Introduction to Neural Networks

Machine Learning Decal

Hosted by Machine Learning at Berkeley

Agenda

Demos!

Intuition

The Model

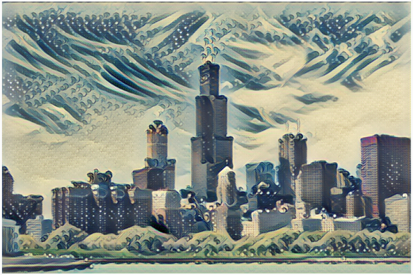
Teaching Neural Networks

Questions

More Demos!

Demos!

Style Transfer



- **Neural Networks** used to be state-of-the-art
  - Now, they form the basis for the state-of-the-art
- They are one of the most flexible machine learning models
  - Style Transfer
  - Optical Character Recognition
  - Dimensionality Reduction
  - Stock Prediction

More Demos

- Helicopter demo: https://see.stanford.edu/Course/CS229/47
    - 1:03:40
- Google Deepmind: Deep Q-Networks
    - https://www.youtube.com/watch?v=V1eYniJ0Rnk
    - 1:10
- Music Generation with RNN/CNNs
    - http://www.hexahedria.com/2015/08/03/composing-music-with-recurrent-neural-networks/

Intuition

What does a neural network do?

- Neural networks are function approximators
  - So what?
  - Everything we are interested in is a function!
- What is a function?
  - Anything that maps an input to a single output

$$f : X \to Y$$

- Take for example a self driving car

$$f : X \to Y$$

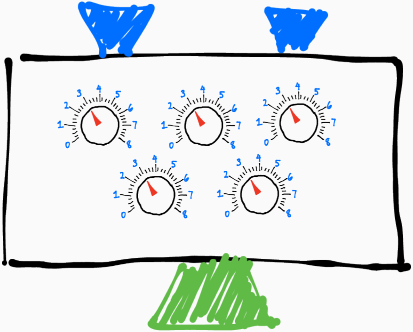
  - f is a function that maps sensor readings to a driver's action
  - X is the set of all possible combinations of sensor readings
  - Y is the set of all possible outputs to a car


Google
self-driving car

Plato

- Somewhere out there, there's a perfect function that tells you exactly what to do for some sensor input (**Platonist view**)
  - We want to approximate that function using a neural network
  - Using training data we've obtained from somewhere

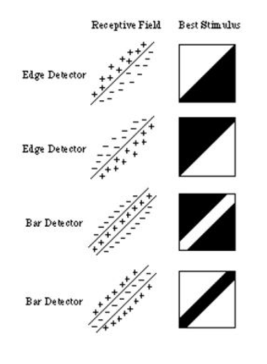
Neural Networks as a Black Box



- Neural networks approximate functions by adjusting **parameters**
  - Modern networks often times have hundreds of millions of parameters
  - We train neural networks to find parameters that approximate our function as closely as possible

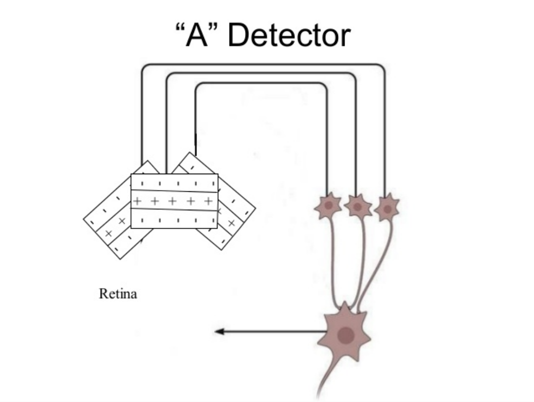
Biological Inspiration for Neural Networks

- Hubel and Weisel
    - https://www.youtube.com/watch?v=IOHayh06LJ4
- Biological neurons in the visual cortex are edge detectors



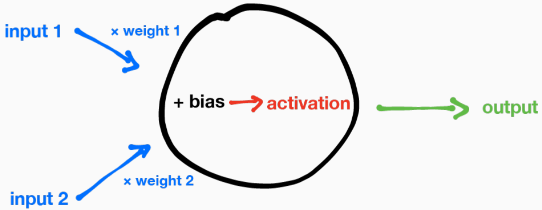
Biological Inspiration for Neural Networks

- By combining the output
  of edge detecting neurons, we can make more complex detectors



The Model

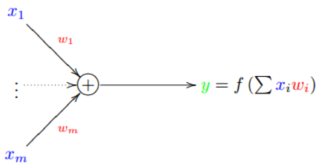
The Artificial Neuron



- Inputs to the neuron are multiplied by weights (the parameters) and then summed
- A bias term (another parameter) is added to the sum
- An activation is then applied (for instance, tanh(x) or ReLU(x))

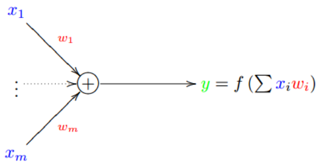
The Artificial Neuron

McCulloch and Pitts (1943) proposed the 'integrate and fire' model:



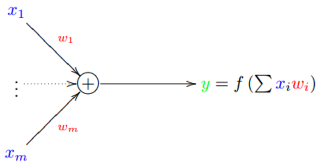
Let $f(x) = x^2$, x = (1, 4, -2), w = (-2, 3, 2). What is the output?

McCulloch and Pitts (1943) proposed the 'integrate and fire' model:



The Artificial Neuron

McCulloch and Pitts (1943) proposed the 'integrate and fire' model:



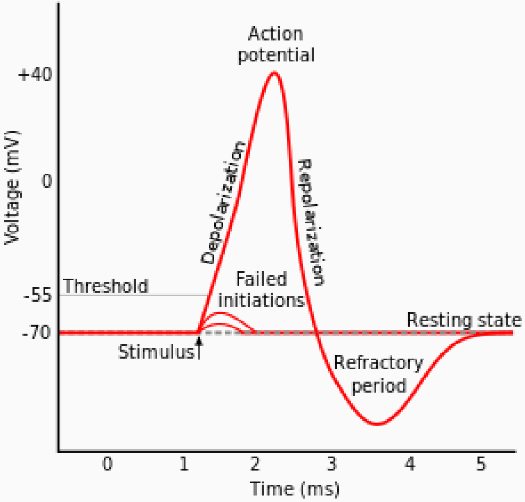
$$\sum_i x_i w_i = (1)(-2) + (4)(3) + (-2)(2) = 6$$

$$f\left(\sum_i x_i w_i\right) = f(6) = 36$$

Activation Function Digression

- What happens if there are no activation functions?
- We need (non-linear) activation functions!
- They give neural networks expressive power
- Activation functions are often called ”non-linearities”

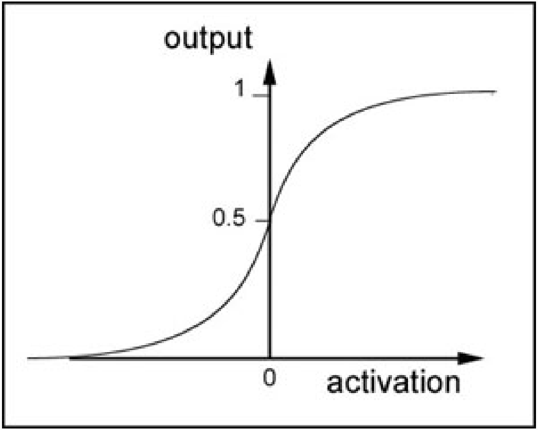
Biological Inspiration



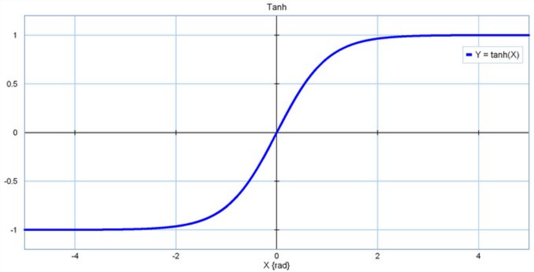
Requirements for activation functions

- Continuous
- Monotonically increasing
- **Differentiable**

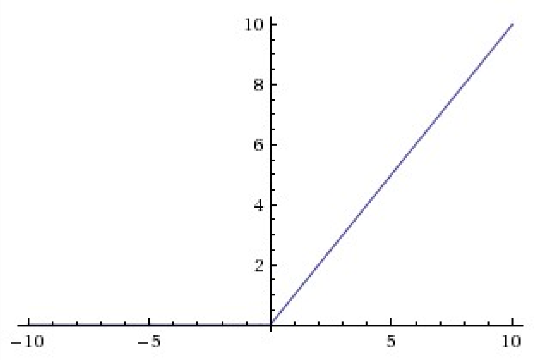
Sigmoid/Logistic

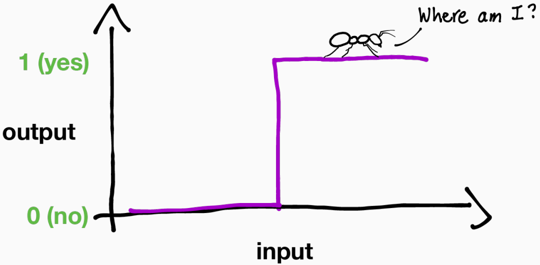


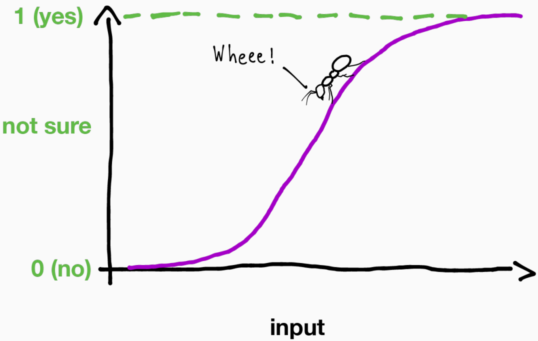
Hyperbolic Tangent (tanh)



Rectified Linear Unit (ReLU): $f(x) = \max(0, x)$

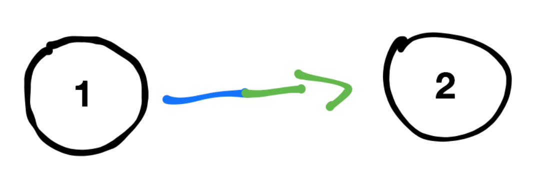



Where am I?
1 (yes)
output
0 (no)
input



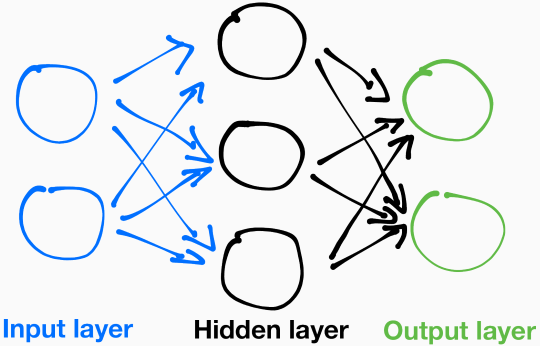
- We want the gradient (derivative) of the activation function to be continuous

Linking it Together



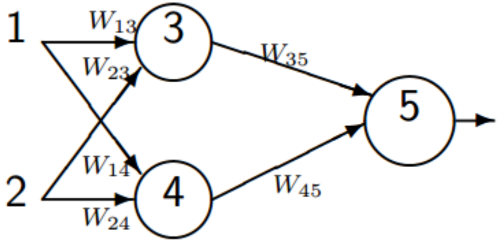
- The output of a neuron becomes the input for another neuron

Neural Network Architecture



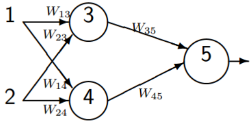
- Layered Architecture
- Three types of layers:
  - **Input Layer** - Data is passed into these neurons
  - **Hidden Layer** - These neurons are "hidden from view"
  - **Output Layer** - These neurons output the result of the network

Feedforward Example



$w_{13} = 2$ $w_{23} = -3$	$w_{35} = 2$
$w_{14} = 1$ $w_{24} = 4$	$w_{45} = -1$

$$f(v) = \begin{cases} 1 & \text{if } v \geq 0 \\ 0 & \text{otherwise} \end{cases}$$



$w_{13} = 2$ $w_{23} = -3$	$w_{35} = 2$ $w_{45} = -1$
$w_{14} = 1$ $w_{24} = 4$	

$$f(v) = \begin{cases} 1 & \text{if } v \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

$$w_{13}(1) + w_{23}(2) = (2)(1) + (-3)(2) = -4$$

$$w_{14}(1) + w_{24}(2) = (1)(1) + (4)(2) = 9$$

$$z_3 = 0$$

$$z_4 = 1$$

$$z_5 = f(w_{35}(0) + w_{45}(1)) = f((-1)(1)) = 0$$

Teaching Neural Networks

Teaching Neural Networks

- So this architecture can (theoretically) approximate any function.
- But how do we actually find the correct parameters?
- Gradient Descent!

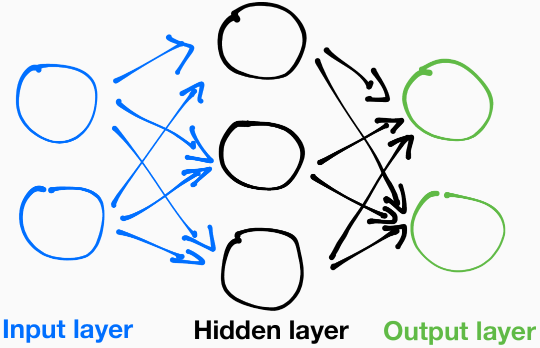
Gradient Descent

- We can define a cost function

$$C(x, \text{parameters}) = \frac{1}{2}(y - \hat{f}(x))^2$$

  - x is our input training example
  - y is our training example label
  - $\hat{f}(x)$ is the output of our network (a function of the parameters and x)
- Gradient descent allows us to find a local minimum of C given the derivatives of C with respect to the parameters

Backpropagation

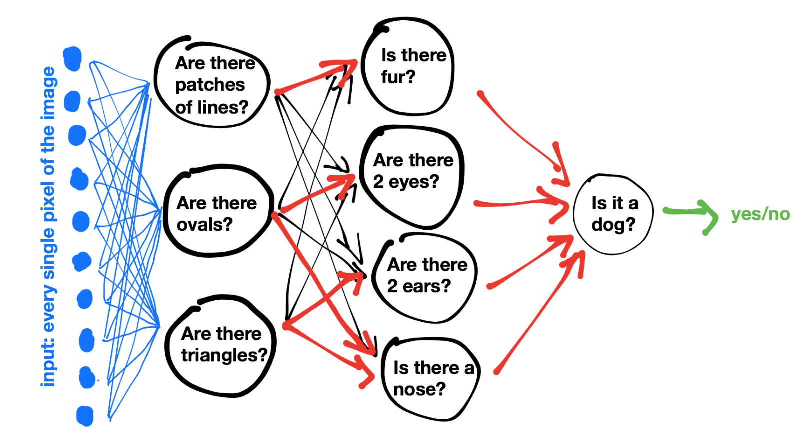


Questions

Questions?

More Demos!

ConvNet Visualization


input: every single pixel of the image
Are there patches of lines?
Are there ovals?
Are there triangles?
Is there fur?
Are there 2 eyes?
Are there 2 ears?
Is there a nose?
Is it a dog?
yes/no

Adversarial Examples