CS532 ANN

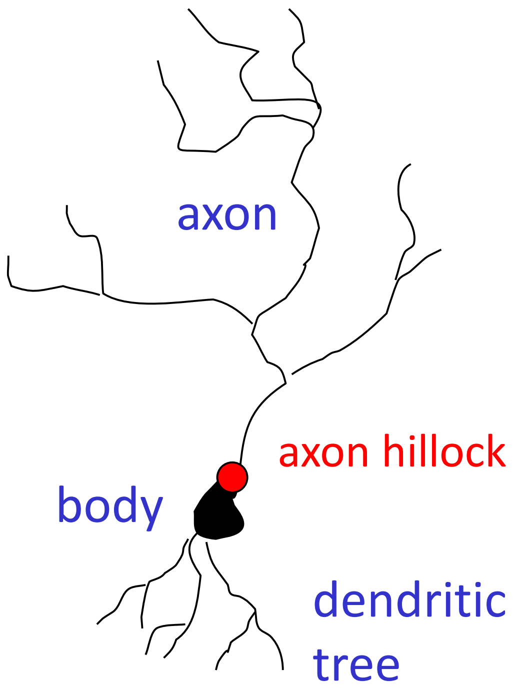
Du barratin autour des réseaux de neurons

WHAT ARE NEURAL NETWORKS?

Credit: G. E. Hinton

A typical cortical neuron

- Gross physical structure:
  - There is one axon that branches
  - There is a dendritic tree that collects input from other neurons.
- Axons typically contact dendritic trees at synapses
  - A spike of activity in the axon causes charge to be injected into the post-synaptic neuron.
- Spike generation:
  - There is an axon hillock that generates outgoing spikes whenever enough charge has flowed in at synapses to depolarize the cell membrane.



Synapses

- When a spike of activity travels along an axon and arrives at a synapse it causes vesicles of transmitter chemical to be released.
  - There are several kinds of transmitter.

- The transmitter molecules diffuse across the synaptic cleft and bind to receptor molecules in the membrane of the post-synaptic neuron thus changing their shape.
  - This opens up holes that allow specific ions in or out.

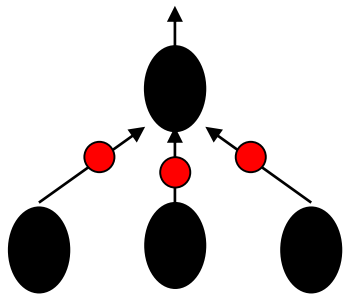
How synapses adapt

- The effectiveness of the synapse can be changed:
  - vary the number of vesicles of transmitter.
  - vary the number of receptor molecules.

- Synapses are slow, but they have advantages over RAM
  - They are very small and very low-power.
  - They adapt using locally available signals
    - But what rules do they use to decide how to change?

How the brain works on one slide!

- Each neuron receives inputs from other neurons
  - A few neurons also connect to receptors.
  - Cortical neurons use spikes to communicate.
- The effect of each input line on the neuron is controlled by a synaptic weight
  - The weights can be positive or negative.
- The synaptic weights adapt so that the whole network learns to perform useful computations
  - Recognizing objects, understanding language, making plans, controlling the body.
- You have about 10^{11} neurons each with about 10^{4} weights.
  - A huge number of weights can affect the computation in a very short time. Much better bandwidth than a workstation.



Modularity and the brain

- Different bits of the cortex do different things.
  – Local damage to the brain has specific effects.
  – Specific tasks increase the blood flow to specific regions.
- But cortex looks pretty much the same all over.
  – Early brain damage makes functions relocate.
- Cortex is made of general purpose stuff that has the ability to turn into special purpose hardware in response to experience.
  – This gives rapid parallel computation plus flexibility.
  – Conventional computers get flexibility by having stored sequential programs, but this requires very fast central processors to perform long sequential computations.

SOME SIMPLE MODELS OF NEURONS

Idealized neurons

- To model things we have to idealize them (e.g. atoms)
  - Idealization removes complicated details that are not essential for understanding the main principles.
  - It allows us to apply mathematics and to make analogies to other, familiar systems.
  - Once we understand the basic principles, its easy to add complexity to make the model more faithful.
- It is often worth understanding models that are known to be wrong (but we must not forget that they are wrong!)
  - E.g. neurons that communicate real values rather than discrete spikes of activity.

Types of Activation functions

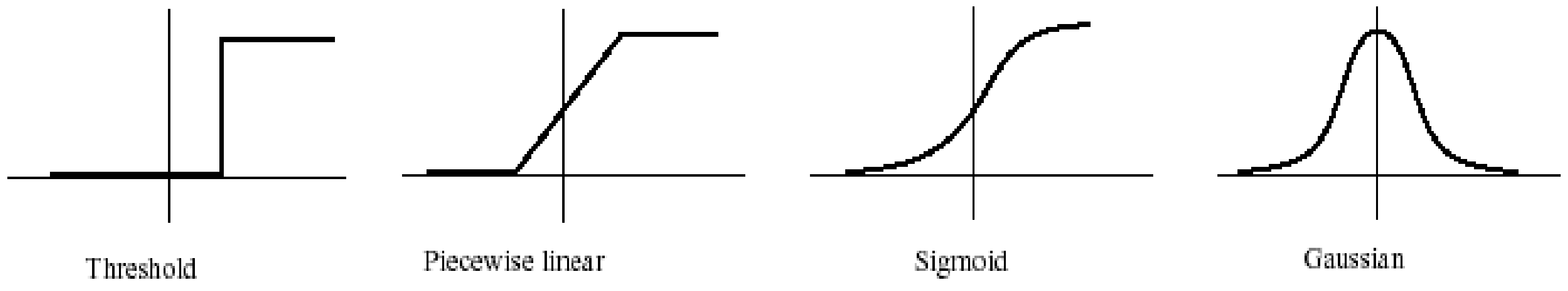


Figure 5: Different types of activation functions.

Linear neurons

- These are simple but computationally limited
  - If we can make them learn we may get insight into more complicated neurons.

bias

i th input

$$y = b + \sum_i x_i w_i$$

output

index over input connections

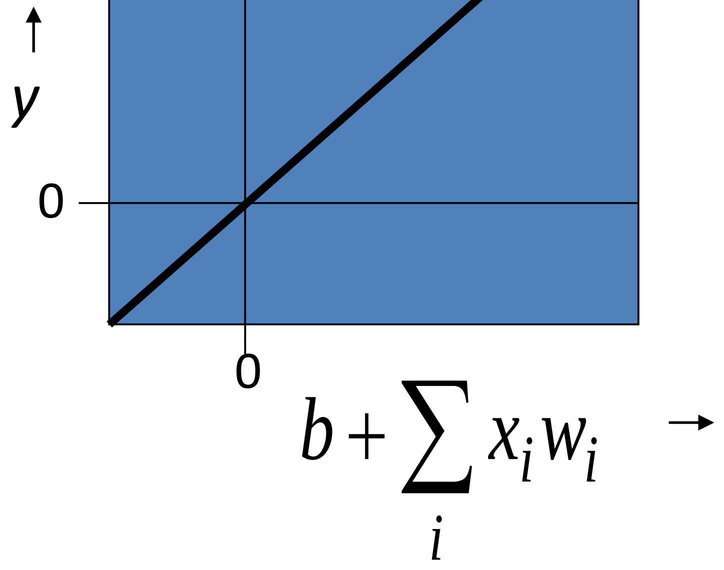
weight on

i th input

Linear neurons

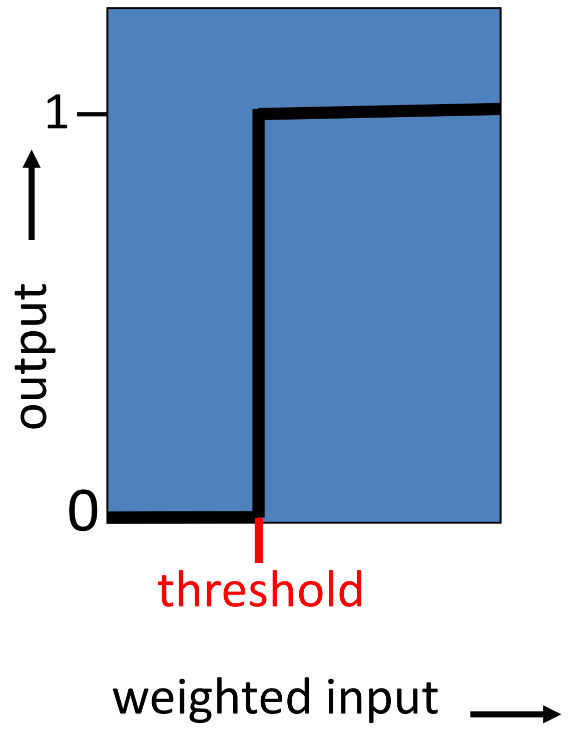
- These are simple but computationally limited
  - If we can make them learn we may get insight into more complicated neurons.

$$y = b + \sum_i x_i w_i$$



Binary threshold neurons

- McCulloch-Pitts (1943): influenced Von Neumann.
  - First compute a weighted sum of the inputs.
  - Then send out a fixed size spike of activity if the weighted sum exceeds a threshold.
  - McCulloch and Pitts thought that each spike is like the truth value of a proposition and each neuron combines truth values to compute the truth value of another proposition!



Binary threshold neurons

- There are two equivalent ways to write the equations for a binary threshold neuron:

$$z = \sum_i x_i w_i$$

$$y = \begin{cases} 1 \text{ if } & z \geq \theta \\ 0 \text{ otherwise} & \end{cases}$$

$$\boxed{q = - b}$$

$$z = b + \underset{i}{\text{å}}\, x_i w_i$$

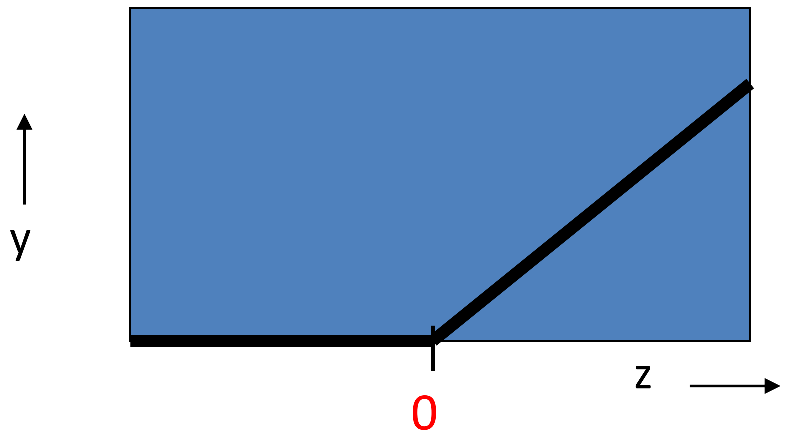
$$y = \begin{cases} 1 \text{ if } & z^3\ 0 \\ 0 \text{ otherwise} & \end{cases}$$

Rectified Linear Neurons
(sometimes called linear threshold neurons)

They compute a linear weighted sum of their inputs.
The output is a non-linear function of the total input.

$$z = b + \sum_i x_i w_i$$

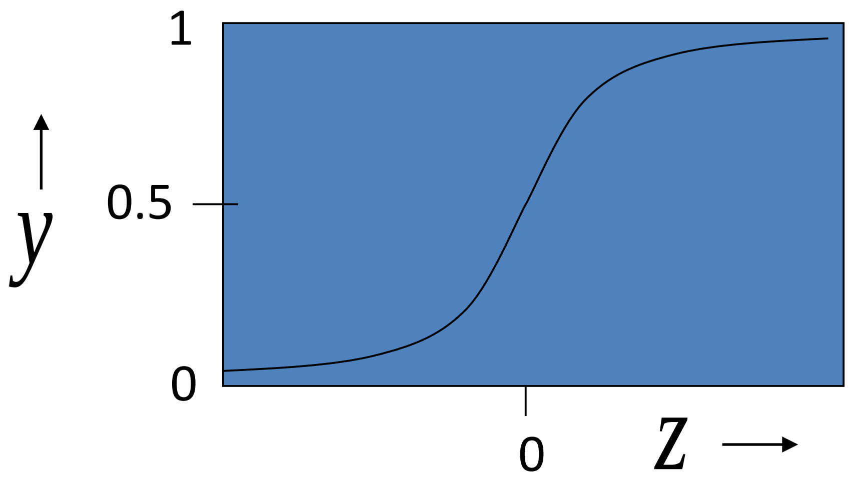
$$y = \begin{cases} z & \text{if } z > 0 \\ 0 & \text{otherwise} \end{cases}$$



Sigmoid neurons

- These give a real-valued output that is a smooth and bounded function of their total input.
  - Typically they use the logistic function
  - They have nice derivatives which make learning easy (see lecture 3).

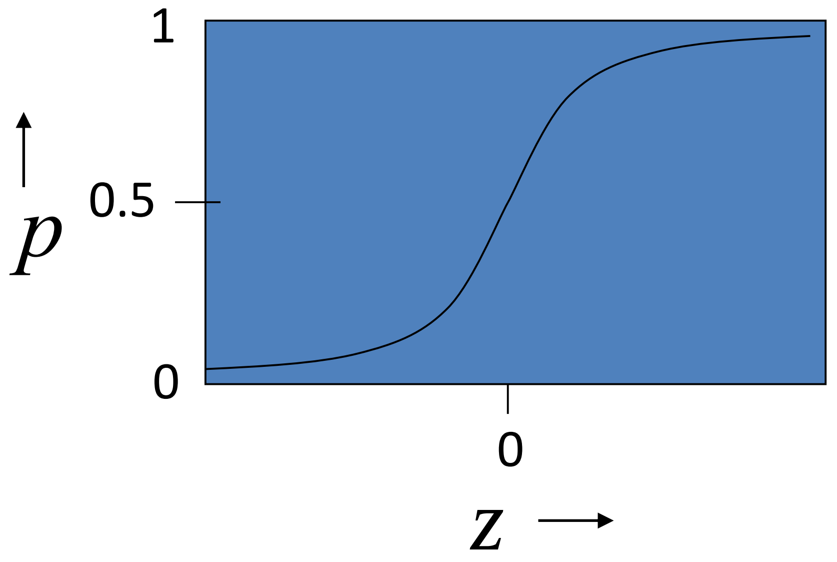
$$z = b + \sum_i x_i w_i \qquad y = \frac{1}{1 + e^{-z}}$$



Stochastic binary neurons

- These use the same equations as logistic units.
  - But they treat the output of the logistic as the probability of producing a spike in a short time window.
- We can do a similar trick for rectified linear units:
  - The output is treated as the Poisson rate for spikes.

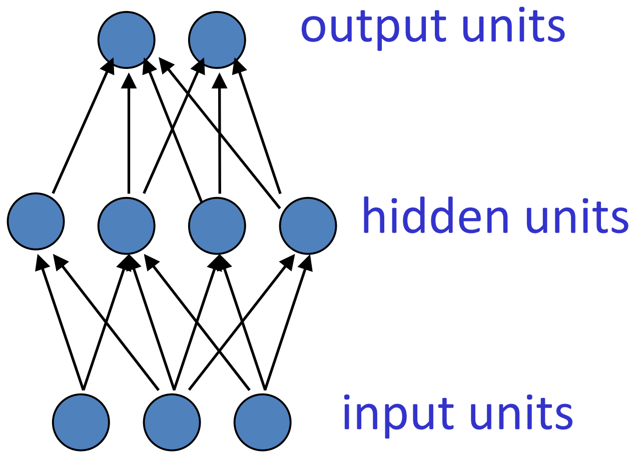
$$z = b + \sum_i x_i w_i \qquad p(s=1) = \frac{1}{1+e^{-z}}$$



ANN ARCHITECTURES

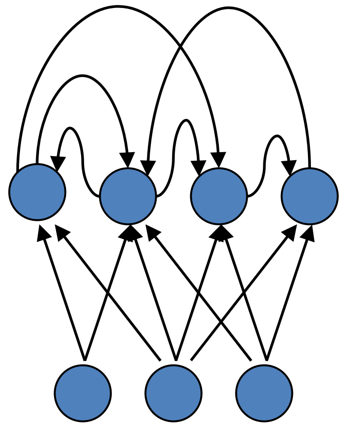
Feed-forward neural networks

- These are the commonest type of neural network in practical applications.
  - The first layer is the input and the last layer is the output.
  - If there is more than one hidden layer, we call them “deep” neural networks.
- They compute a series of transformations that change the similarities between cases.
  - The activities of the neurons in each layer are a non-linear function of the activities in the layer below.



Recurrent networks

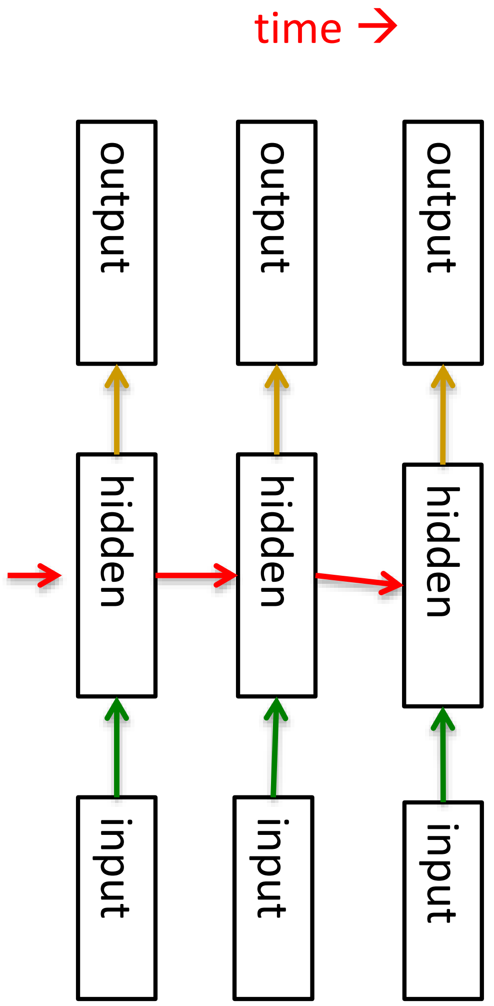
- These have directed cycles in their connection graph.
  - That means you can sometimes get back to where you started by following the arrows.
- They can have complicated dynamics and this can make them very difficult to train.
  - There is a lot of interest at present in finding efficient ways of training recurrent nets.
- They are more biologically realistic.



Recurrent nets with multiple hidden layers are just a special case that has some of the hidden$\rightarrow$hidden connections missing.

Recurrent neural networks for modeling sequences

- Recurrent neural networks are a very natural way to model sequential data:
  - They are equivalent to very deep nets with one hidden layer per time slice.
  - Except that they use the same weights at every time slice and they get input at every time slice.
- They have the ability to remember information in their hidden state for a long time.
  - But its very hard to train them to use this potential.


time →
output
output
output
hidden
hidden
hidden
input
input
input