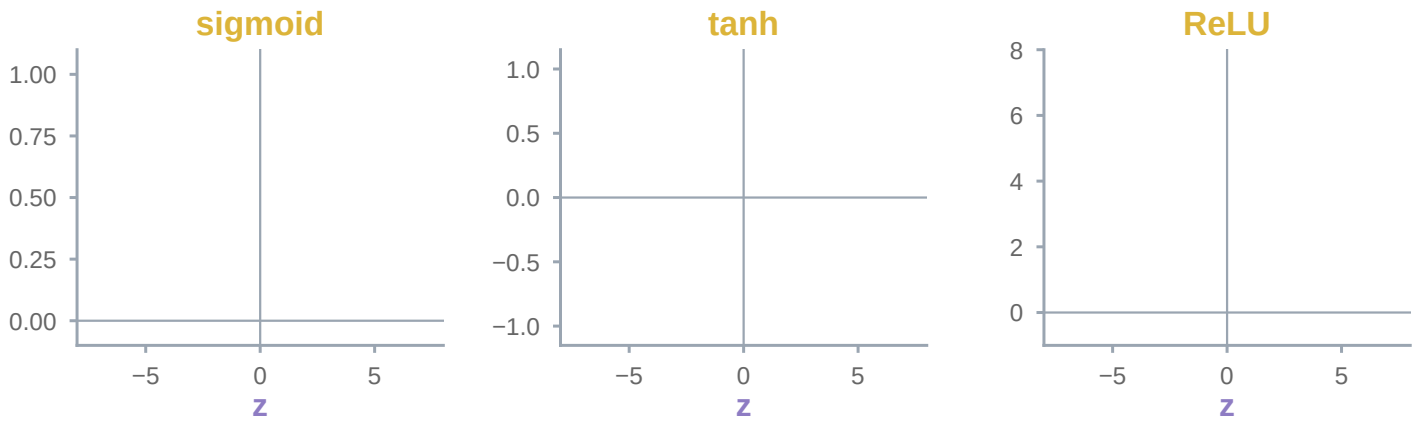
a single layer: its weights are the product of the two, its bias the bracket

$$= XW^T + b$$

If we didn't have **activation functions**, our **neural network** would be no more powerful than just a single **neuron**. However, this power comes at the cost of _____.

Activation Functions

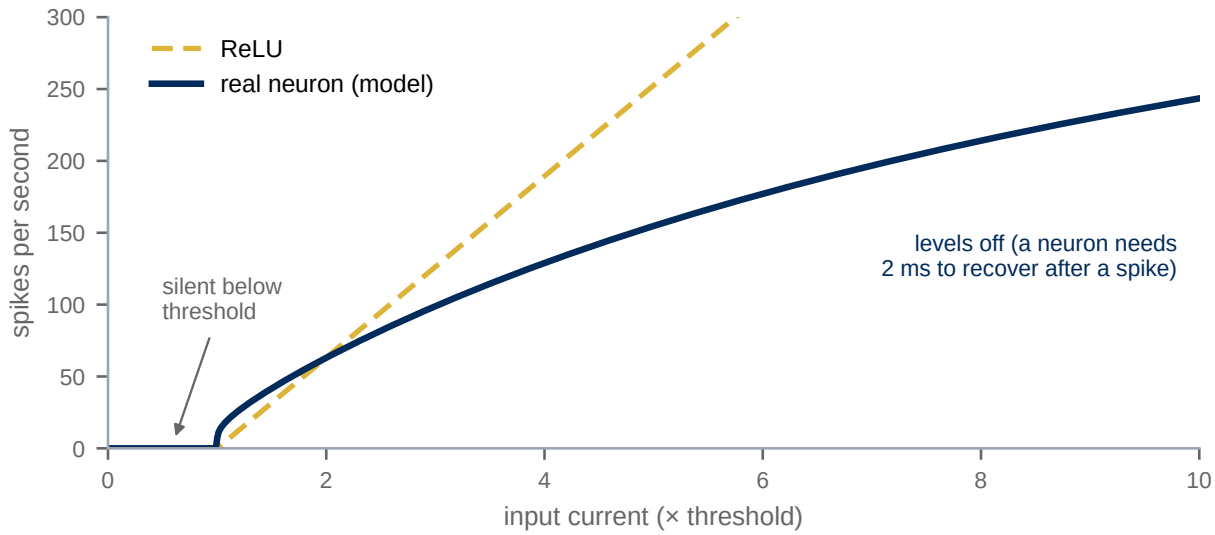
For our simple **neural network**, we are using the **sigmoid function** as the **activation function**, but it's not the only choice.



Other common options are the **hyperbolic tangent** and the **ReLU** function

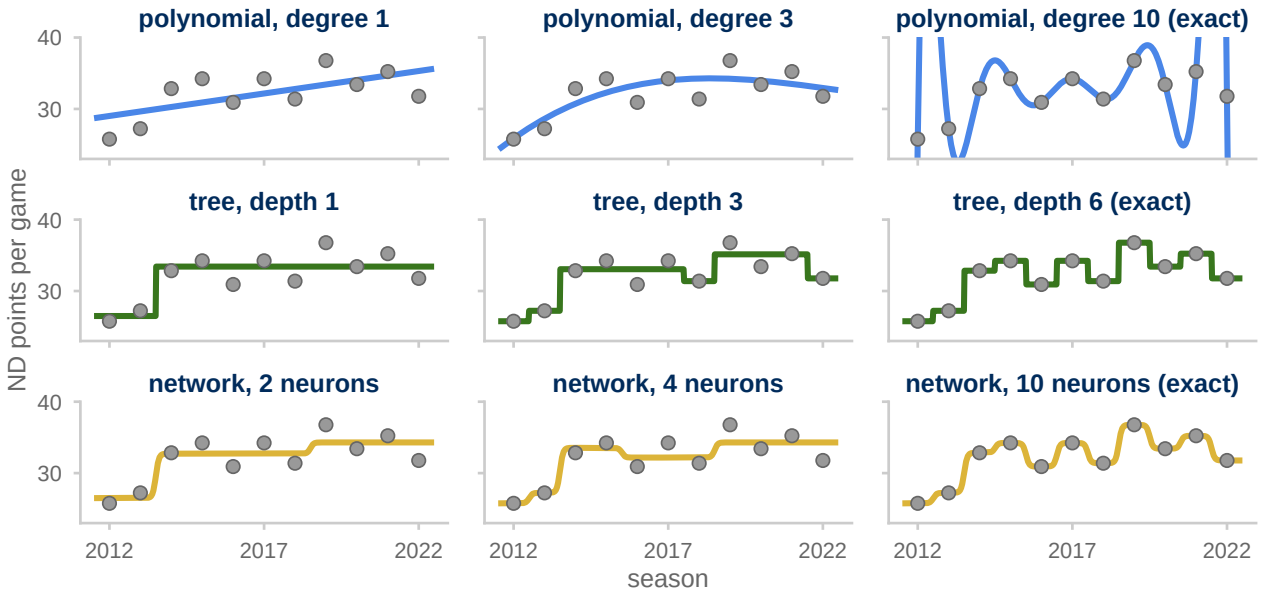
Activation Functions in Real Neurons

A real neuron stays **silent below its threshold**, then fires faster the harder it's pushed, until it levels off.



Near the threshold that's a lot closer to a _____ than to a sigmoid. Glorot et al. (**2011**) used this to argue for ReLU, along with the fact that only about **1–4**% of the brain's neurons are active at once (**sparse**, like a ReLU layer's zeros).

Universal Approximation Theorem



The **universal approximation theorem** says a **neural network** with a single **hidden layer**, given enough **hidden neurons**, can approximate any continuous function

One Neuron, One Network?

Our artificial neuron adds up its inputs once. A real neuron's dendrites branch, and **each branch does its own non-linear sum** before the cell body adds them up.

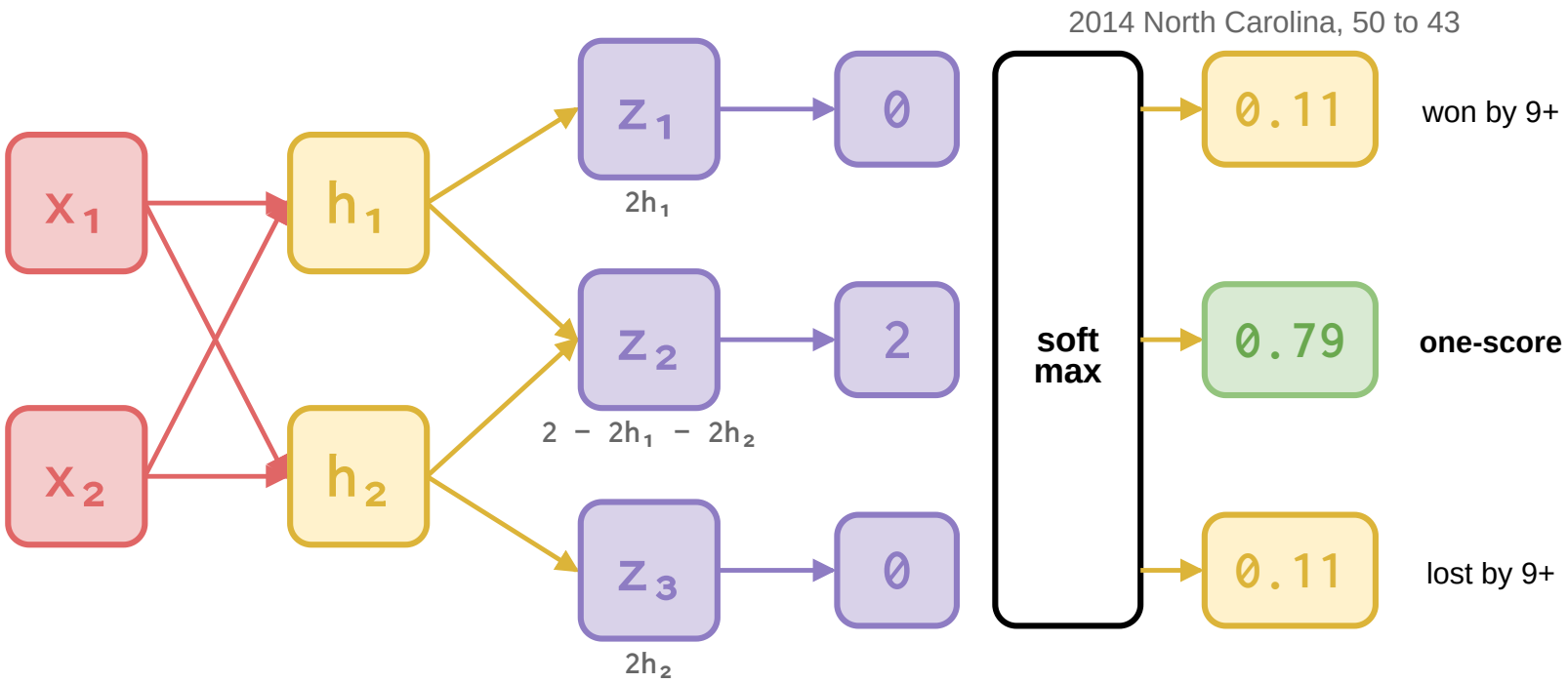
Poirazi, Brannon and Mel (**2003**) modeled a pyramidal neuron this way and found it behaves like a _____ **neural network**, every branch a hidden neuron.

Beniaguev, Segev and London (**2021**) trained networks to copy a detailed model of one cortical neuron, spike for spike. It took a network **7** **layers deep**.

So **one real neuron is already a small network**, and our **3**-neuron FFN is a long way from even one of them.

Multiclass Classification

Another powerful new ability we gain with **neural networks**, is we are no longer limited to binary outputs, we can use them naturally for **multi-class classification**



Multi-class Classification

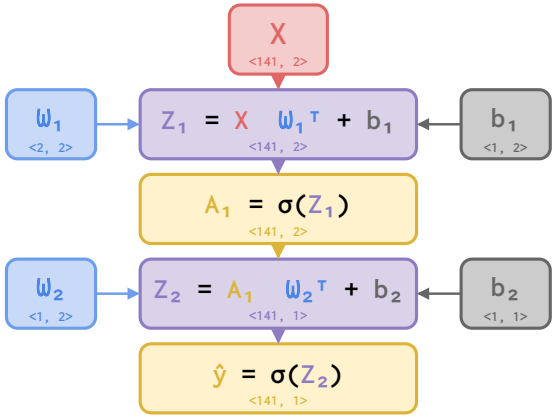
Our three output **scores** aren't **probabilities**. To turn them into probabilities that add up to **1**, we use the _____ function:

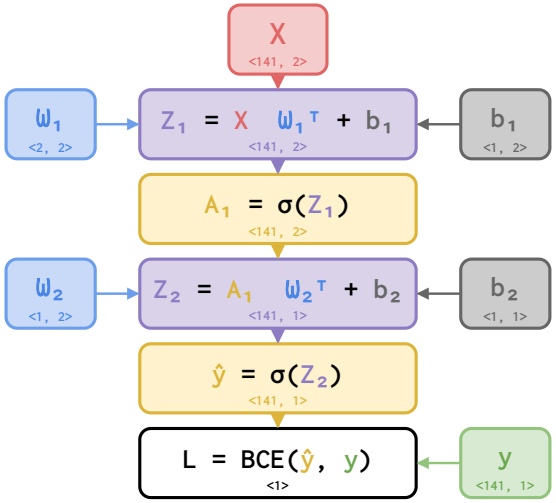
$$\text{softmax}(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

Softmax is the usual **activation function** for the _____ of a **classifier**. With only two classes it's the same as the _____.

Representing Neural Networks

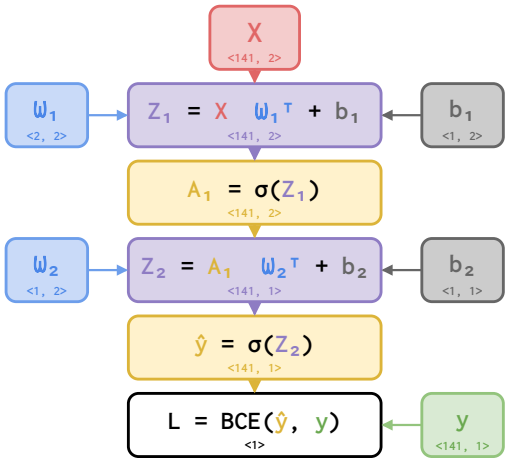
Another common way to represent the **architecture** of a **neural network** is as a _____.





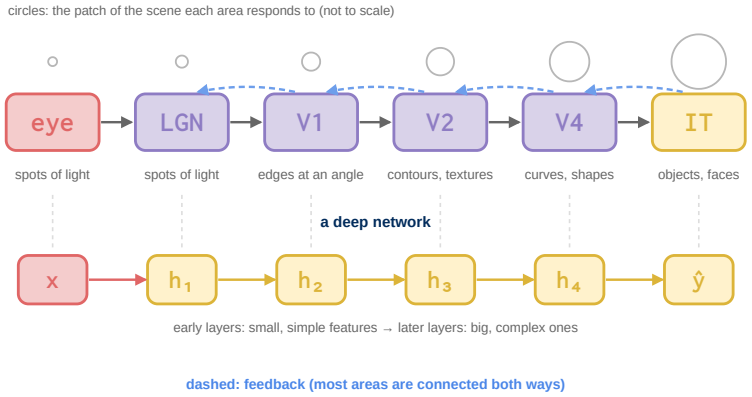
Representing Neural Networks

From the **computation graph**, we can also represent our **neural network** as a single large _____.



This is **critical for training** our **neural networks** (next class)

What FFNs Leave Out



- _____: most areas are connected both ways, and only about **5–10**% of the input to the LGN comes from the eye (most of the rest comes back down from the cortex)
- **Spikes**: real neurons send all-or-nothing spikes, and their timing matters, not just one number
- **Learning**: nobody has found the brain doing what we'll do next lecture (**backprop**)

An FFN is a model of the brain's **first, fast sweep**, not the whole thing.

An Old Exam Question

```python
import torch
import torch.nn as nn
import torch.nn.functional as F

class SimpleNetwork(nn.Module):
    def __init__(self):
        super(SimpleNetwork, self).__init__()
        self.fc1 = nn.Linear(2, 32)
        self.fc2 = nn.Linear(32, 16)
        self.fc3 = nn.Linear(16, 1)

    def forward(self, x):
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = torch.sigmoid(self.fc3(x))
        return x

criterion = nn.BCELoss()
```

a.) Draw the computation graph for the network. (4 pts.)

b.) Give the loss of the neural network (L) as a single statement of nested functions. (1 pt.)