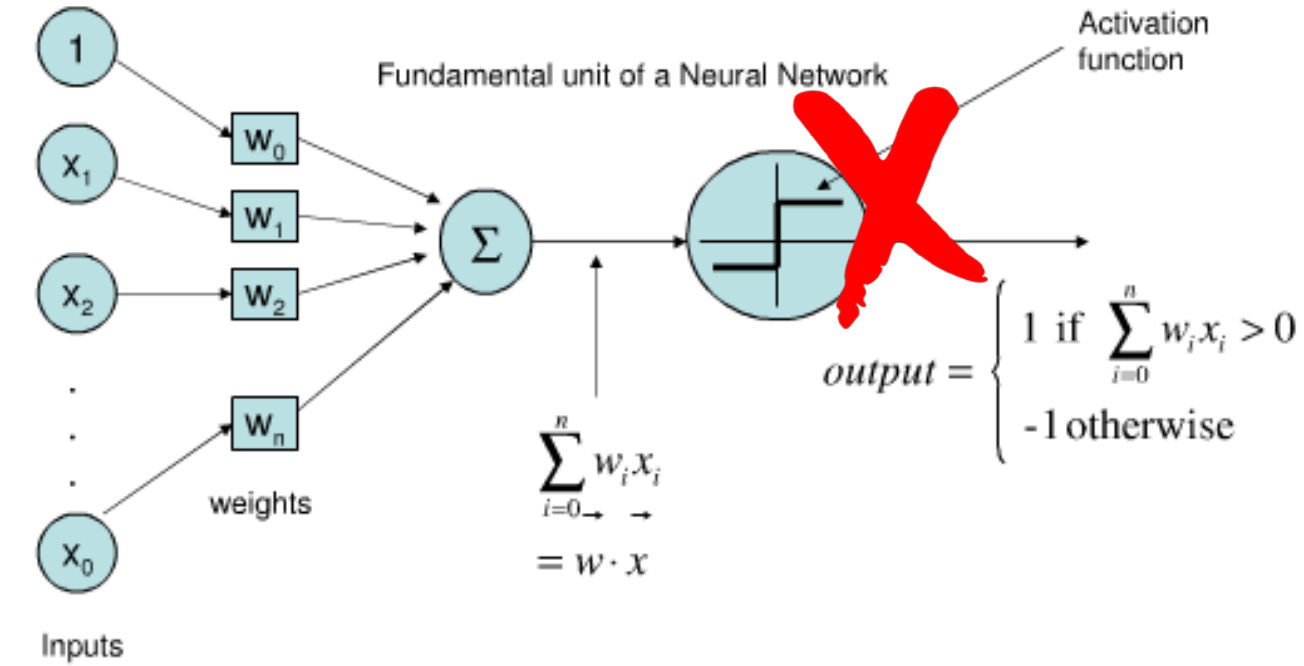
COMP 4752
Computational Intelligence

Lecture 20

More Neural Networks

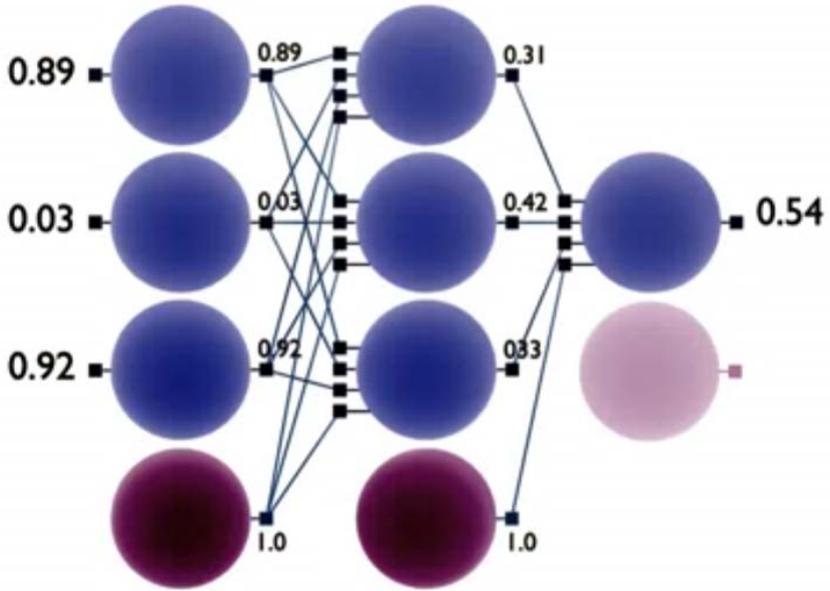

Fundamental unit of a Neural Network
Activation function
1
x_1
x_2
x_0
Inputs
w_0
w_1
w_2
w_n
weights
Σ
$\sum_{i=0}^{n} w_i x_i = \vec{w} \cdot \vec{x}$
$output = \begin{cases} 1 \text{ if } \sum_{i=0}^{n} w_i x_i > 0 \\ -1 \text{ otherwise} \end{cases}$

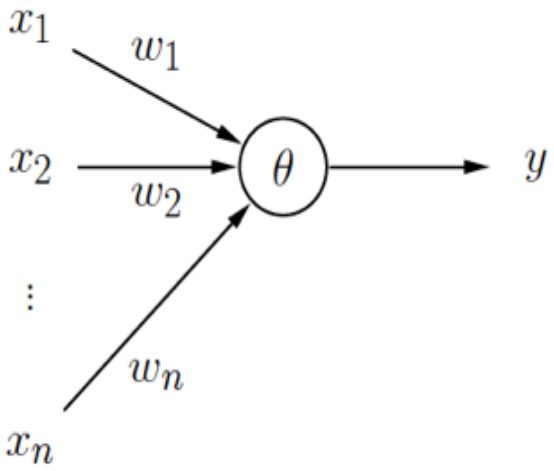
Step 1: Removing Threshold

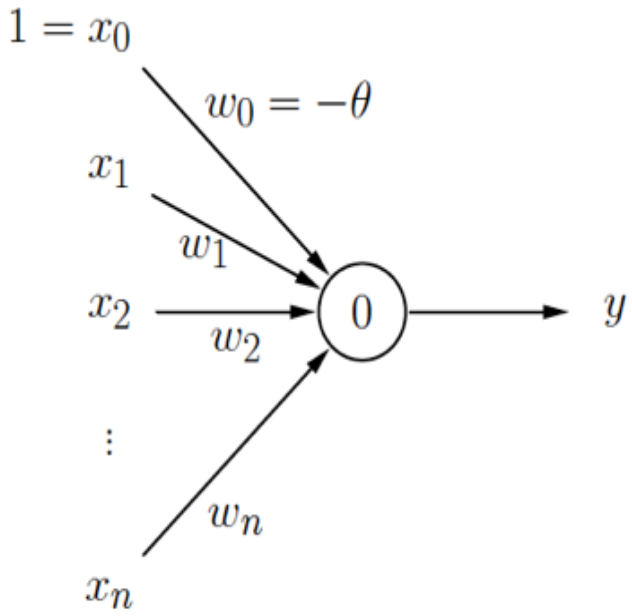
- Thresholding makes our function non-differentiable, also annoying to compute
- Ideally, we want $\overline{y} = f(\overline{x}, \overline{w})$
- Took 25 years to figure out a good way to accomplish this
- Enter the Bias Neuron

Bias Neuron

- An extra neuron added to each layer
- Has fixed output value of 1.0
- Has effect similar to that of thresholding with easier compute





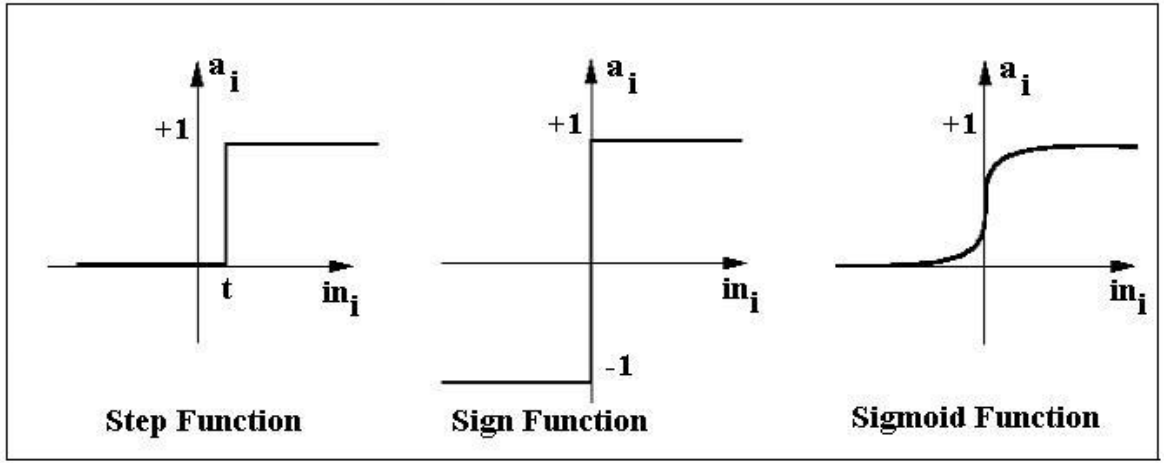


$$\sum_{i=1}^{n} w_i x_i \geq \theta$$

$$\sum_{i=1}^{n} w_i x_i - \theta \geq 0$$

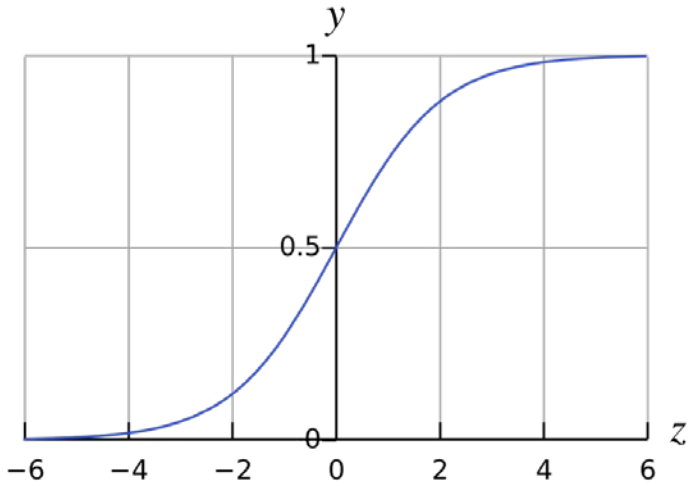
Step 2: Smoothing the Activation

- Step function non-differentiable
- Apply a sigmoid function to smooth it out



Step 2: Smoothing the Activation

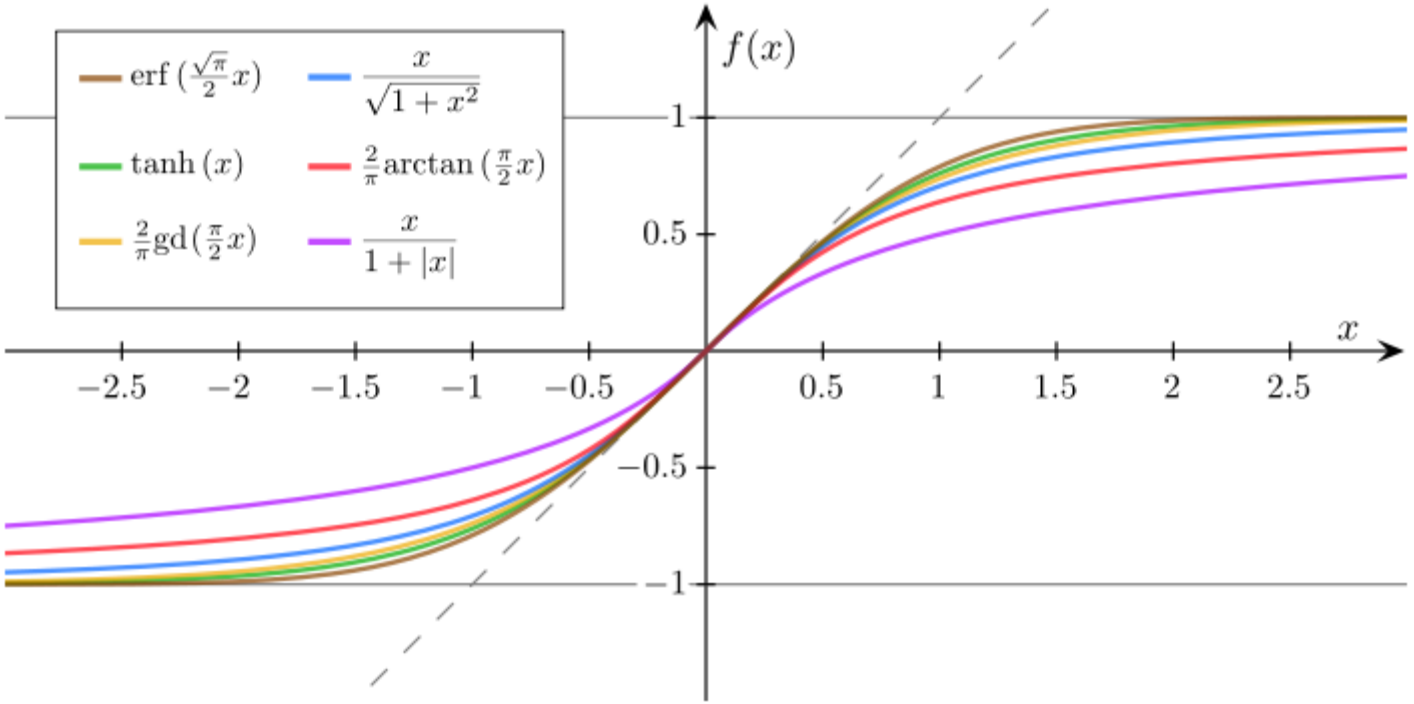
- Step function non-differentiable
- Apply a sigmoid function to smooth it out



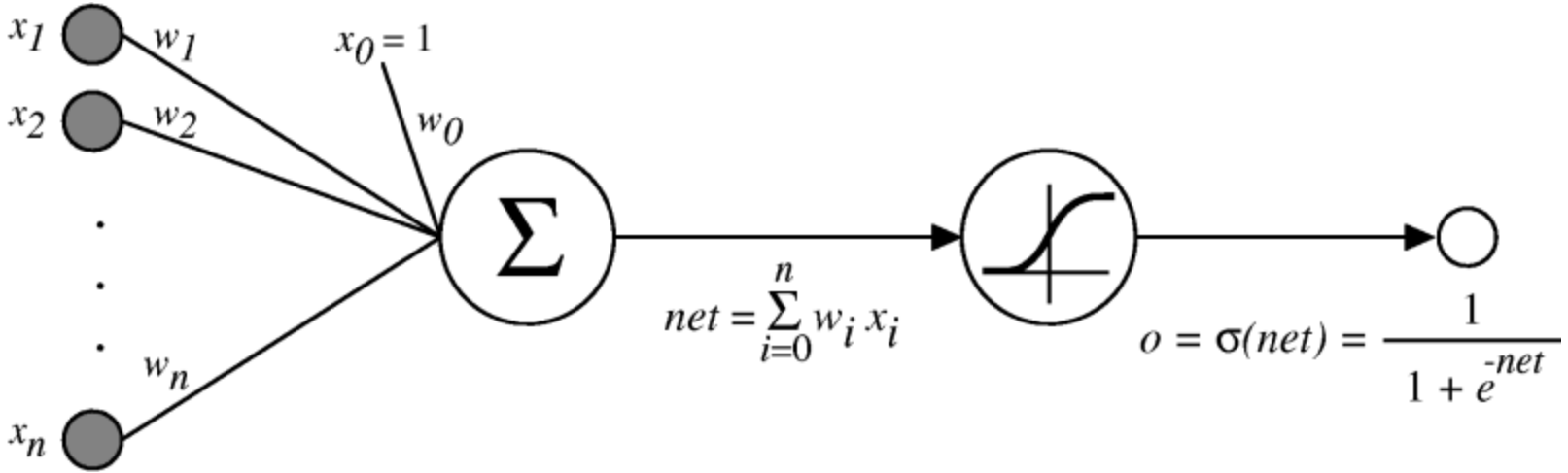
$$z = w_1 x_1 + w_2 x_2 + \ldots + w_n x_n$$

$$y = \frac{1}{1 + e^{-z}}$$

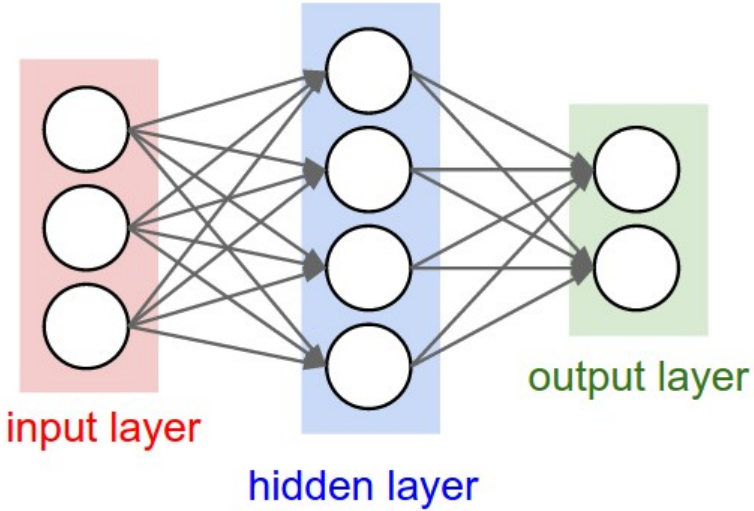
Sigmoid Functions

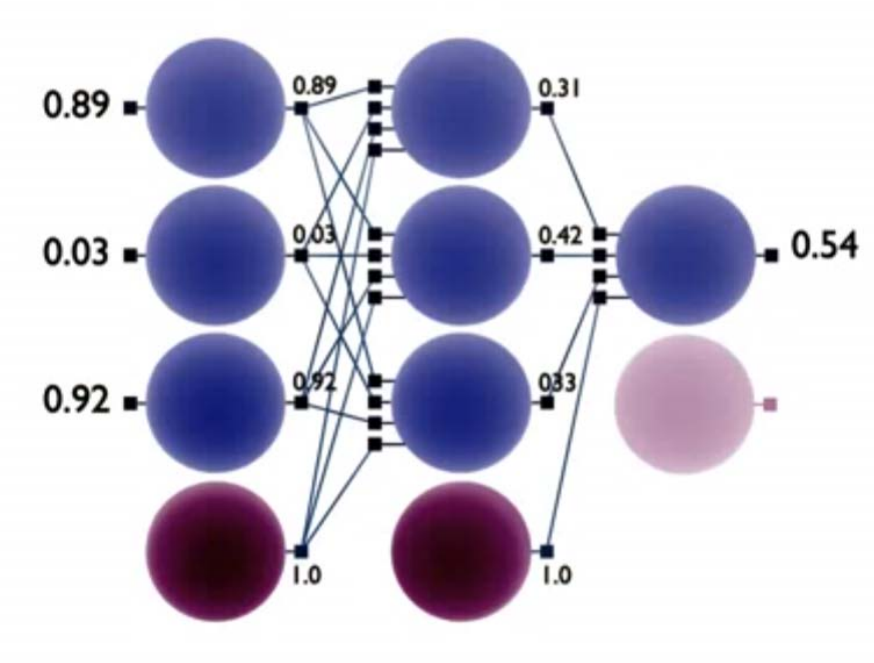


The Improved Neuron



Neural Net Structure





Neural Net Structure

- Common Structure: Fully Connected
- If we start with a fully connected network, then unimportant connections will eventually train to a low weight
- Network can have an arbitrary structure
- Some research has learned net structure via genetic algorithms

Neural Net Structure

- Demo

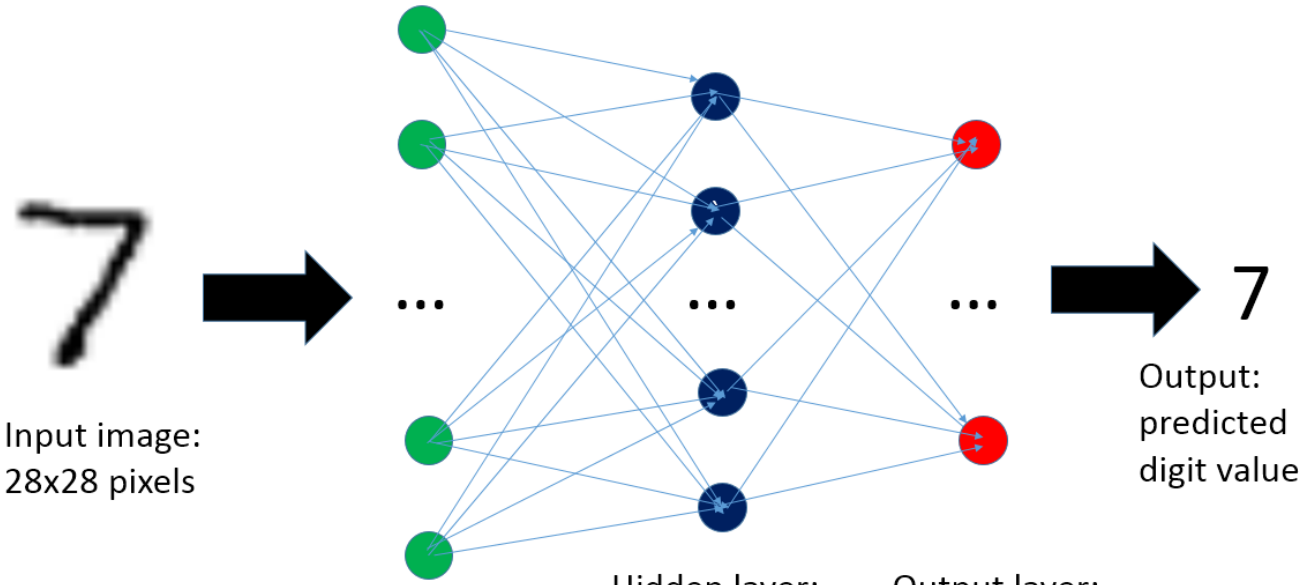
- http://playground.tensorflow.org/

Neural Network Applications

- Neural Networks are a method for performing function approximation

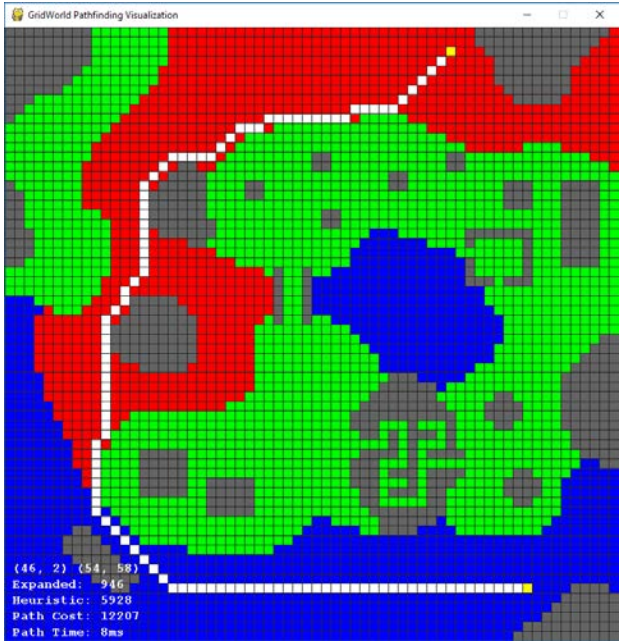
- Given a number of inputs, and desired outputs (training data), try and learn the function that computes correct values

Image Classification



Learning Heuristics

- Compute all pairs shortest path distances on map
- Neural Network
  - Inputs = (x, y) location
  - Target = Path Distance
- Train neural network, then use it as the heuristic function for future calls for A*



AlphaGo

- Combined Deep Learning with Heuristic Search (MCTS)

- Two Networks were trained:
  - Value Network (how good is state S)
  - Policy Network (what move to do at state S)

Learning from Replays

- DeepMind had thousands of professional human games to learn from
- Policy Network
  - Input = State
  - Target = Move the expert human performed
- Value Network
  - Input = State
  - Target = Who won the game from that state

Learning from Self Play

- Once it had learned all it could from humans, it played against itself
- Continued to learn, and get stronger
- Now it learns from a stronger player each time it plays a new game
- Also key: throw in some random moves to learn about states you may not visit

Heuristic Search

- Go has a large branching factor (300)
- Humans have good abstractions of what moves to perform at a state
- AlphaGo learned a policy network
- Now it can narrow the heuristic search to only consider moves in the policy network
- Search also needs a heuristic function to determine how good a state is (value network)