Neural Networks
and
Computation Graphs

THE UNIVERSITY OF UTAH

Based on slides and material from Geoffrey Hinton, Richard Socher, Yoav Goldberg, Chris Dyer, Graham Neubig and others.

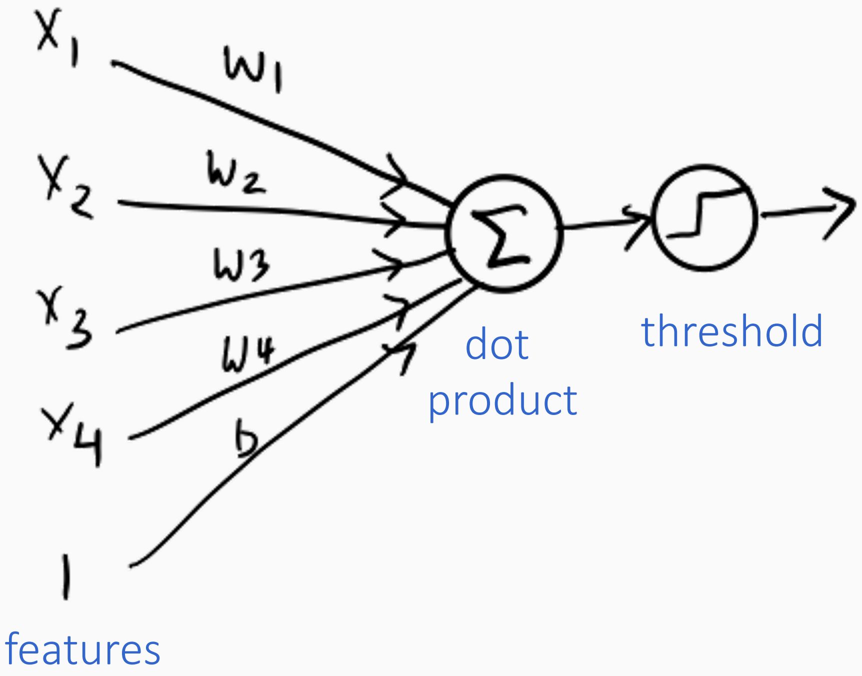
This lecture

- What is a neural network?
- Computation Graphs
- Algorithms over computation graphs
  - The forward pass
  - The backward pass

Where are we?

- What is a neural network?
  - A quick refresher
- Computation Graphs
- Algorithms over computation graphs
  - The forward pass
  - The backward pass

Linear classifiers assign weights to features



Prediction

$$sgn\ (\boldsymbol{w}^T\boldsymbol{x} + b) = sgn(\sum w_i x_i + b)$$

Learning

various algorithms

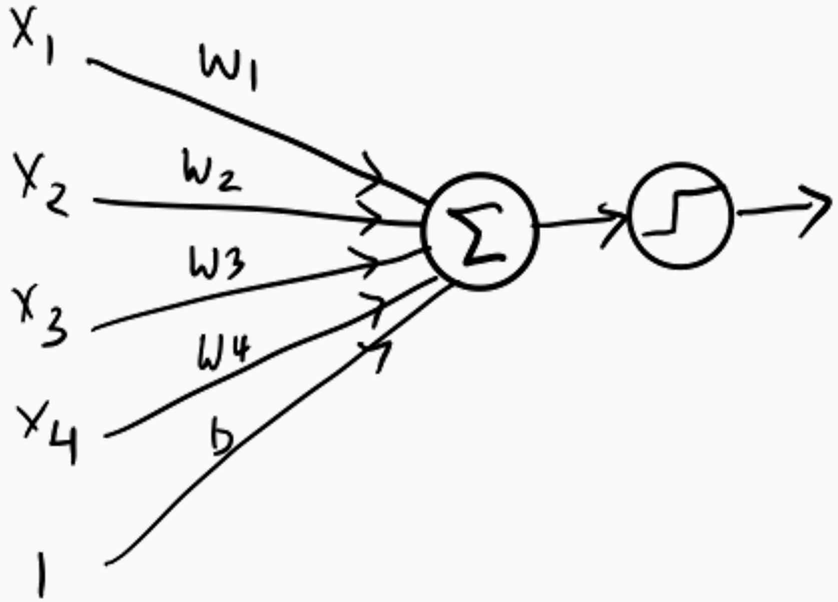
perceptron, SVM, logistic regression,…

in general, minimize loss

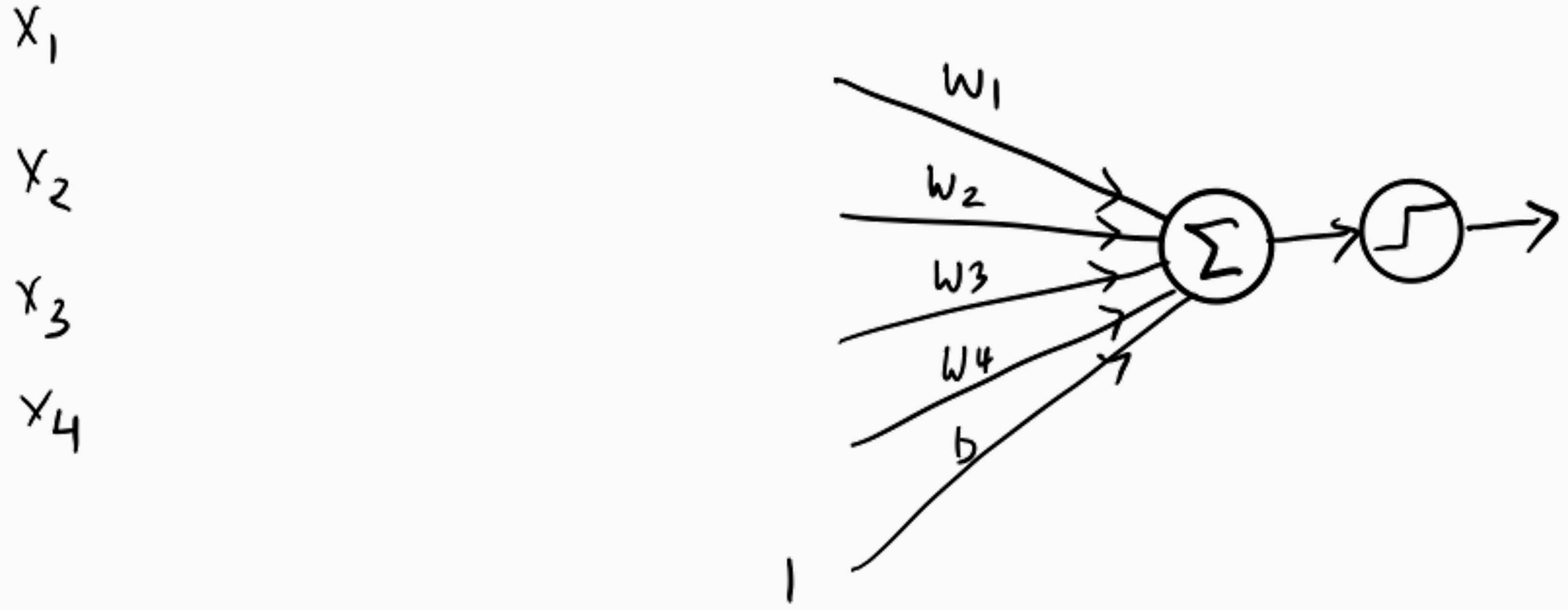
But where do these input features come from?

What if the features were outputs of another classifier?

Features from classifiers

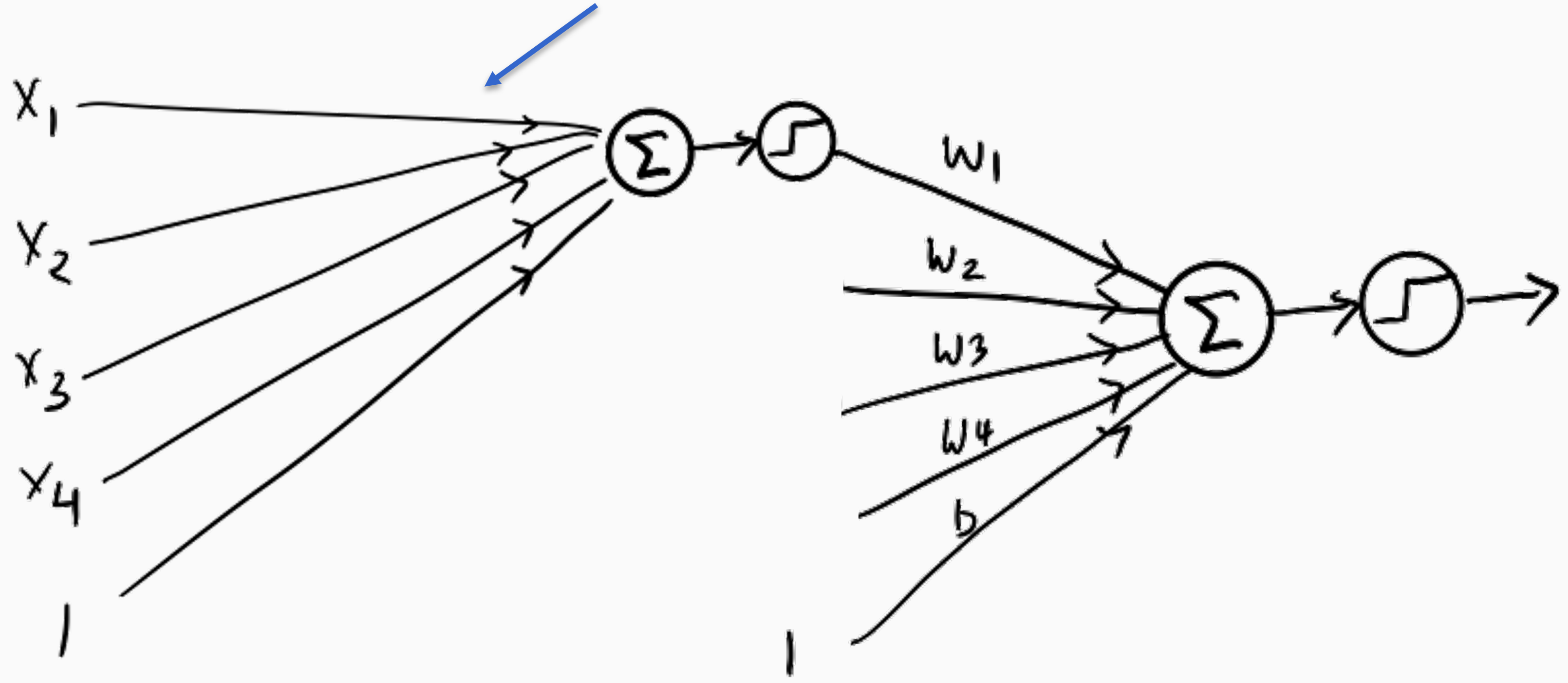


Features from classifiers

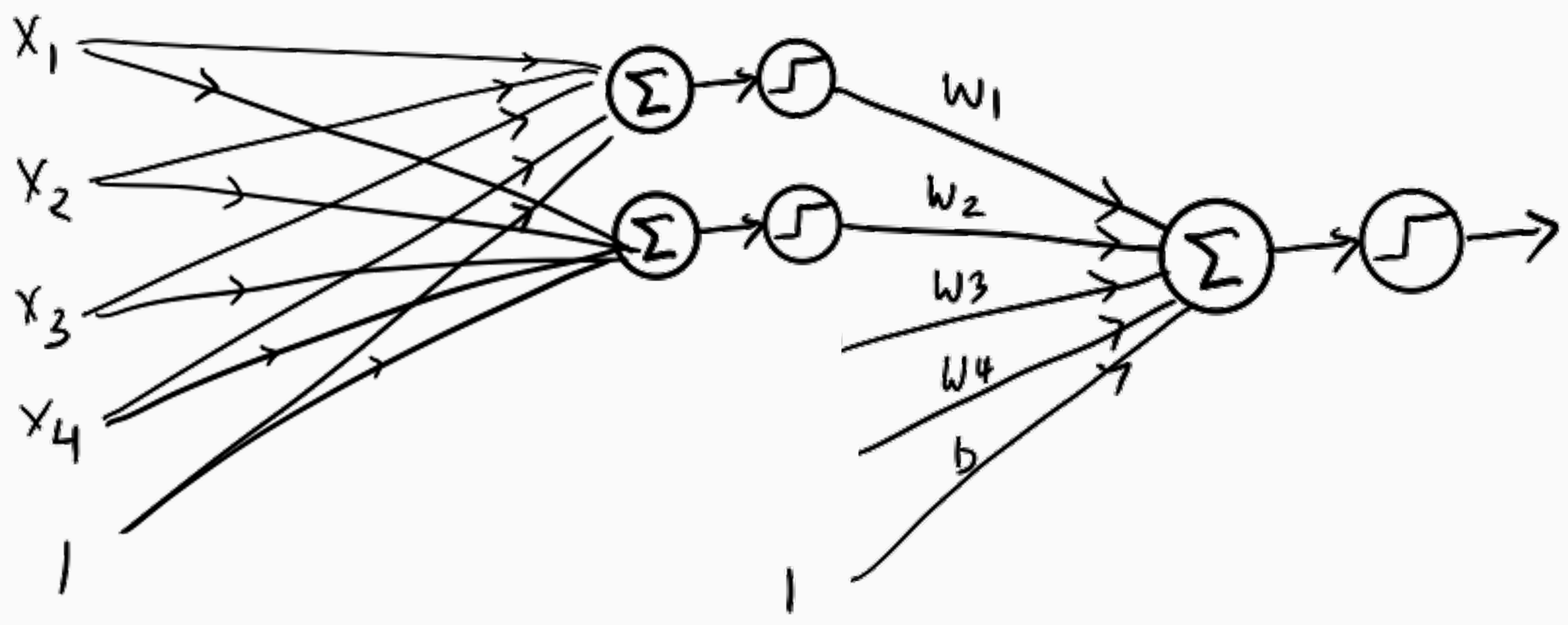


Features from classifiers

Each of these connections have their own weights as well

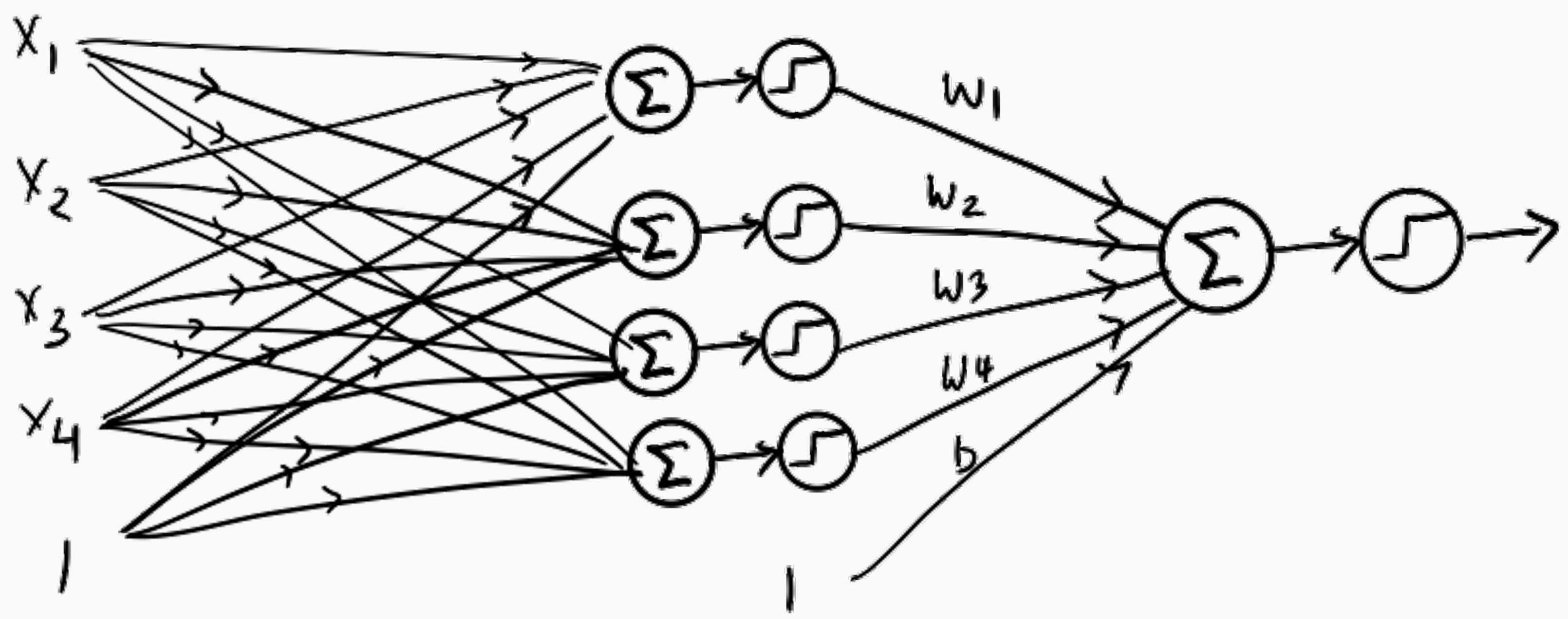


Features from classifiers



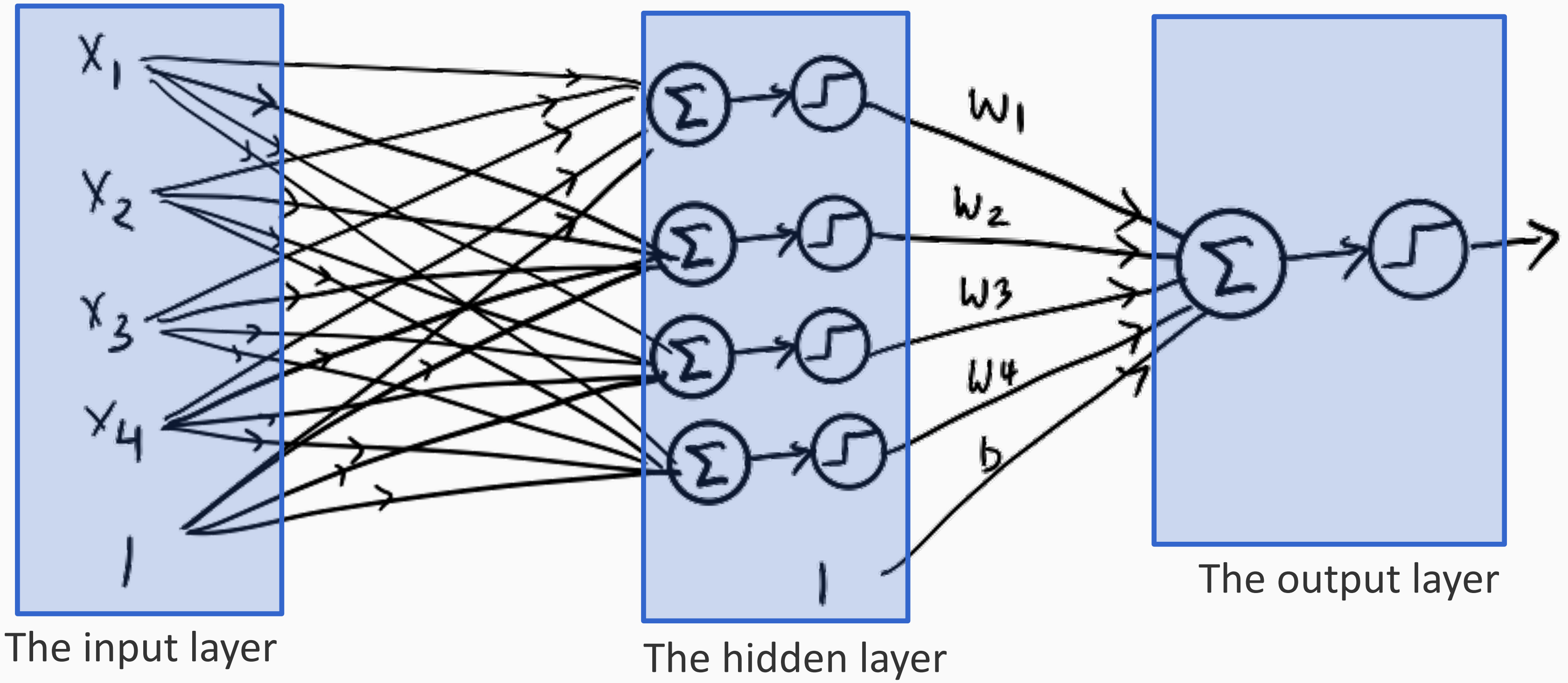
Features from classifiers

This is a **two layer** feed forward neural network



Features from classifiers

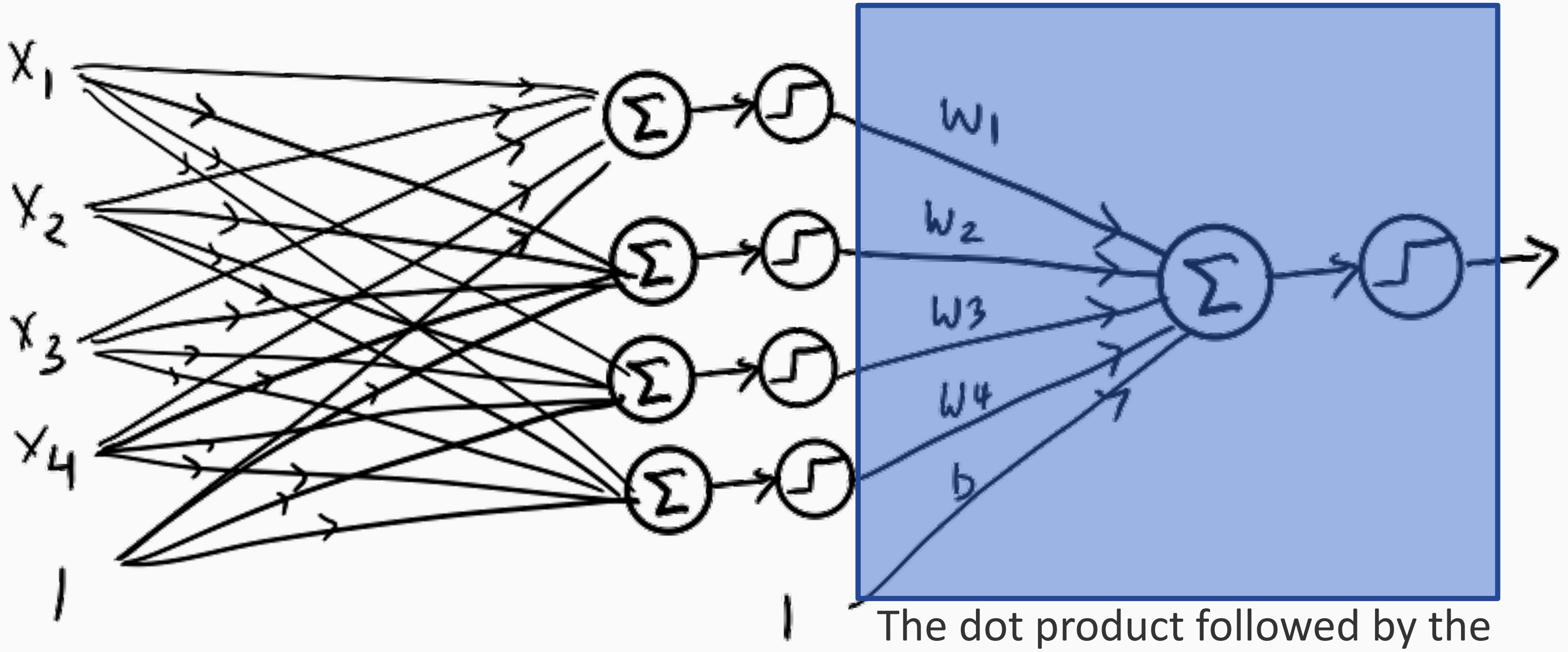
This is a **two layer** feed forward neural network



Think of the hidden layer as learning a good **representation** of the inputs

Features from classifiers

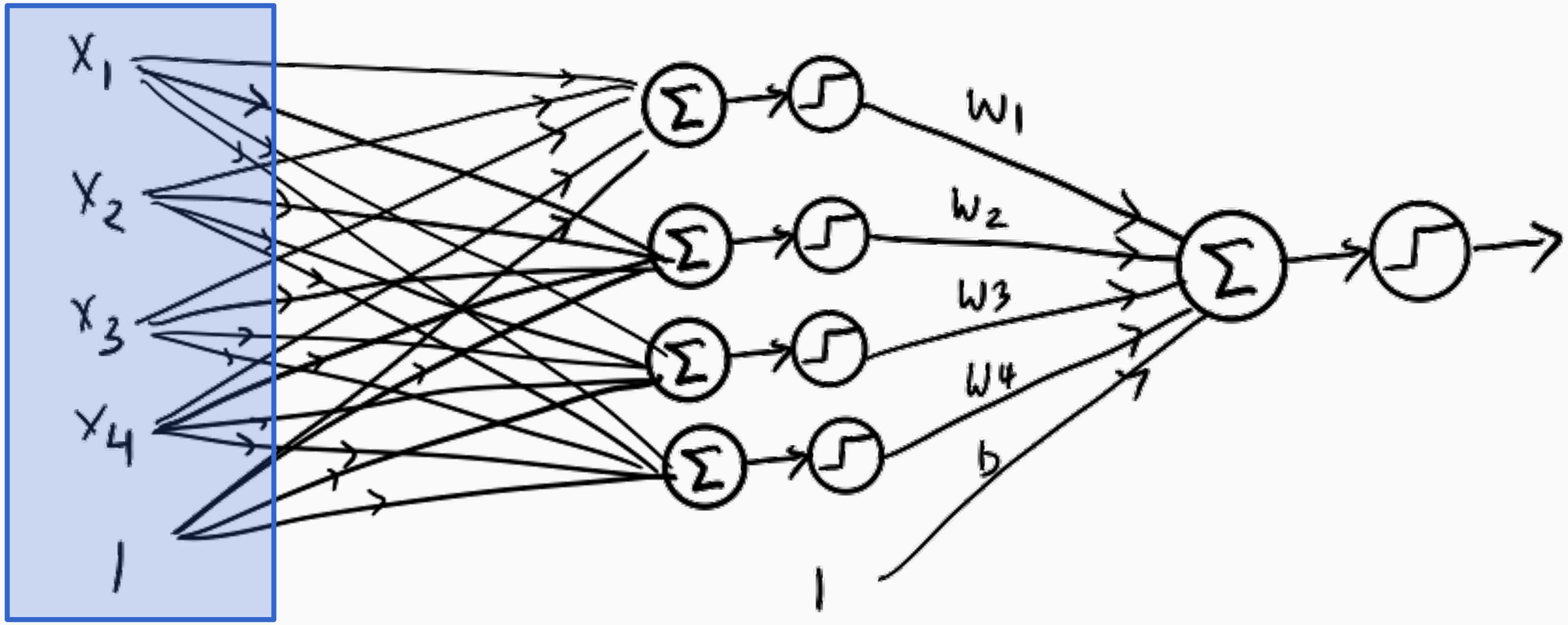
This is a **two layer** feed forward neural network



The dot product followed by the threshold constitutes a ***neuron***

Five neurons in this picture (four in hidden layer and one output)

But where do the inputs come from?



The input layer

What if the inputs were the outputs of a classifier?

We can make a **three** layer network…. And so on.

Let us try to formalize this

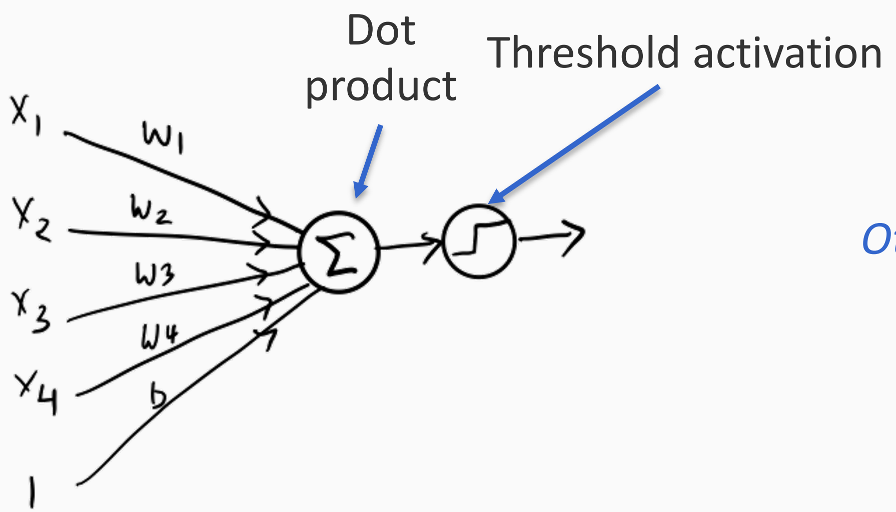
Artificial neurons

*Functions that **very loosely** mimic a biological neuron*

A neuron accepts a collection of inputs (a vector **x**) and produces an output by:

- Applying a dot product with weights **w** and adding a bias b
- Applying a (possibly non-linear) transformation called an ***activation***

$$output = activation(\boldsymbol{w}^T\boldsymbol{x} + b)$$



Other activations are possible

Activation functions

Also called transfer functions

$$output = activation(\boldsymbol{w}^T \boldsymbol{x} + b)$$

Name of the neuron	**Activation function:** $activation(z)$
Linear unit	z
Threshold/sign unit	$\mathrm{sgn}(z)$
Sigmoid unit	$\frac{1}{1 + \exp(-z)}$
Rectified linear unit (ReLU)	$\max(0, z)$
Tanh unit	$\tanh(z)$

Many more activation functions exist (sinusoid, sinc, gaussian, polynomial…)

A neural network

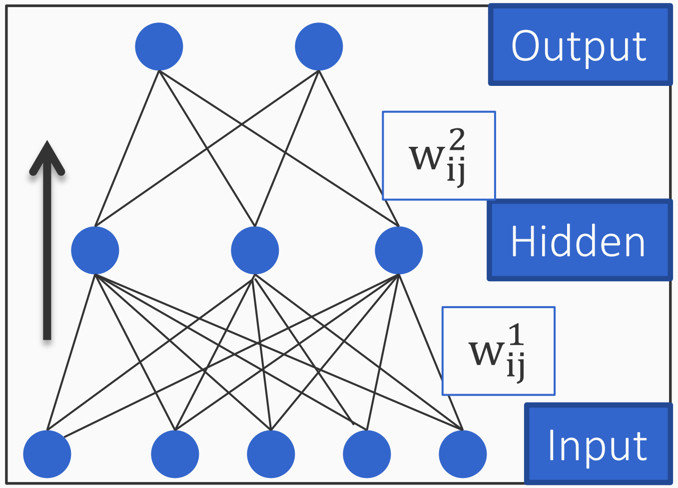
A function that converts inputs to outputs defined by a directed acyclic graph

- Nodes organized in layers, correspond to neurons
- Edges carry output of one neuron to another, associated with weights

A neural network

A function that converts inputs to outputs defined by a directed acyclic graph

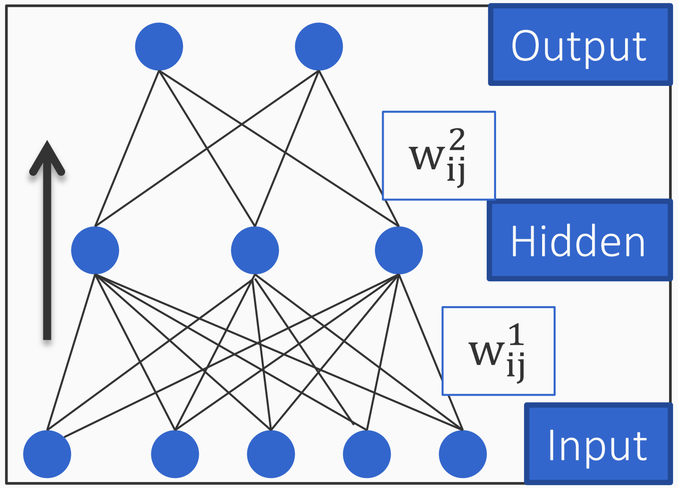
- Nodes organized in layers, correspond to neurons
- Edges carry output of one neuron to another, associated with weights



A neural network

A function that converts inputs to outputs defined by a directed acyclic graph

– Nodes organized in layers, correspond to neurons
– Edges carry output of one neuron to another, associated with weights



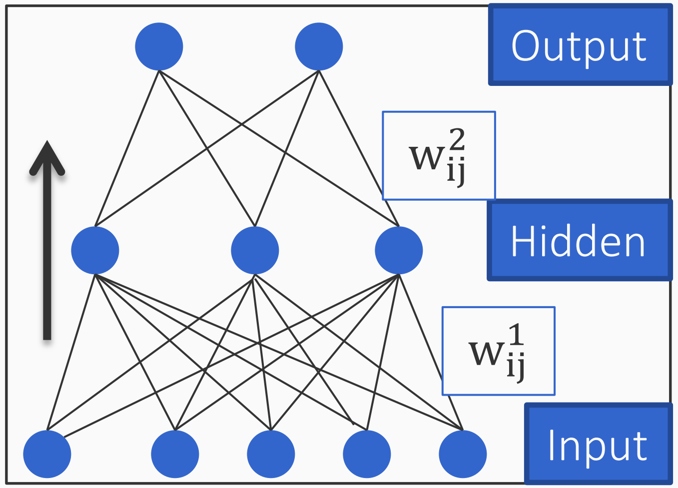
To define a neural network, we need to specify:

– The structure of the graph
  • How many nodes, the connectivity
– The activation function on each node
– The edge weights

A neural network

A function that converts inputs to outputs defined by a directed acyclic graph

- Nodes organized in layers, correspond to neurons
- Edges carry output of one neuron to another, associated with weights



To define a neural network, we need to specify:

- The structure of the graph
  - How many nodes, the connectivity
- The activation function on each node
- The edge weights

Called the *architecture* of the network
Typically predefined, part of the design of the classifier

Learned from data

A brief history of neural networks

- 1943: McCullough and Pitts showed how linear threshold units can compute logical functions
- 1949: Hebb suggested a learning rule that has some physiological plausibility
- 1950s: Rosenblatt, the Peceptron algorithm for a single threshold neuron
- 1969: Minsky and Papert studied the neuron from a geometrical perspective
- 1980s: Convolutional neural networks (Fukushima, LeCun), the backpropagation algorithm (various)
- 2003-today: More compute, more data, deeper networks
- 2012: AlexNet (a CNN) convincingly beats all non-neural methods
- 2017-today: The transformer era

Neural networks are universal function approximators

- Any continuous function can be approximated to arbitrary accuracy using one hidden layer of sigmoid units [Cybenko 1989]
- Approximation error is insensitive to the choice of activation functions [DasGupta et al 1993]
- Two layer threshold networks can express *any* Boolean function
  - **Exercise**: Prove this
- VC dimension of threshold network with edges E: $VC = O(|E| \log |E|)$
- VC dimension of sigmoid networks with nodes V and edges E:
  - Upper bound: $O(|V|^2 |E|^2)$
  - Lower bound: $\Omega(|E|^2)$