Neural Networks

- These representations are inspired by neurons and their connections in the brain.
- Artificial neurons, or units, have inputs, and an output. The output can be connected to the inputs of other units.
- The output of a unit is a parameterized non-linear function of its inputs.
- Learning occurs by adjusting parameters to fit data.
- Neural networks can represent an approximation to any function.

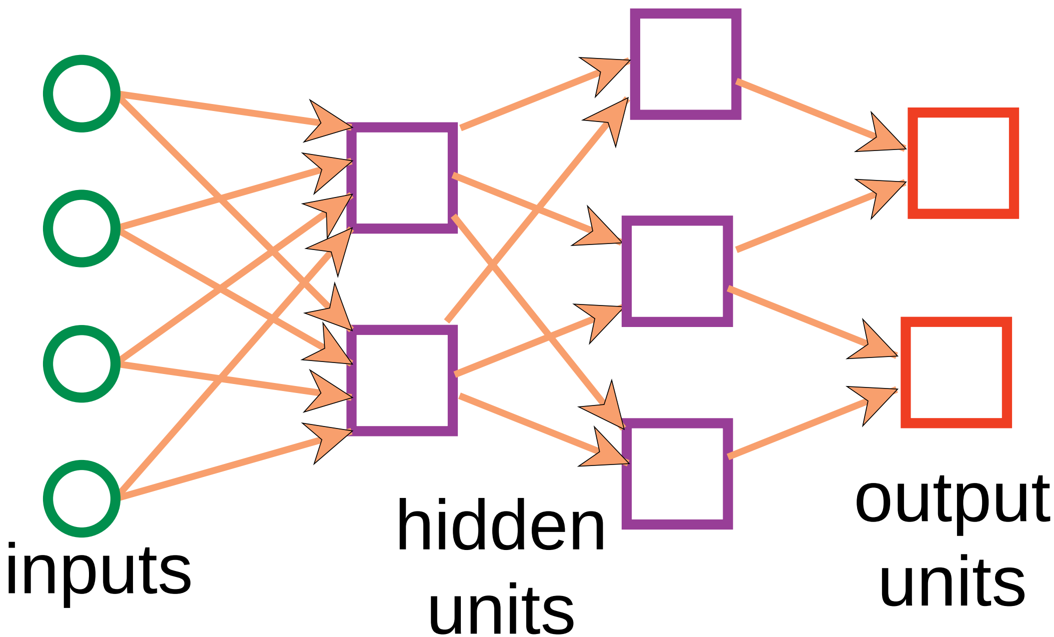
Why Neural Networks?

- As part of neuroscience, in order to understand real neural systems, researchers are simulating the neural systems of simple animals such as worms.
- It seems reasonable to try to build the functionality of the brain via the mechanism of the brain (suitably abstracted).
- The brain inspires new ways to think about computation.
- Neural networks provide a different measure of simplicity as a learning bias.

Feed-forward neural networks

- Feed-forward neural networks are the most common models.

- These are directed acyclic graphs:



The Units

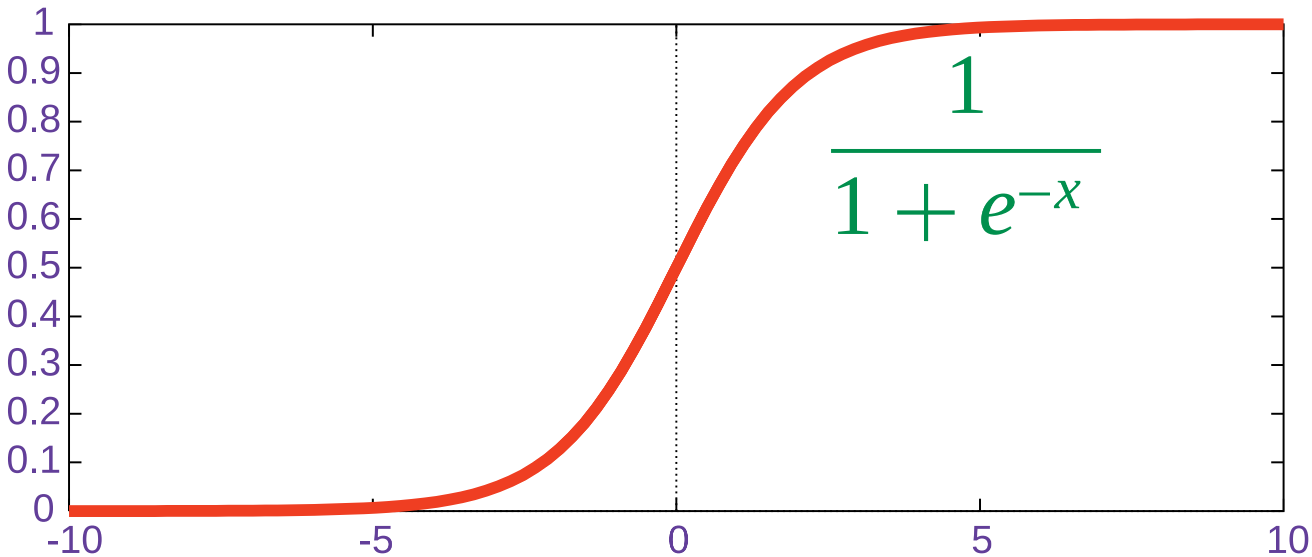
A unit with k inputs is like the parameterized logic program:

$$
\begin{aligned}
&prop(Obj, output, V) \leftarrow \\
&\qquad prop(Obj, in_1, I_1) \wedge \\
&\qquad prop(Obj, in_2, I_2) \wedge \\
&\qquad \ldots \\
&\qquad prop(Obj, in_k, I_k) \wedge \\
&\qquad V \text{ is } f(w_0 + w_1 \times I_1 + w_2 \times I_2 + \cdots + w_k \times I_k).
\end{aligned}
$$

- I_j are real-valued inputs.
- w_j are adjustable real parameters.
- f is an activation function.

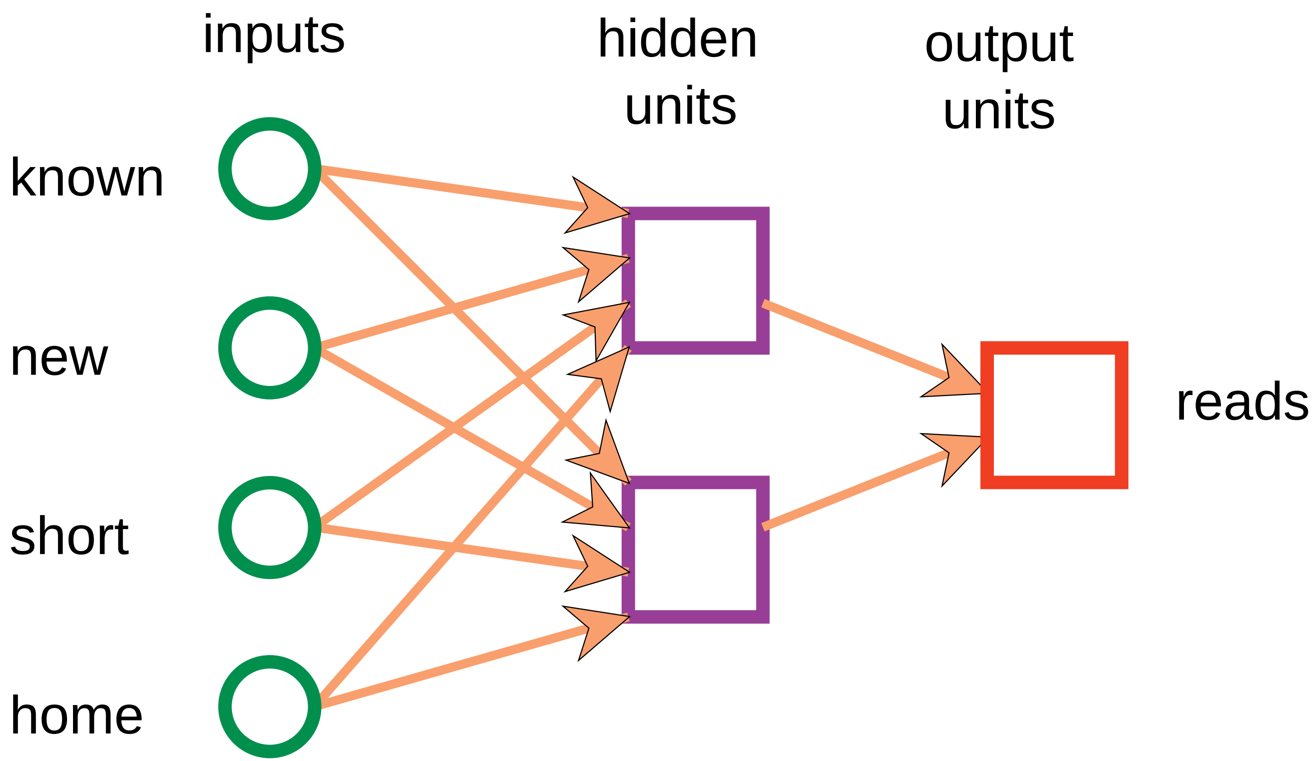
Activation function

A typical activation function is the sigmoid function:



$$f(x) = \frac{1}{1+e^{-x}} \qquad f'(x) = f(x)(1-f(x))$$

Neural Network for the news example



Axiomatizing the Network

- The values of the attributes are real numbers.
- Thirteen parameters $w_0, \ldots, w_{12}$ are real numbers.
- The attributes h_1 and h_2 correspond to the values of hidden units.
- There are 13 real numbers to be learned. The hypothesis space is thus a 13-dimensional real space.
- Each point in this 13-dimensional space corresponds to a particular logic program that predicts a value for *reads* given *known*, *new*, *short*, and *home*.

$$
\begin{aligned}
&\mathit{predicted\_prop}(\mathit{Obj}, \mathit{reads}, V) \leftarrow \\
&\qquad \mathit{prop}(\mathit{Obj}, h_1, I_1) \wedge \mathit{prop}(\mathit{Obj}, h_2, I_2) \wedge \\
&\qquad V \text{ is } f(w_0 + w_1 \times I_1 + w_2 \times I_2). \\
&\mathit{prop}(\mathit{Obj}, h_1, V) \leftarrow \\
&\qquad \mathit{prop}(\mathit{Obj}, \mathit{known}, I_1) \wedge \mathit{prop}(\mathit{Obj}, \mathit{new}, I_2) \wedge \\
&\qquad \mathit{prop}(\mathit{Obj}, \mathit{short}, I_3) \wedge \mathit{prop}(\mathit{Obj}, \mathit{home}, I_4) \wedge \\
&\qquad V \text{ is } f(w_3 + w_4 \times I_1 + w_5 \times I_2 + w_6 \times I_3 + w_7 \times I_4). \\
&\mathit{prop}(\mathit{Obj}, h_2, V) \leftarrow \\
&\qquad \mathit{prop}(\mathit{Obj}, \mathit{known}, I_1) \wedge \mathit{prop}(\mathit{Obj}, \mathit{new}, I_2) \wedge \\
&\qquad \mathit{prop}(\mathit{Obj}, \mathit{short}, I_3) \wedge \mathit{prop}(\mathit{Obj}, \mathit{home}, I_4) \wedge \\
&\qquad V \text{ is } f(w_8 + w_9 \times I_1 + w_{10} \times I_2 + w_{11} \times I_3 + w_{12} \times I_4)
\end{aligned}
$$

Prediction Error

- For particular values for the parameters $\overline{w} = w_0, \ldots w_m$ and a set E of examples, the sum-of-squares error is

$$Error_E(\overline{w}) = \sum_{e \in E} (p_e^{\overline{w}} - o_e)^2,$$

  - $p_e^{\overline{w}}$ is the predicted output by a neural network with parameter values given by $\overline{w}$ for example e
  - o_e is the observed output for example e.

- The aim of neural network learning is, given a set of examples, to find parameter settings that minimize the error.

Neural Network Learning

- Aim of neural network learning: given a set of examples, find parameter settings that minimize the error.

- Back-propagation learning is gradient descent search through the parameter space to minimize the sum-of-squares error.

Backpropagation Learning

- **Inputs:**
  - A network, including all units and their connections
  - Stopping Criteria
  - Learning Rate (constant of proportionality of gradient descent search)
  - Initial values for the parameters
  - A set of classified training data
- **Output:** Updated values for the parameters

Backpropagation Learning Algorithm

- Repeat
  - evaluate the network on each example given the current parameter settings
  - determine the derivative of the error for each parameter
  - change each parameter in proportion to its derivative
- until the stopping criteria is met

Gradient Descent for Neural Net Learning

- At each iteration, update parameter w_i

$$w_i \leftarrow \left(w_i - \eta \frac{\partial error(w_i)}{\partial w_i} \right)$$

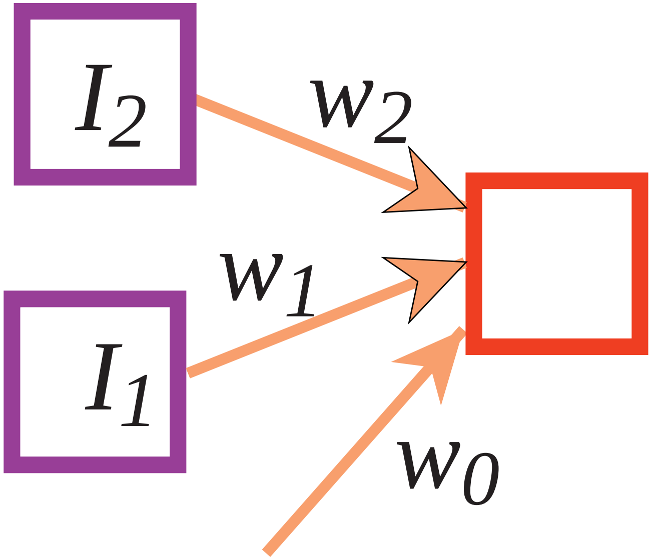
  η is the learning rate

- You can compute partial derivative:
  - numerically: for small Δ
  $$\frac{error(w_i + \Delta) - error(w_i)}{\Delta}$$
  - analytically: $f'(x) = f(x)(1 - f(x))$ + chain rule

Simulation of Neural Net Learning

Parameter	iteration 0		iteration 1	iteration 80
	Value	Deriv	Value	Value
w_0	0.2	0.768	−0.18	−2.98
w_1	0.12	0.373	−0.07	6.88
w_2	0.112	0.425	−0.10	−2.10
w_3	0.22	0.0262	0.21	−5.25
w_4	0.23	0.0179	0.22	1.98
Error:	4.6121		4.6128	0.178

What Can a Neural Network Represent?



w_0	w_1	w_2	Logic
-15	10	10	and
-5	10	10	or
5	-10	-10	nor

Output is $f(w_0 + w_1 \times I_1 + w_2 \times I_2)$.

A single unit can't represent *xor*.

Bias in neural networks and decision trees

- It's easy for a neural network to represent "at least two of $I_1, \ldots, I_k$ are true":

w_0	w_1	$\cdots$	w_k
-15	10	$\cdots$	10

  This concept forms a large decision tree.

- Consider representing a conditional: "If c then a else b":
  - Simple in a decision tree.
  - Needs a complicated neural network to represent $(c \wedge a) \vee (\neg c \wedge b)$.

Neural Networks and Logic

- Meaning is attached to the input and output units.
- There is no a priori meaning associated with the hidden units.
- What the hidden units actually represent is something that's learned.