Lecture 27: Neural Networks and Deep Learning

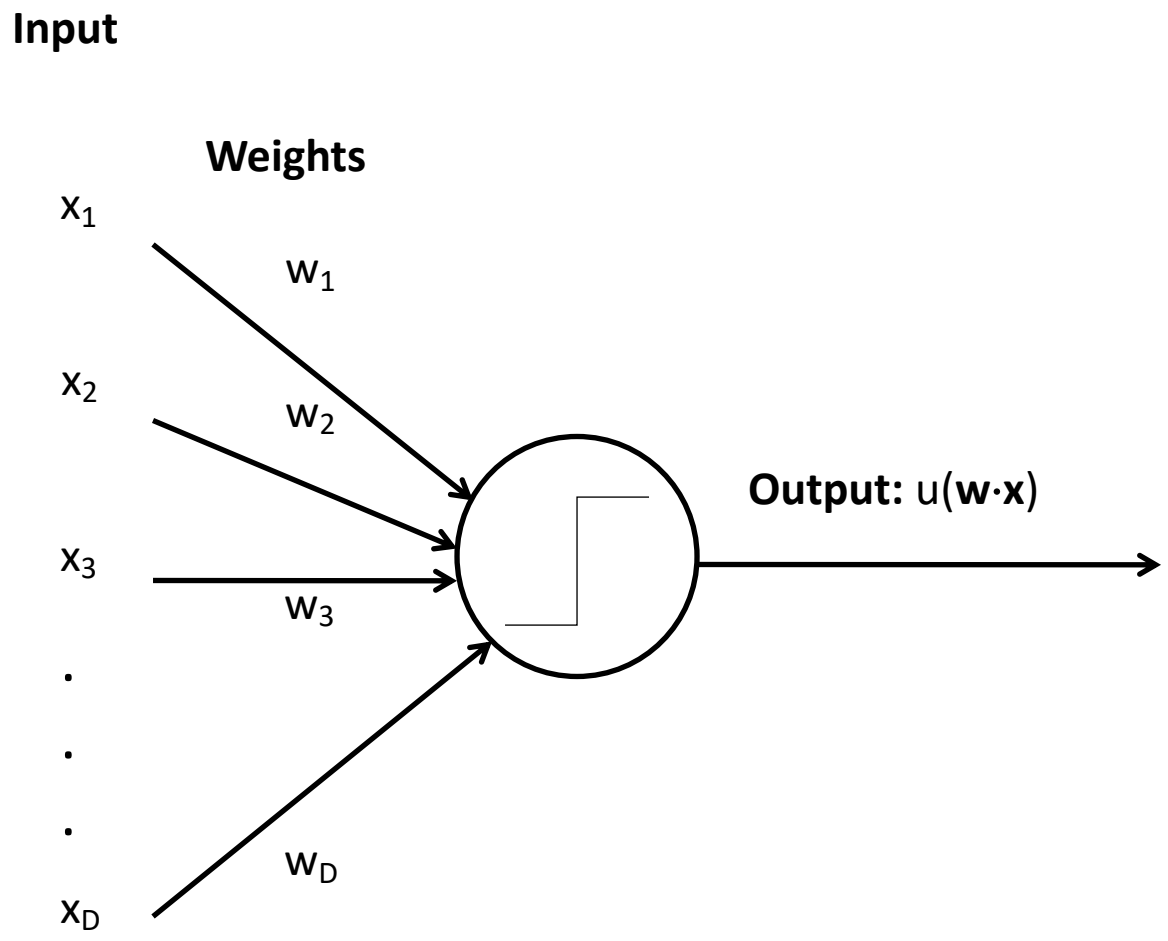
Mark Hasegawa-Johnson

April 6, 2020

Outline

- Why use more than one layer?
  - Biological inspiration
  - Representational power: the XOR function
- Two-layer neural networks
  - The Fundamental Theorem of Calculus
  - Feature learning for linear classifiers
- Deep networks
  - Biological inspiration: features computed from features
  - Flexibility: convolutional, recurrent, and gated architectures

Biological Inspiration: McCulloch-Pitts Artificial Neuron, 1943

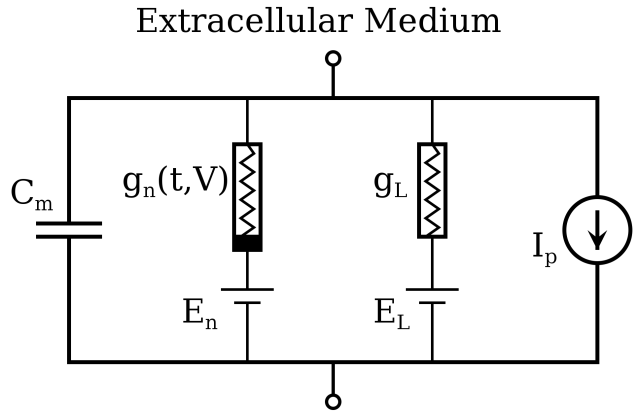


- In 1943, McCulloch & Pitts proposed that biological neurons have a nonlinear activation function (a step function) whose input is a weighted linear combination of the currents generated by other neurons.
- They showed lots of examples of mathematical and logical functions that could be computed using networks of simple neurons like this.

Biological Inspiration: Hodgkin & Huxley

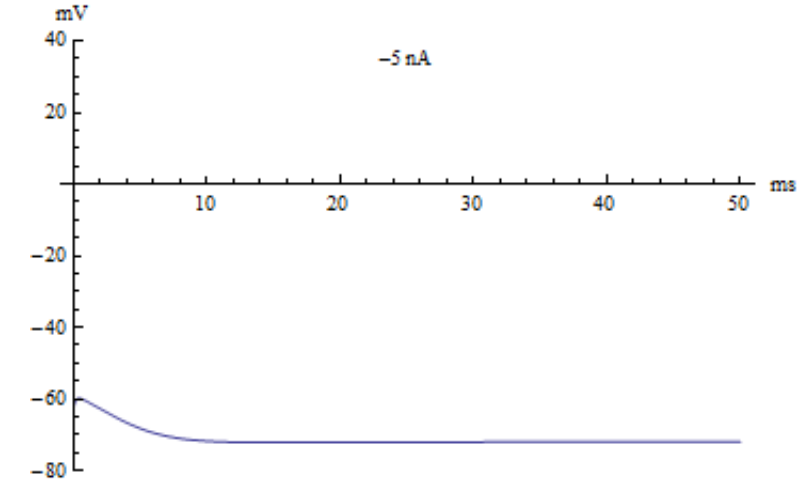
Hodgkin & Huxley won the Nobel prize for their model of cell membranes, which provided lots more detail about how the McCulloch-Pitts model works in nature. Their nonlinear model has two step functions:

- I < threshold1: V= $-75mV$
- threshold1 < I < threshold2: V has a spike, then returns to rest.
- threshold 2 < I: V spikes periodically



Hodgkin & Huxley Circuit Model of a Neuron Membrane

By Krishnavedala - Own work, CC0, https://commons.wikimedia.org/w/index.php?curid=21725464



Membrane voltage versus time. As current passes 0mA, spike appears. As current passes 10mA, spike train appears.

By Alexander J. White - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=30310965

Biological Inspiration: Neuronal Circuits

- Even the simplest actions involve more than one neuron, acting in sequence in a neuronal circuit.
- One of the simplest neuronal circuits is a reflex arc, which may contain just two neurons:
  - The **sensor neuron** detects a stimulus, and communicates an electrical signal to …
  - The **motor neuron**, which activates the muscle.

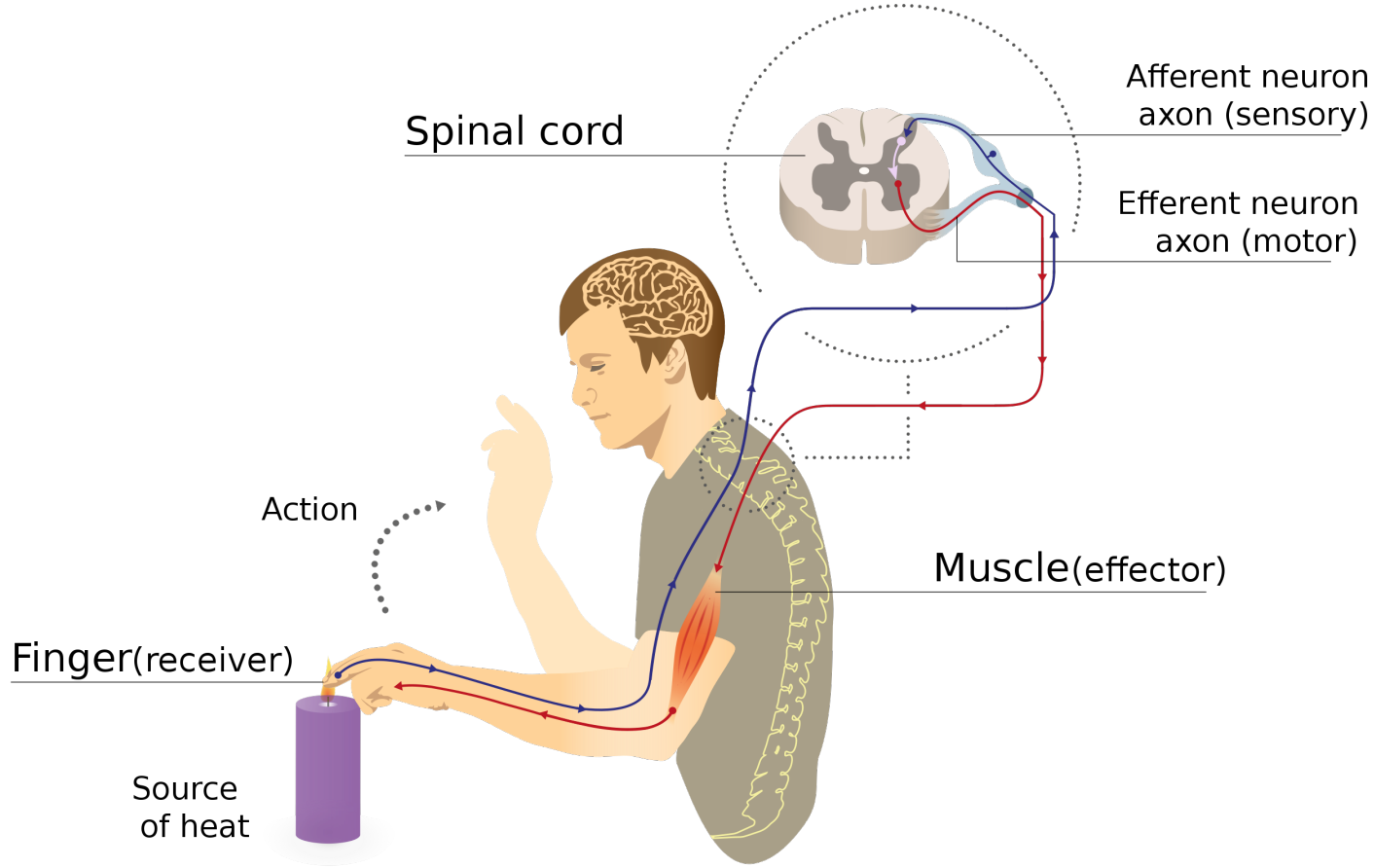


Illustration of a reflex arc: sensor neuron sends a voltage spike to the spinal column, where the resulting current causes a spike in a motor neuron, whose spike activates the muscle.

Biological Inspiration: Neuronal Circuits

- A circuit composed of many neurons can compute the autocorrelation function of an input sound, and from the autocorrelation, can estimate the pitch frequency.
- The circuit depends on output neurons, C, that each compute a step function in response to the sum of two different input neurons, A and B.

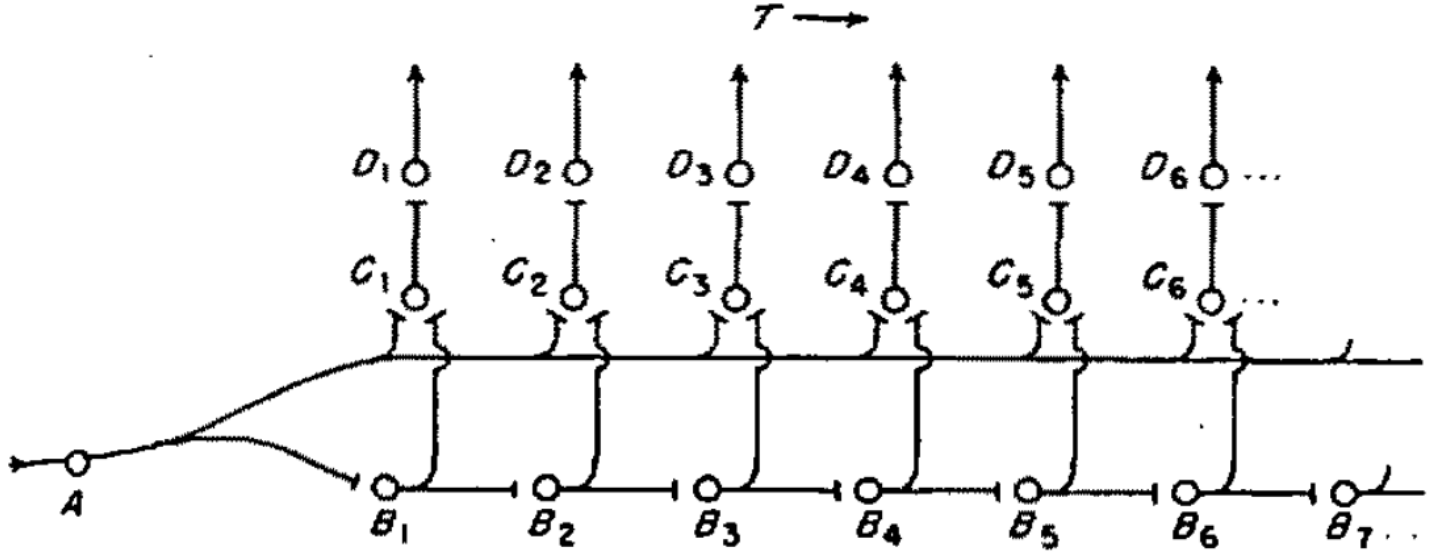
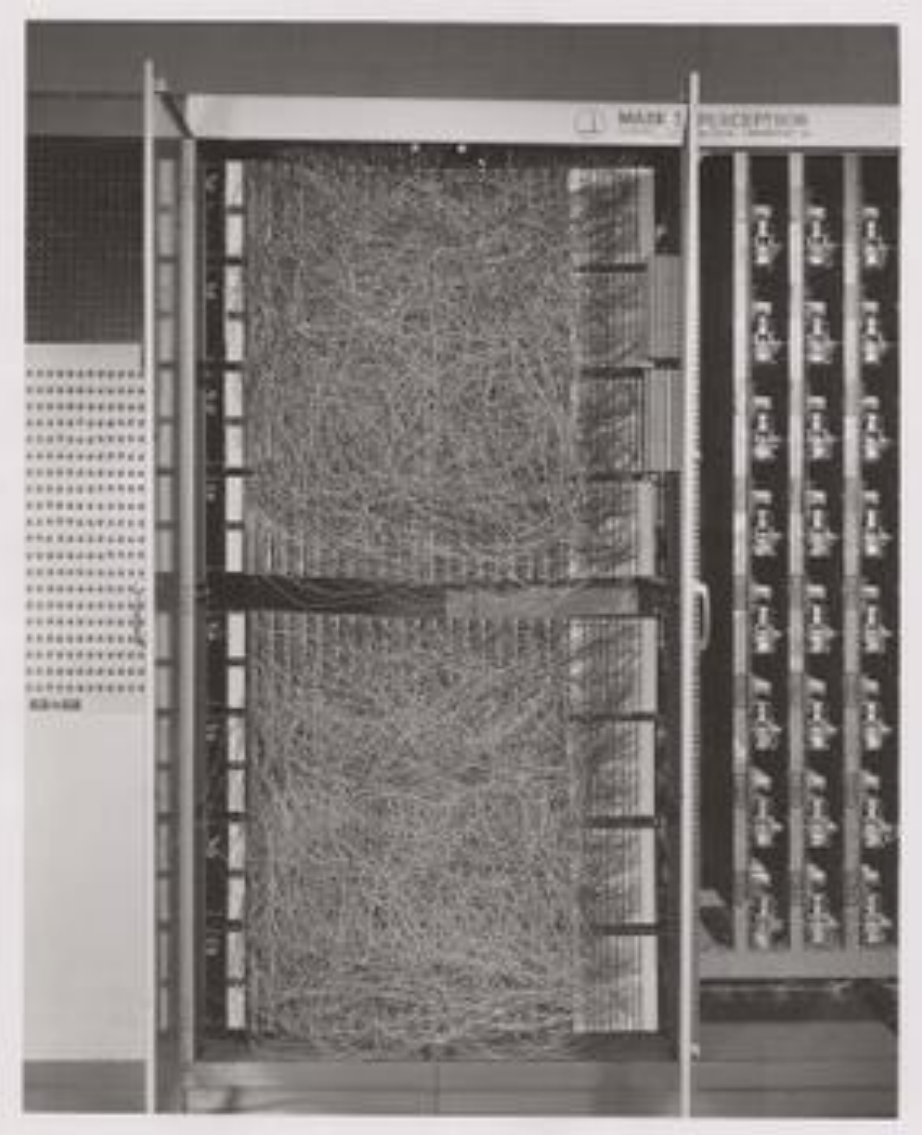


Fig. 1. – *Basic schema of neuronal autocorrelator.* A is the input neuron, B_1, B_2, B_3, ... is a delay chain. The original signal and the delayed signal are multiplied when A and B_k feed C_k, and a running integral of the product is obtained at the synapse between C_k and D_k, where excitation accumulates whenever C_k discharges and dissipates itself at a rate proportional to the amount accumulated. Since these operations correspond to the definition of running autocorrelation, the excitatory states at D_1, D_2, D_3, ... provide a display of the running autocorrelation function of the input time function, the temporal course of the discharges of A.

J.C.R. Licklider, "A Duplex Theory of Pitch Perception," Experientia VII(4):128-134, 1951

Perceptron



- Rosenblatt was granted a patent for the “perceptron,” an electrical circuit model of a neuron.
- The perceptron is basically a network of McCulloch-Pitts neurons.
- Rosenblatt’s key innovation was the perceptron learning algorithm.

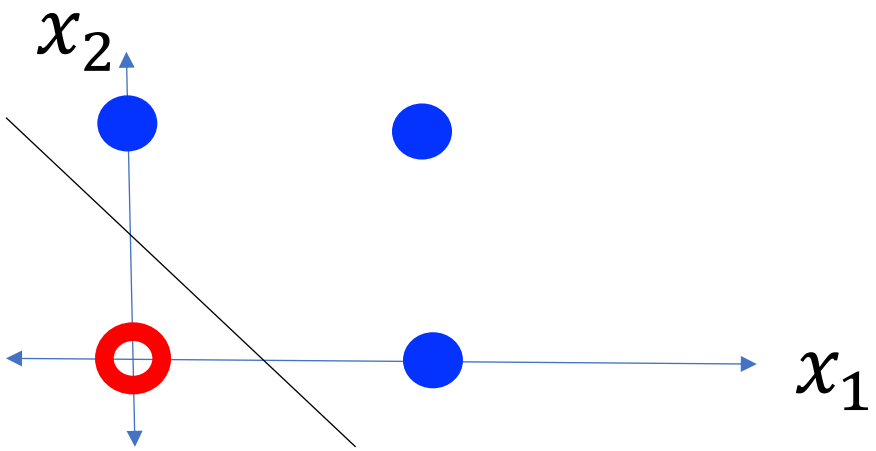
A McCulloch-Pitts Neuron can compute some logical functions…

When the features are binary ($x_j \in \{0,1\}$), many (but not all!) binary functions can be re-written as linear functions. For example, the function

$$Y^* = (x_1 \vee x_2)$$

can be re-written as

$$Y^* = 1 \quad \text{if: } x_1 + x_2 - 0.5 > 0$$

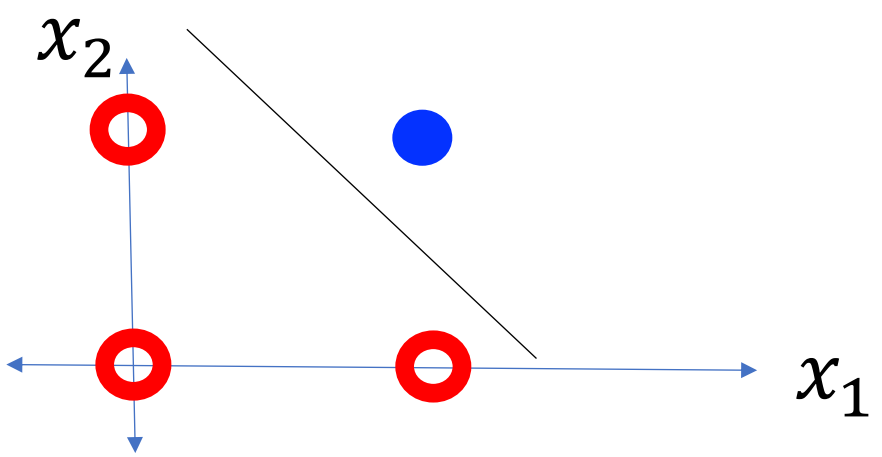


Similarly, the function

$$Y^* = (x_1 \wedge x_2)$$

can be re-written as

$$Y^* = 1 \quad \text{if: } x_1 + x_2 - 1.5 > 0$$

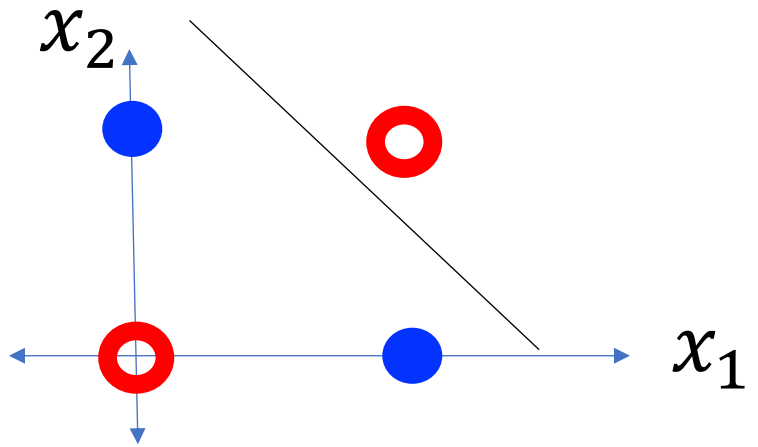


… but not all.

- Not all logical functions can be written as linear classifiers!
- Minsky and Papert wrote a book called *Perceptrons* in 1969. Although the book said many other things, the only thing most people remembered about the book was that:

"A linear classifier cannot learn an XOR function."

- Because of that statement, most people gave up working on neural networks from about 1969 to about 2006.
- Minsky and Papert also proved that a ***two-layer neural net*** can compute an XOR function. But most people didn't notice.



Outline

- Why use more than one layer?
  - Biological inspiration
  - Representational power: the XOR function
- Two-layer neural networks
  - The Fundamental Theorem of Calculus
  - Feature learning for linear classifiers
- Deep networks
  - Biological inspiration: features computed from features
  - Flexibility: convolutional, recurrent, and gated architectures

The Fundamental Theorem of Calculus

The Fundamental Theorem of Calculus (proved by Isaac Newton) says that

$$f(x) = \lim_{\Delta \to 0} \frac{A(x+\Delta) - A(x)}{\Delta}$$

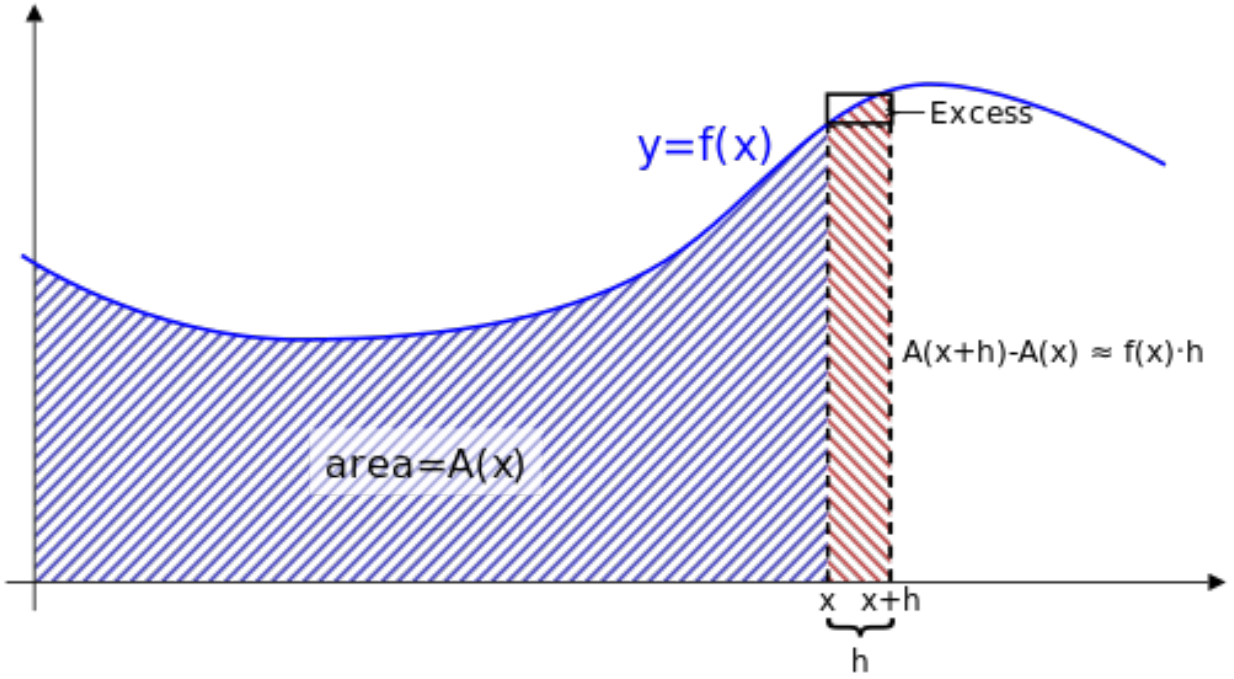


Illustration of the Fundamental Theorem of Calculus: any smooth function is the derivative of its own integral. The integral can be approximated as the sum of rectangles, with error going to zero as the width goes to zero.

The Fundamental Theorem of Calculus

Imagine the following neural network.

Each neuron computes

$$h_k(x) = u(x - k\Delta)$$

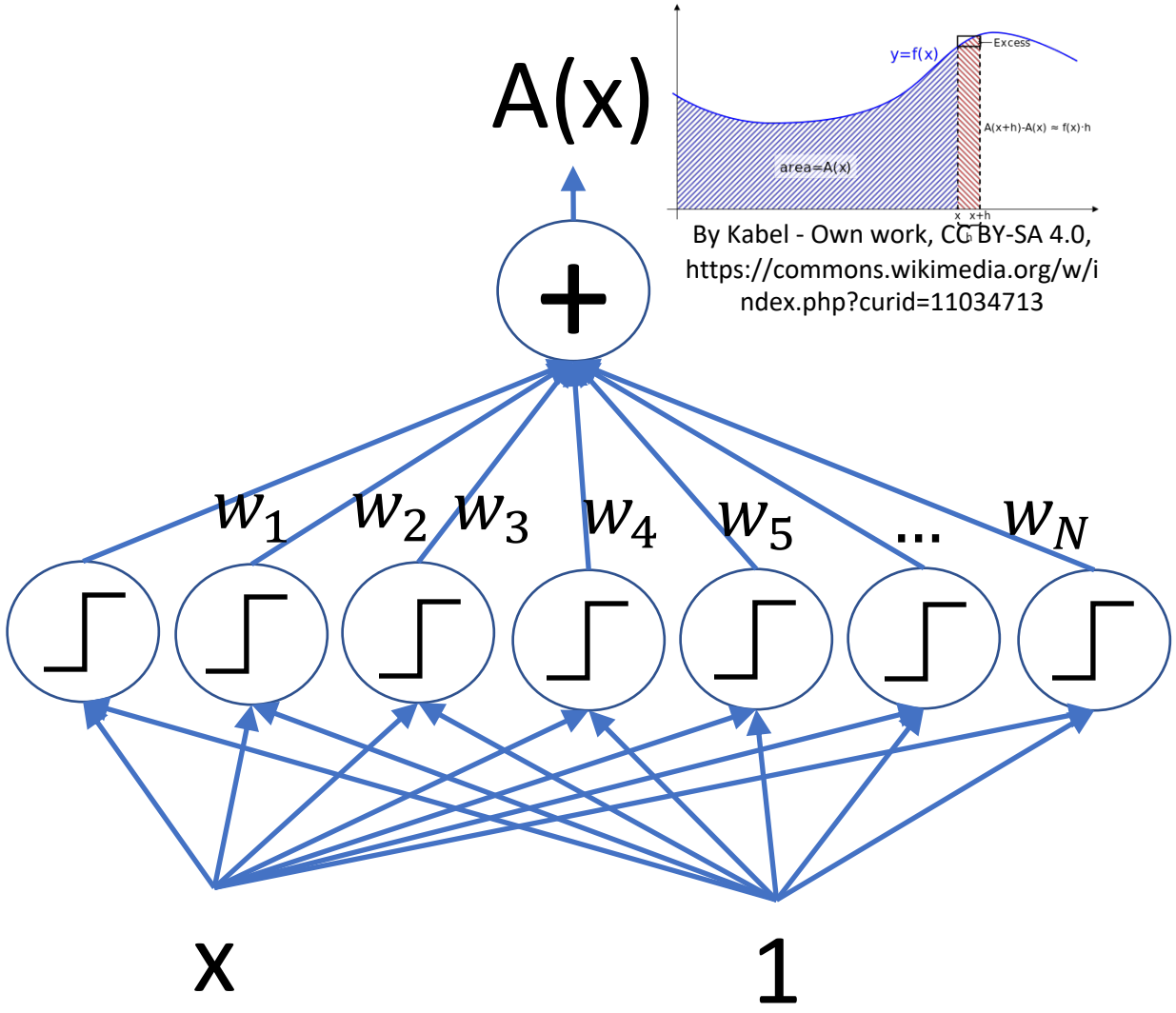
Where u(x) is the unit step function.

Define

$$w_k = A(k\Delta) - A((k-1)\Delta)$$

Then, for any smooth function A(x),

$$A(x) = \lim_{\Delta \to 0} \sum_{k=-\infty}^{\infty} w_k h_k(x)$$



By Kabel - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=11034713

The Fundamental Theorem of Calculus

Imagine the following neural network.

Each neuron computes

$$h_k(x) = u(x - k\Delta)$$

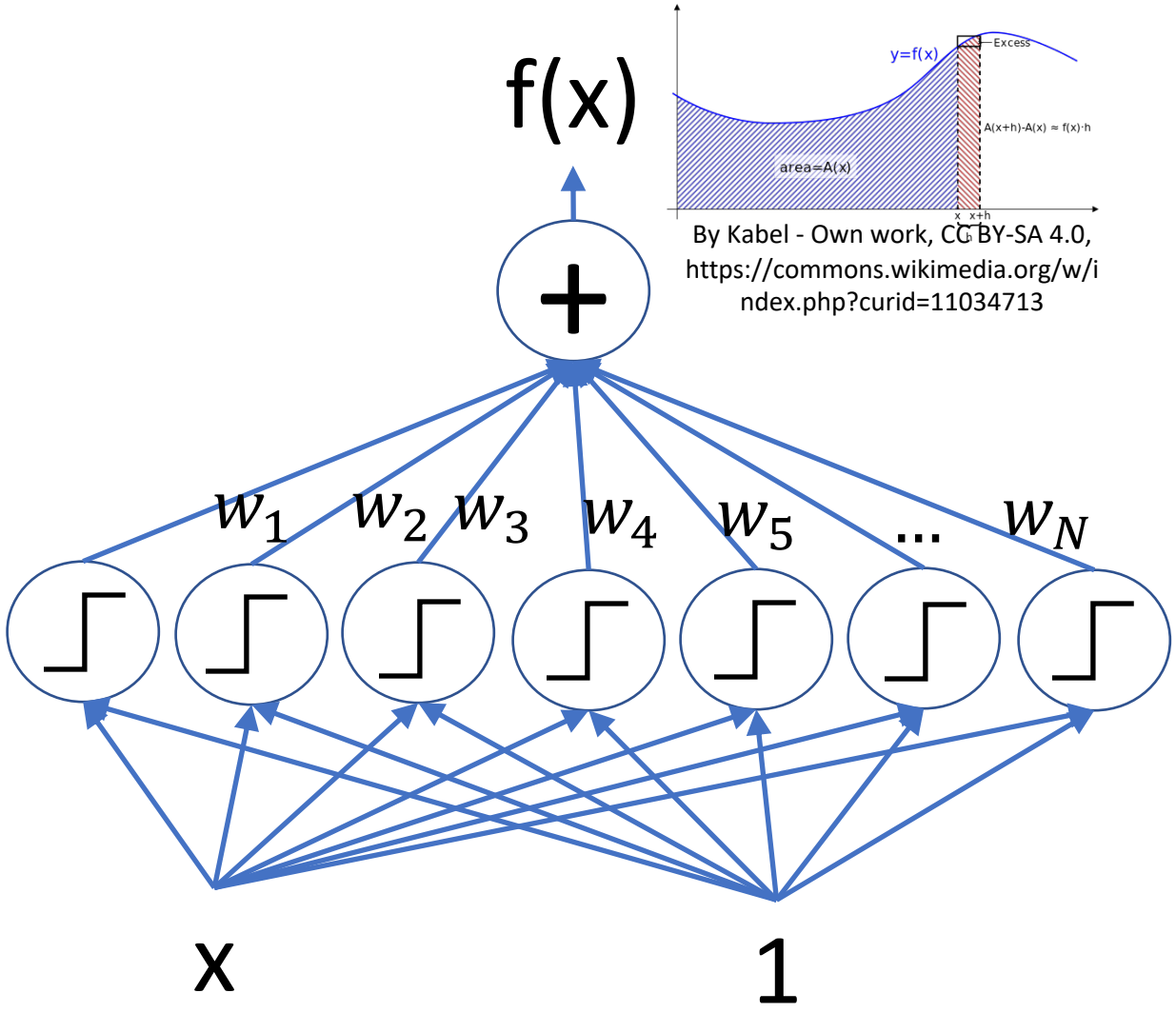
Where u(x) is the unit step function.

Define

$$w_k = f(k\Delta) - f((k-1)\Delta)$$

Then, for any smooth function f(x),

$$f(x) = \lim_{\Delta \to 0} \sum_{k=-\infty}^{\infty} w_k h_k(x)$$



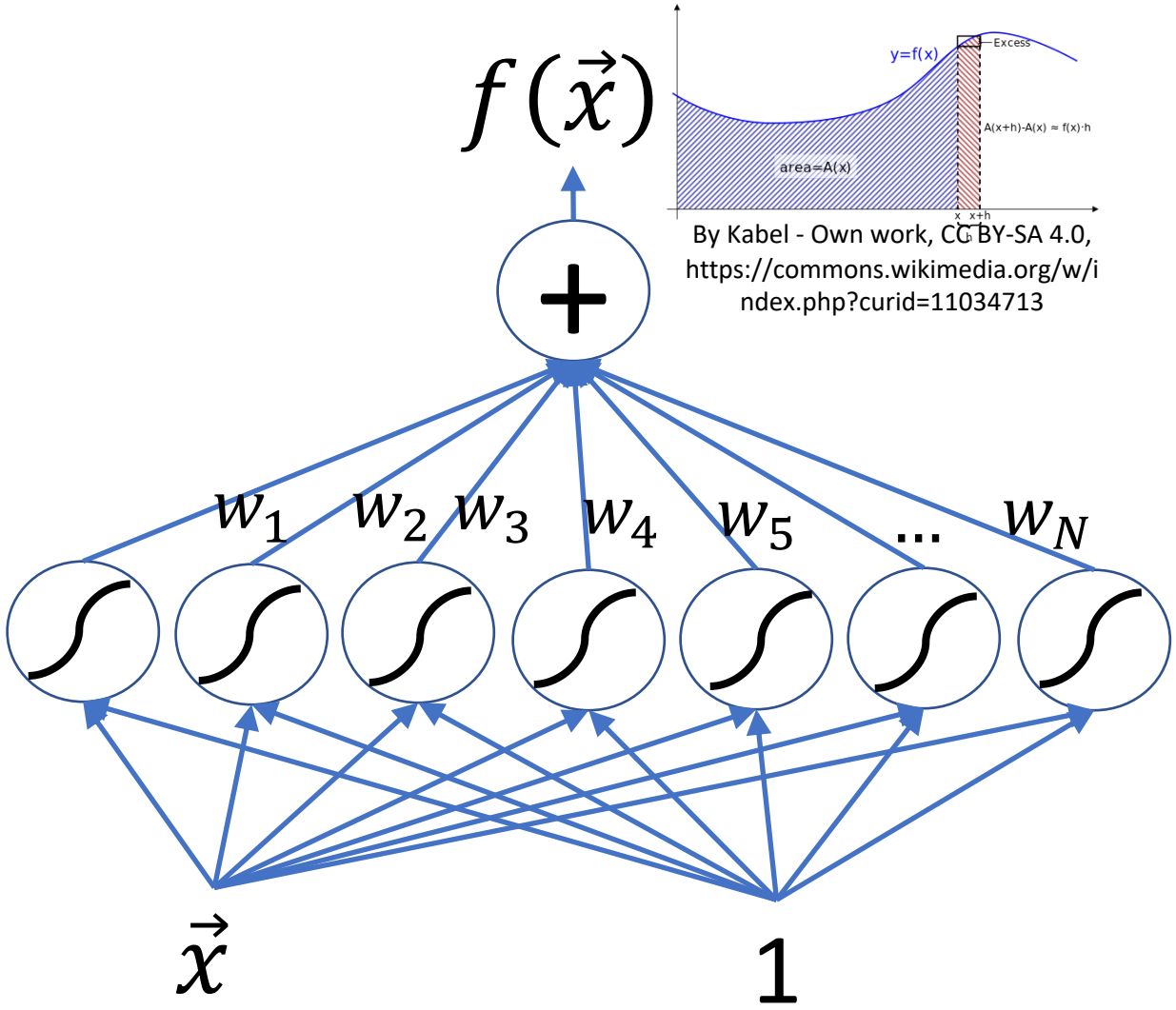
By Kabel - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=11034713

The Neural Network Representer Theorem

(Barron, 1993, "Universal Approximation Bounds for Superpositions of a Sigmoidal Function")

For ***any*** vector function $f(\vec{x})$ that is sufficiently smooth, and whose limit as $\vec{x} \to \infty$ decays sufficiently, there is a two-layer neural network with N sigmoidal hidden nodes $h_k(\vec{x})$ and second-layer weights w_k such that

$$f(\vec{x}) = \lim_{N \to \infty} \sum_{k=1}^{N} w_k h_k(\vec{x})$$



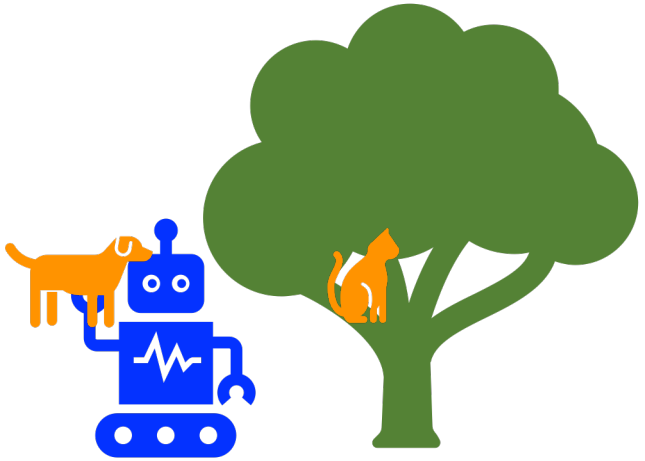
By Kabel - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=11034713

Outline

- Why use more than one layer?
  - Biological inspiration
  - Representational power: the XOR function
- Two-layer neural networks
  - The Fundamental Theorem of Calculus
  - Feature learning for linear classifiers
- Deep networks
  - Biological inspiration: features computed from features
  - Flexibility: convolutional, recurrent, and gated architectures

Classifiers example: dogs versus cats

Can you write a program that can tell which ones are dogs, and which ones are cats?



Idea #3:

$x_1 =$ tameness (# times the animal comes when called, out of 40).

$x_2 =$ weight of the animal, in pounds.

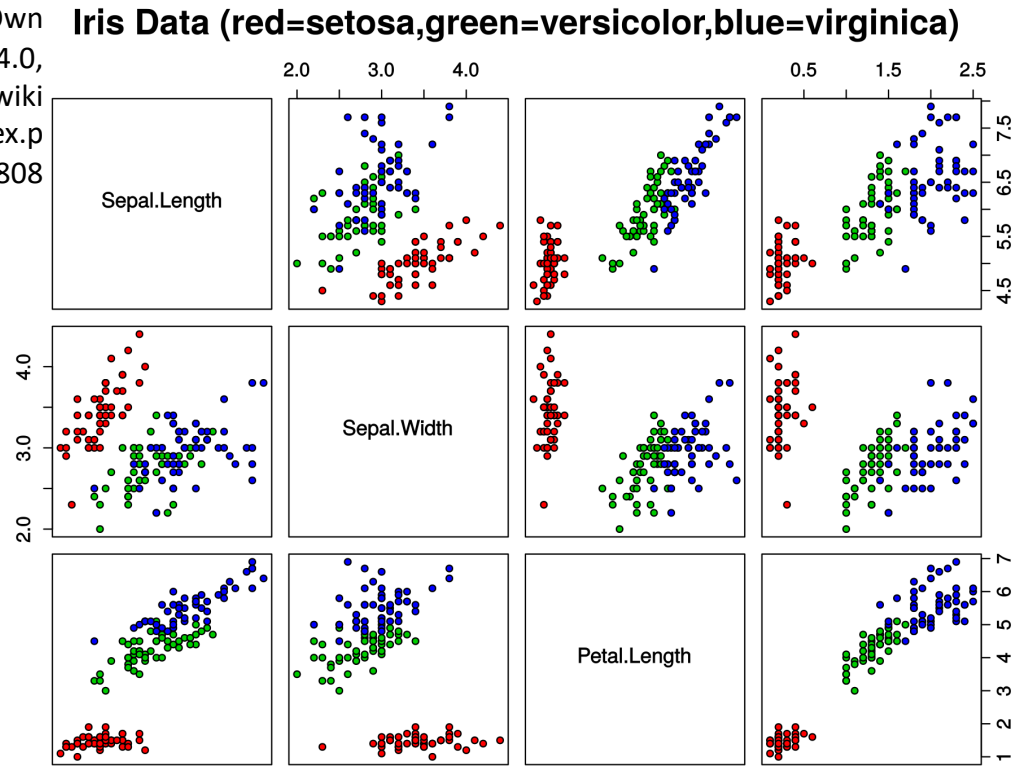
If $0.5x_1 + 0.5x_2 > 20$, call it a dog.

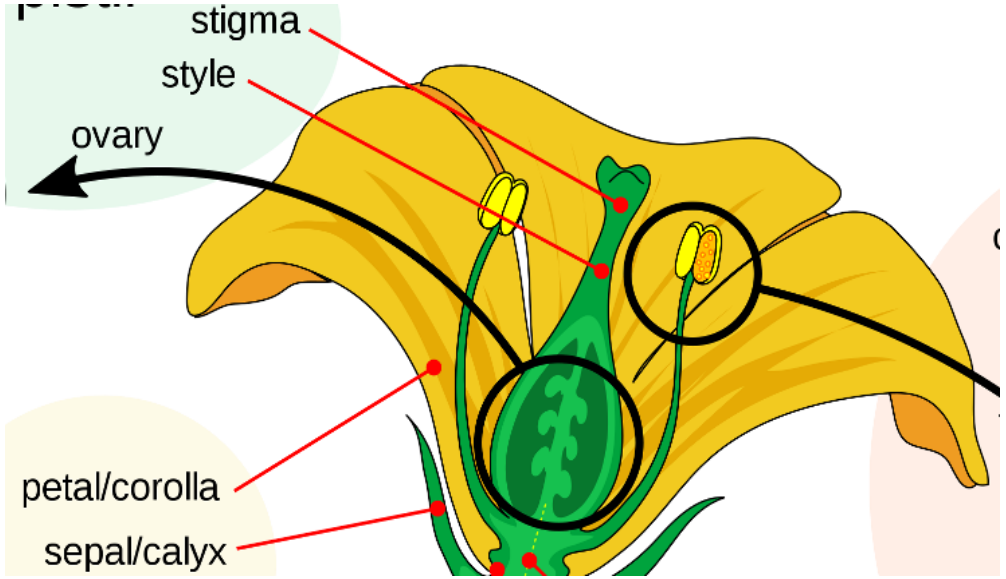
Otherwise, call it a cat.

This is called a "linear classifier" because $0.5x_1 + 0.5x_2 = 20$ is the equation for a line.

The feature selection problem

- The biggest problem people had with linear classifiers, until back-propagation came along, was: Which features should I observe?
  - (TAMENESS? Really? What is that, and how do you measure it?)
- Example: linear discriminant analysis was invented by Ronald Fisher (1936) using 4 measurements of irises:
  - Sepal width & length
  - Petal width & length
- How did he come up with those measurements? Why are they good measurements?





Extracted from Mature_flower_diagram.svg
By Mariana Ruiz LadyofHats - Own work, Public Domain, https://commons.wikimedia.org/w/index.php?curid=2273307

Feature Learning: A way to think about neural nets

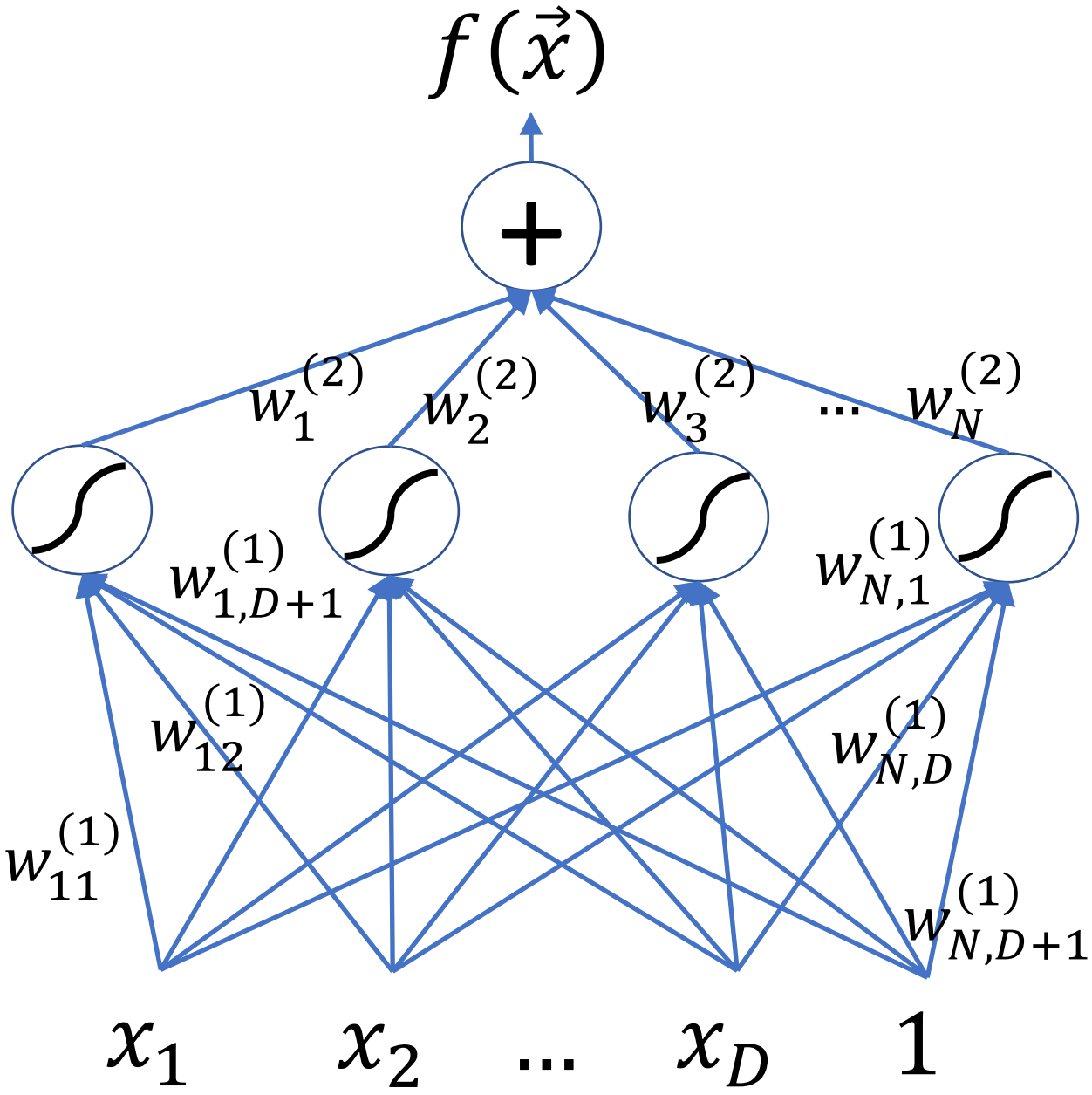
The solution to the "feature selection" problem turns out to be, in many cases, totally easy: if you don't know the features, then learn them!

Define a two-layer neural network. The first-layer weights are $w_{kd}^{(1)}$. The first layer computes

$$h_k(\vec{x}) = \sigma\left(\sum_{d=1}^{D+1} w_{kd}^{(1)} x_d\right)$$

The second-layer weights are $w_k^{(2)}$. It computes

$$f(\vec{x}) = \sum_{k=1}^{N} w_k^{(2)} h_k(\vec{x})$$



Feature Learning: A way to think about neural nets

For example, consider the XOR problem.

Suppose we create two hidden nodes:

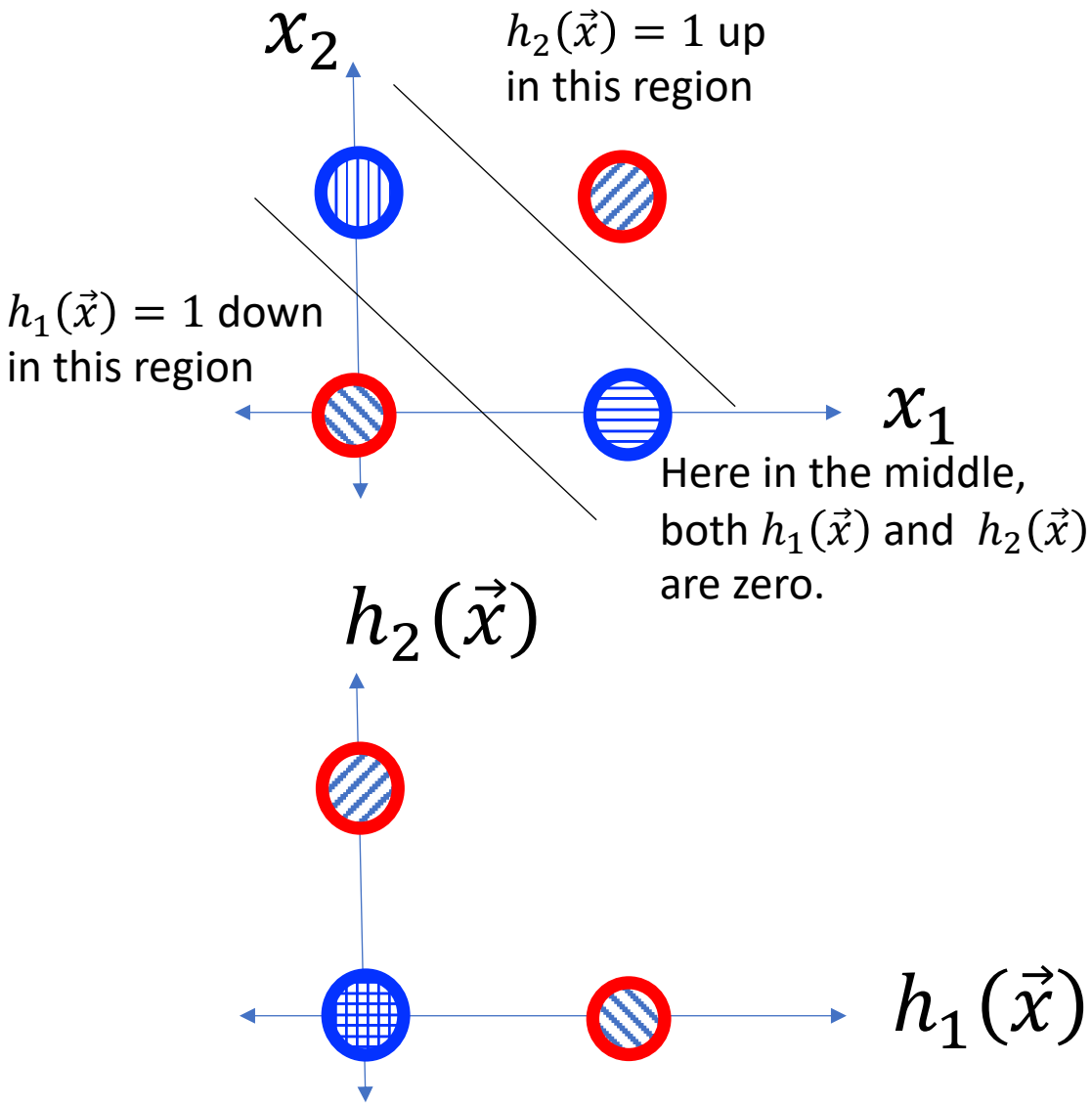
$$h_1(\vec{x}) = u(0.5 - x_1 - x_2)$$

$$h_2(\vec{x}) = u(x_1 + x_2 - 1.5)$$

Then the XOR function $Y^* = (x_1 \oplus x_2)$

is given by

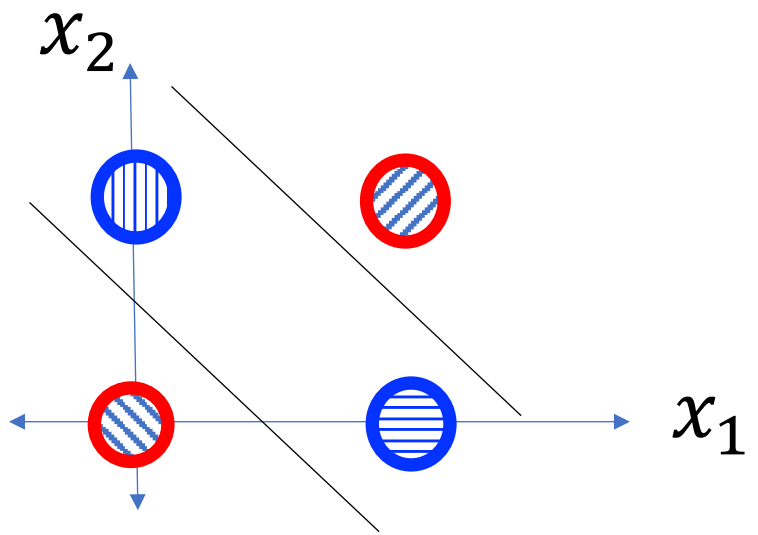
$$Y^* = h_1(\vec{x}) + h_2(\vec{x}) - 1$$

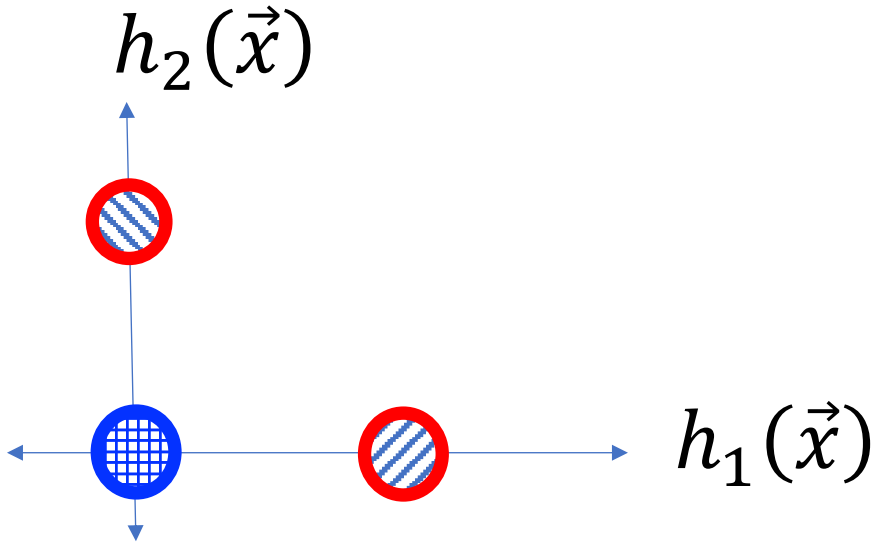


Feature Learning: A way to think about neural nets

In general, this is one of the most useful ways to think about neural nets:

- The first layer learns a set of features.
- The second layer learns a linear classifier, using those features as its input.





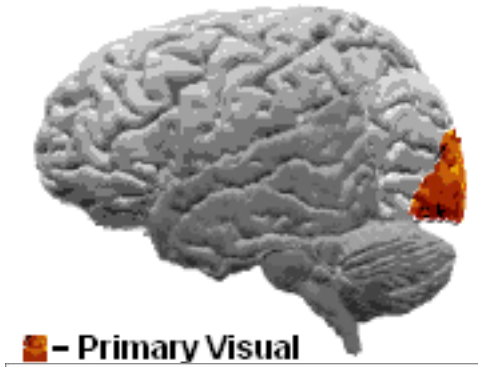
Outline

- Why use more than one layer?
  - Biological inspiration
  - Representational power: the XOR function
- Two-layer neural networks
  - The Fundamental Theorem of Calculus
  - Feature learning for linear classifiers
- Deep networks
  - Biological inspiration: features computed from features
  - Flexibility: convolutional, recurrent, and gated architectures

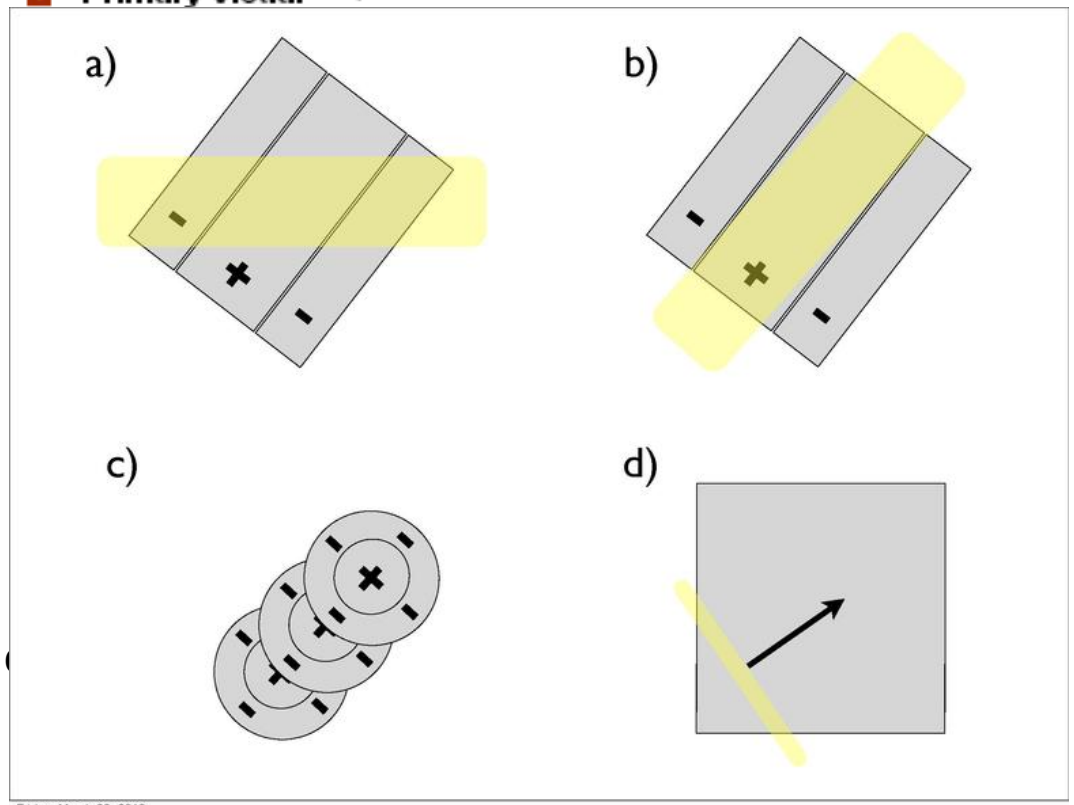
Biological Inspiration: Simple, Complex, and Hypercomplex Cells in the Visual Cortex

D. Hubel and T. Wiesel (1959, 1962, Nobel Prize 1981) found that the human visual cortex consists of a hierarchy of *simple*, *complex*, and *hypercomplex* cells.

- Simple cells (in visual area 1, called V1) fire when you see a simple pattern of colors in a particular orientation (figure (b), at right)



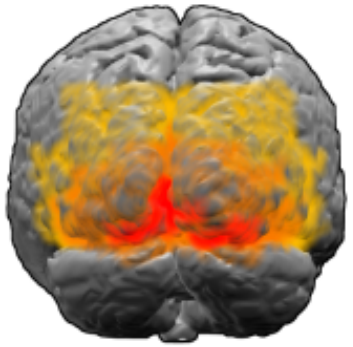
By Chavez01 at English Wikipedia - Transferred from en.wikipedia to Commons by ×⍰×ª× ×⍰×⍰ using CommonsHelper., Public Domain, https://commons.wikimedia.org/w/index.php?curid=4431766



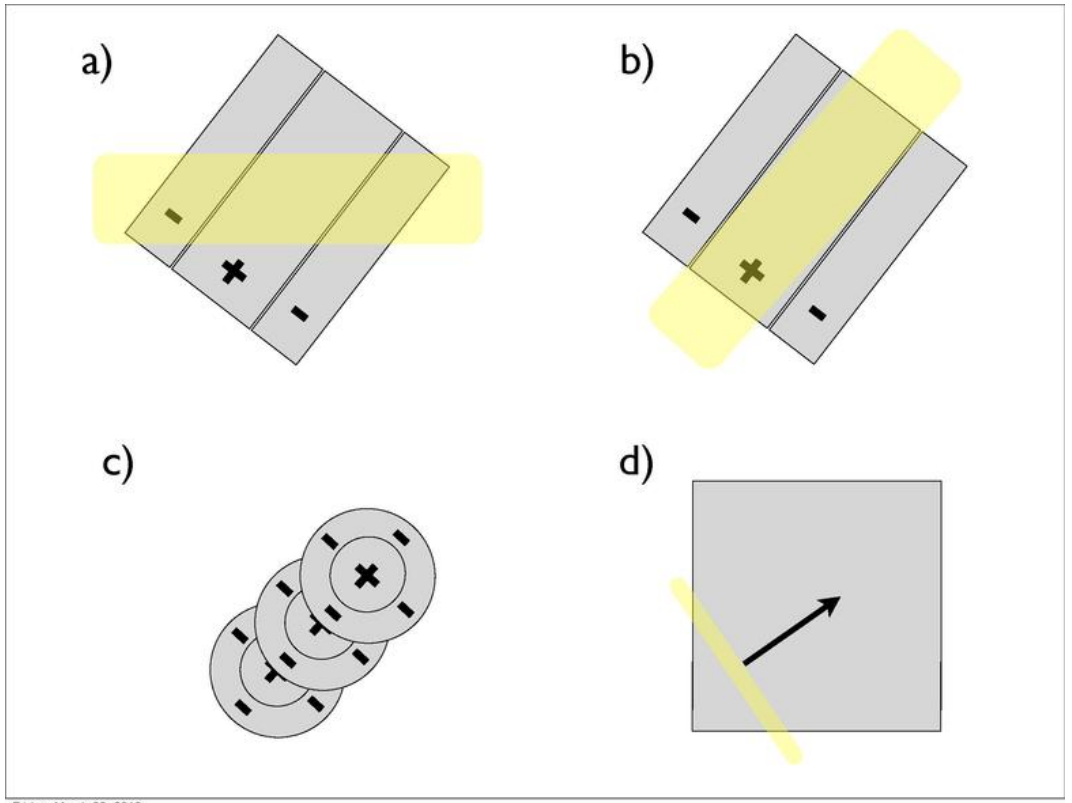
Biological Inspiration: Simple, Complex, and Hypercomplex Cells in the Visual Cortex

D. Hubel and T. Wiesel (1959, 1962, Nobel Prize 1981) found that the human visual cortex consists of a hierarchy of *simple*, *complex*, and *hypercomplex* cells.

- Complex cells are sensitive to moving stimuli of a particular orientation traveling in a particular direction (figure (d) at right).
- Complex cells can be modeled as linear combinations of simple cells!



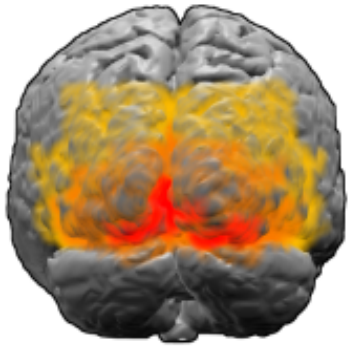
View of the brain from behind. Brodman area 17=Red; 18=orange; 19=yellow. By Washington irving at English Wikipedia, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=1643737



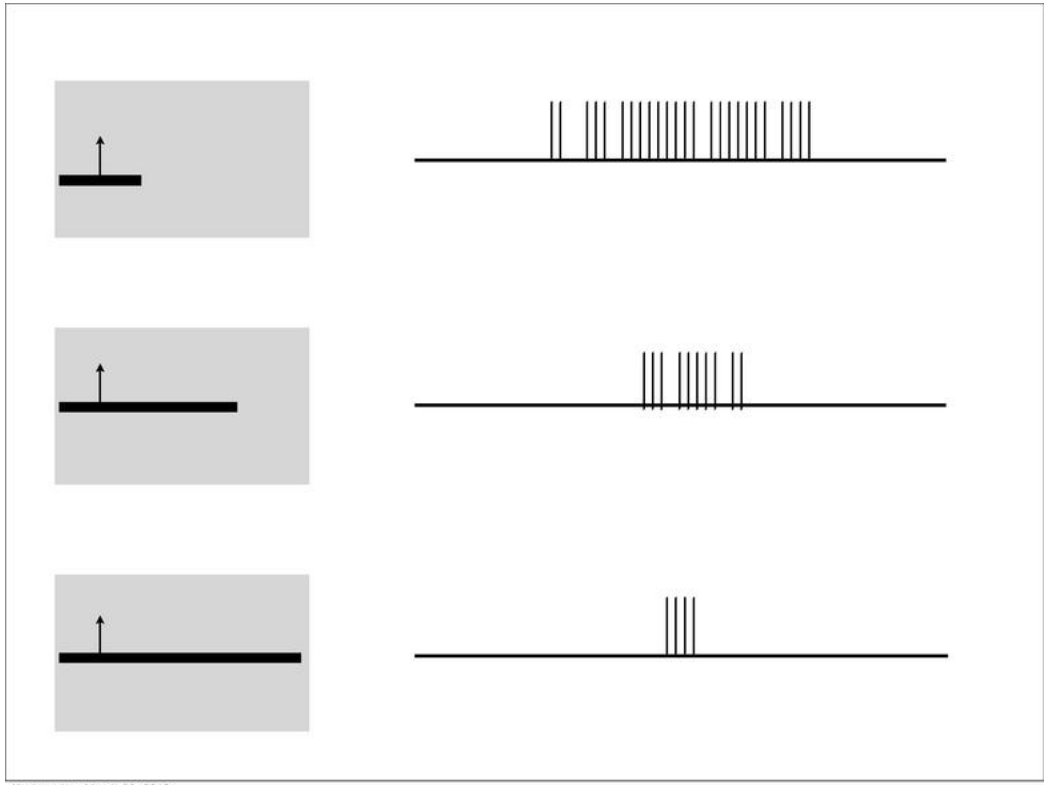
Biological Inspiration: Simple, Complex, and Hypercomplex Cells in the Visual Cortex

D. Hubel and T. Wiesel (1959, 1962, Nobel Prize 1981) found that the human visual cortex consists of a hierarchy of *simple*, *complex*, and *hypercomplex* cells.

- Hypercomplex cells are sensitive to moving stimuli of a particular orientation traveling in a particular direction, and they also stop firing if the stimulus gets too long.
- Hypercomplex cells can be modeled as linear combinations of complex cells!

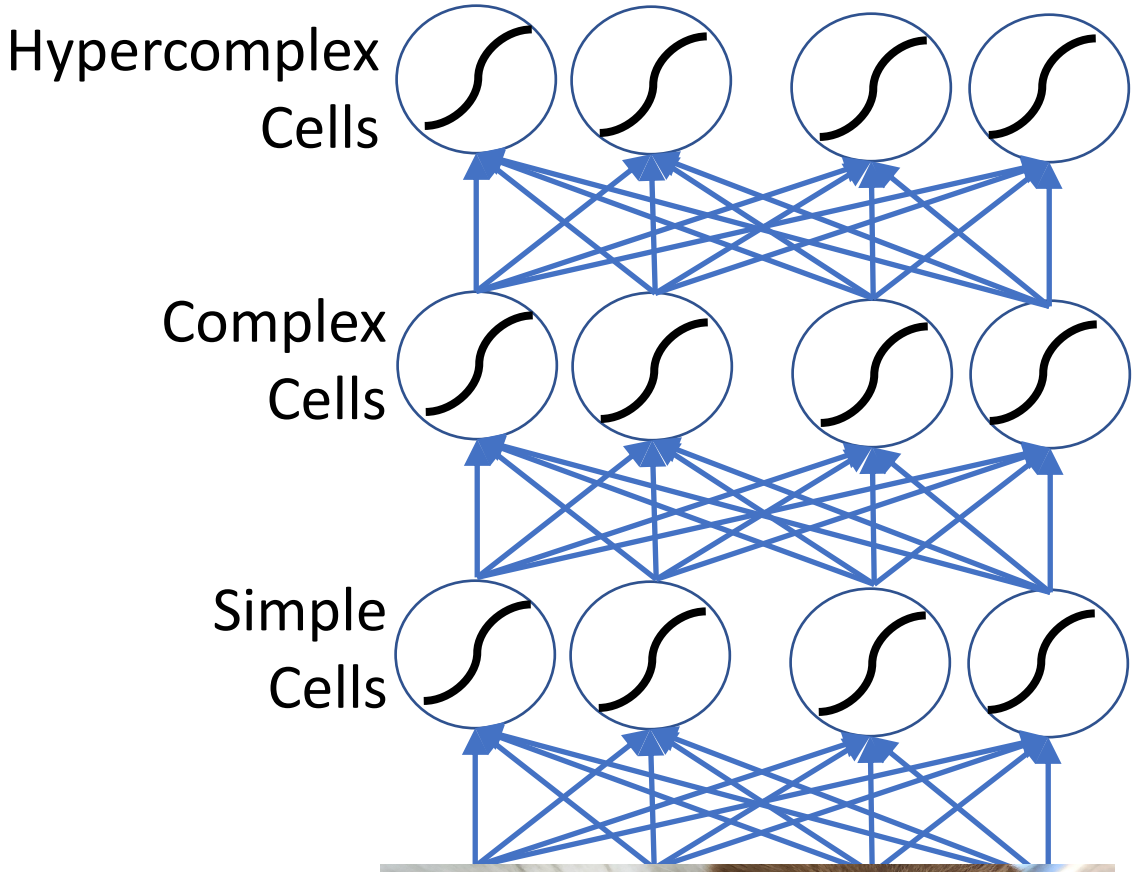


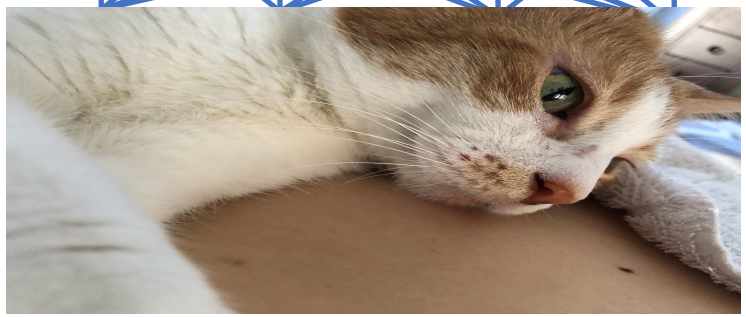
View of the brain from behind. Brodman area 17=Red; 18=orange; 19=yellow. By Washington irving at English Wikipedia, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=1643737



Biological Inspiration: Simple, Complex, and Hypercomplex Cells in the Visual Cortex

Hubel & Wiesel's simple, complex, and hypercomplex cells have been modeled as a hierarchy, in which each type of cell computes a linear combination of the type below it, followed by a nonlinear activation function.





Outline

- Why use more than one layer?
  - Biological inspiration
  - Representational power: the XOR function
- Two-layer neural networks
  - The Fundamental Theorem of Calculus
  - Feature learning for linear classifiers
- Deep networks
  - Biological inspiration: features computed from features
  - Flexibility: convolutional, recurrent, and gated architectures

Flexibility: many types of deep networks

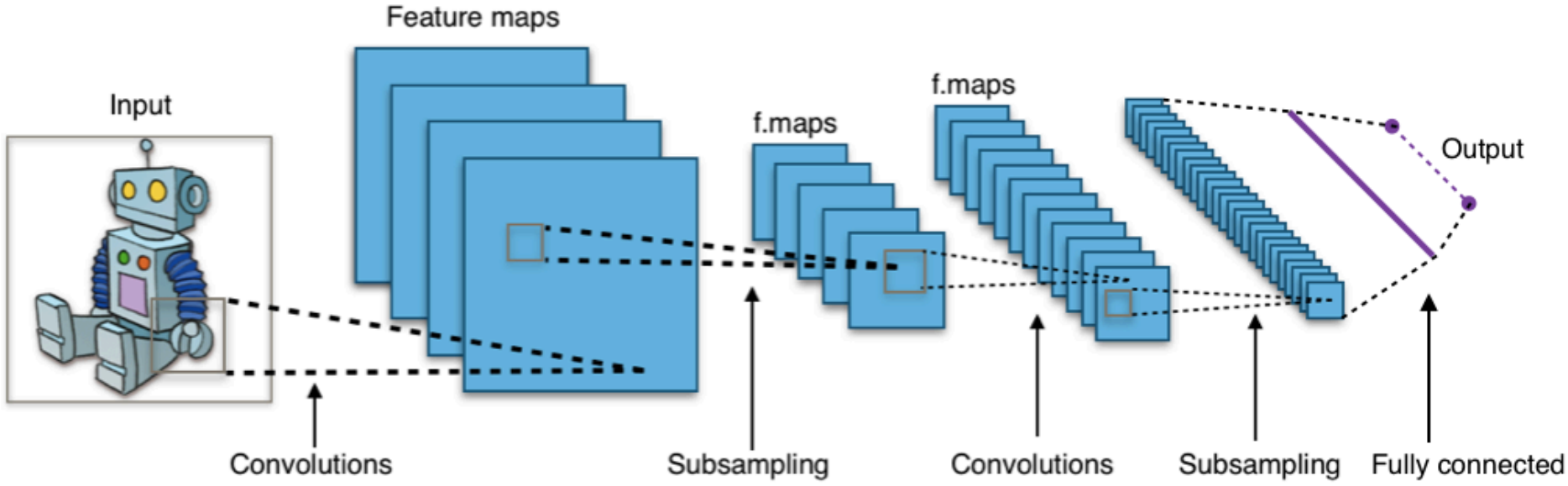
The other reason to use deep neural networks is that, with a deep enough network, many types of learning algorithms are possible, far beyond simple classifiers.

- Convolutional neural networks: output depends on the shape of the input, regardless of where it occurs in the image.
- Recurrent neural network: output depends on past values of the output.
- Gated neural network: one set of cells is capable of turning another set of cells on or off.

Convolutional Neural Network

In a convolutional neural network, the multiplicative first layer is replaced by a convolutional first layer:

$$h_k(\vec{x}) = \sigma\left(\sum_{d=-D}^{D} w_d x_{k-d}\right)$$

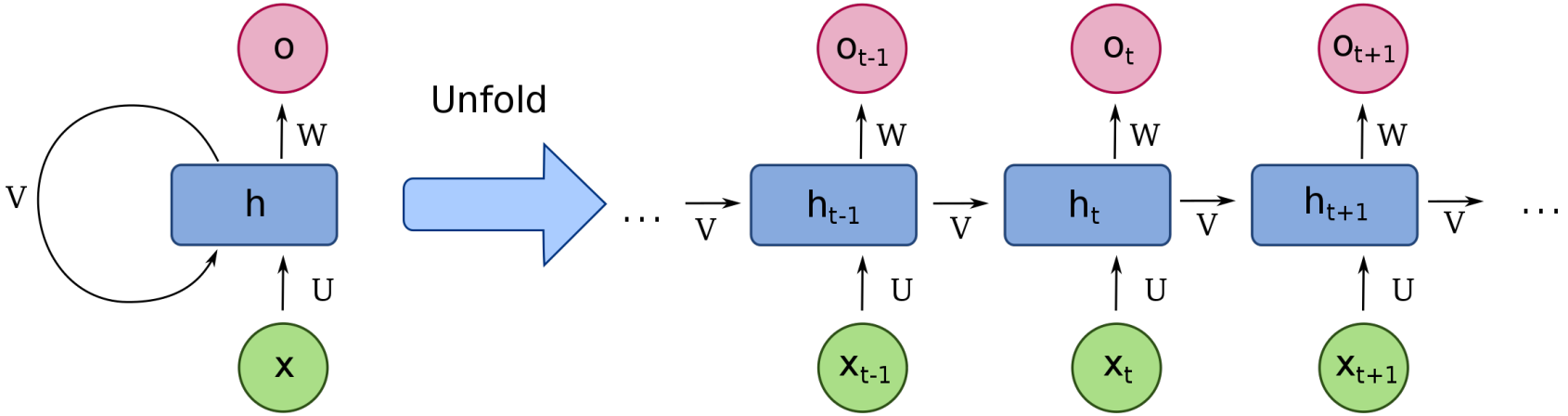


By Aphex34 - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=45679374

Recurrent Neural Network

In a recurrent neural network, the hidden nodes at time t depend on the hidden nodes at time t-1:

$$h_{kt} = \sigma\left(\sum_{d=1}^{D} u_{kd} x_{dt} + \sum_{j=1}^{N} v_{kj} h_{j,t-1}\right)$$



Gated Neural Network

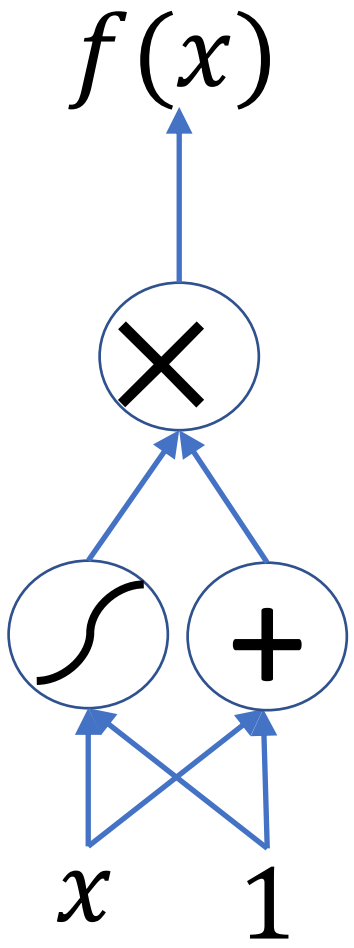
In a gated neural network (like the "long-short-term memory" network shown here), the output of some hidden nodes are called "gates:"

$$g = \sigma(u_1 x + b_1) \in [0,1]$$

The gates are then multiplied by the outputs of other hidden nodes, effectively turning them on or off:

$$c = u_2 x + b_2$$

$$f(x) = g \times c$$



What are these architectures for?

- Convolutional neural networks: output depends on the shape of the input, regardless of where it occurs in the image.
- Recurrent neural network: output depends on past values of the output.
- Gated neural network: one set of cells is capable of turning another set of cells on or off.

Conclusions

- Why use more than one layer?
  - Biological inspiration: the simplest neuronal network in the human body, the reflex arc, still uses at least two neurons
  - Representational power: the XOR function can't be computed with a one-layer network (a perceptron), but it can be computed with two layers
- Two-layer neural networks
  - The Fundamental Theorem of Calculus means that a two-layer network can approximate any function f(x) arbitrarily well, as the number of hidden nodes goes to infinity
  - A useful way to think about neural nets: the last layer is a linear classifier; all of the other layers compute features for the last layer to use
- Deep networks
  - Biological inspiration: human vision (and hearing) compute complex and hypercomplex features from simpler features
  - Flexibility: convolutional=independent of where it occurs in space, recurrent=has internal memory, gated=one hidden node can turn another node on or off