ONCE UPON AN AI

Or getting to know your new robot overlords

With Alex Fitts

Source:
www.neuralnetworksanddeeplearning.com

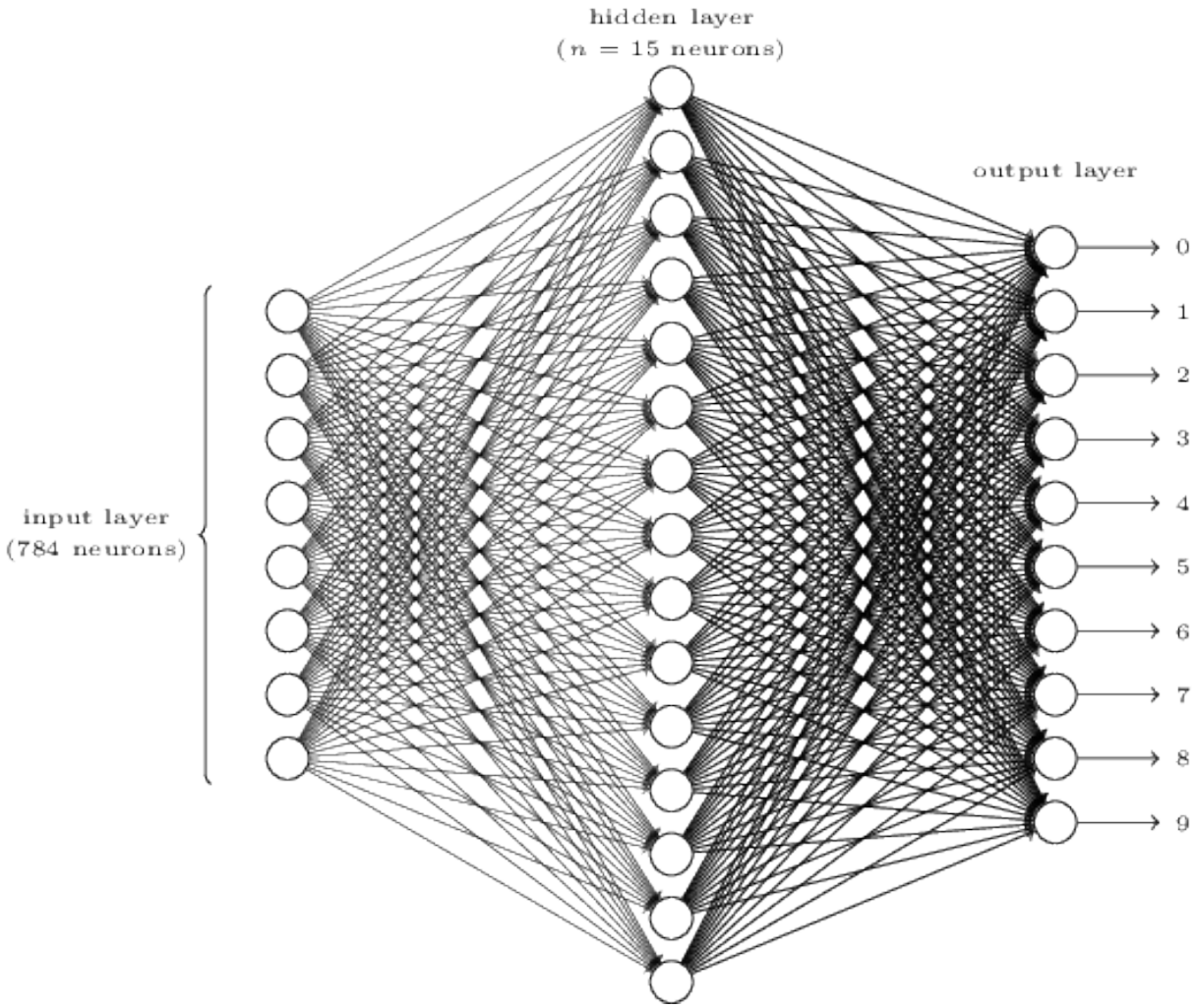
What are neural networks used for?

- Visual Pattern Recognition
  - ‘seeing things’, picture tagging, scanning checks, etc

What are neural networks used for?

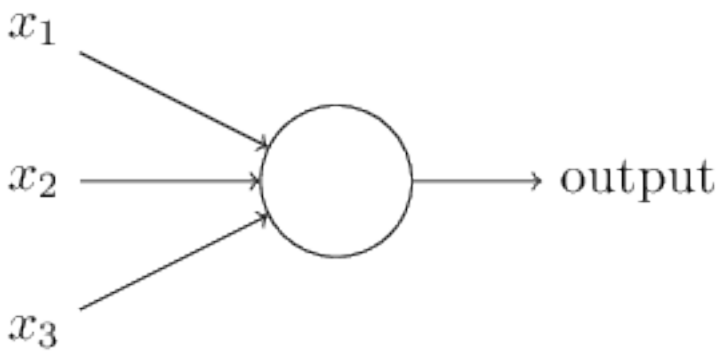
- Visual Pattern Recognition
  - ‘seeing things’, picture tagging, scanning checks, etc
- Natural Language Processing
  - google translate, spam filters, etc.
- Astronomy specific problems
  - Filling DMO simulations with galaxies (Kamdar et al 2015 (a,b))
  - Identifying galaxy morphologies from a picture (Huertas-Company et al. 2015)
- Most tasks can be broken down into two categories:
  - Supervised Learning
  - Unsupervised Learning

What is a neural network?



What is a neuron?

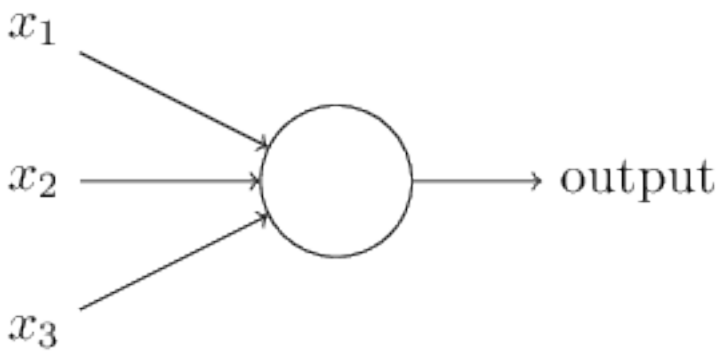
- Perceptron



$$\text{output} = \begin{cases} 0 & \text{if } \sum_j w_j x_j \leq \text{ threshold} \\ 1 & \text{if } \sum_j w_j x_j > \text{ threshold} \end{cases}$$

What is a neuron?

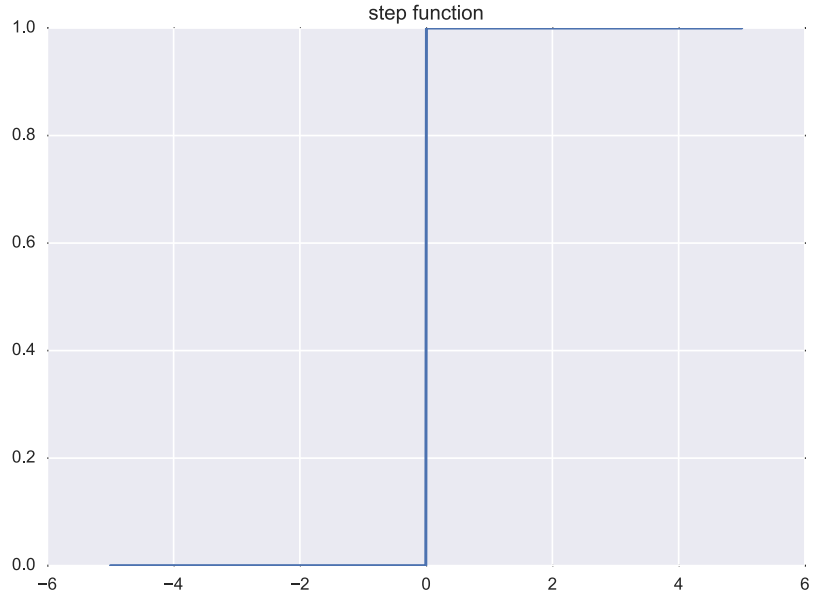
- Perceptron



$$\text{output} = \begin{cases} 0 & \text{if } w \cdot x + b \leq 0 \\ 1 & \text{if } w \cdot x + b > 0 \end{cases}$$

What is a neuron?

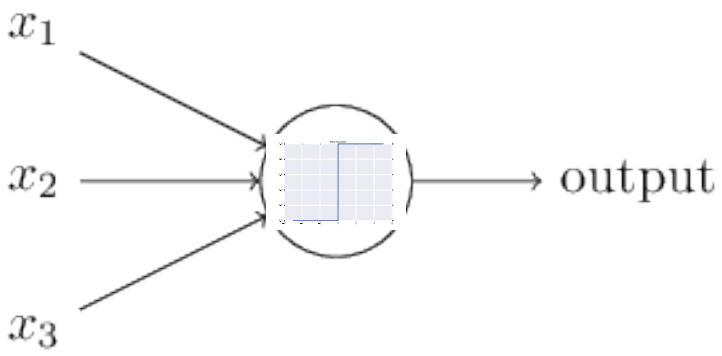
- Perceptron



$$\text{output} = \begin{cases} 0 & \text{if } w \cdot x + b \leq 0 \\ 1 & \text{if } w \cdot x + b > 0 \end{cases}$$

What is a neuron?

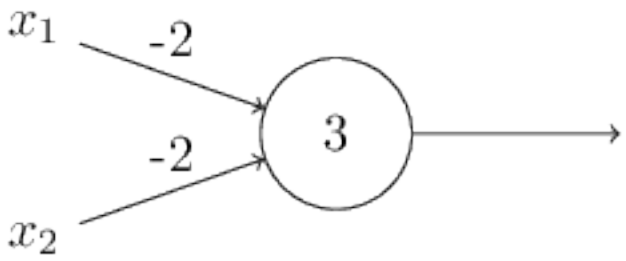
- Perceptron



$$\text{output} = \begin{cases} 0 & \text{if } w \cdot x + b \leq 0 \\ 1 & \text{if } w \cdot x + b > 0 \end{cases}$$

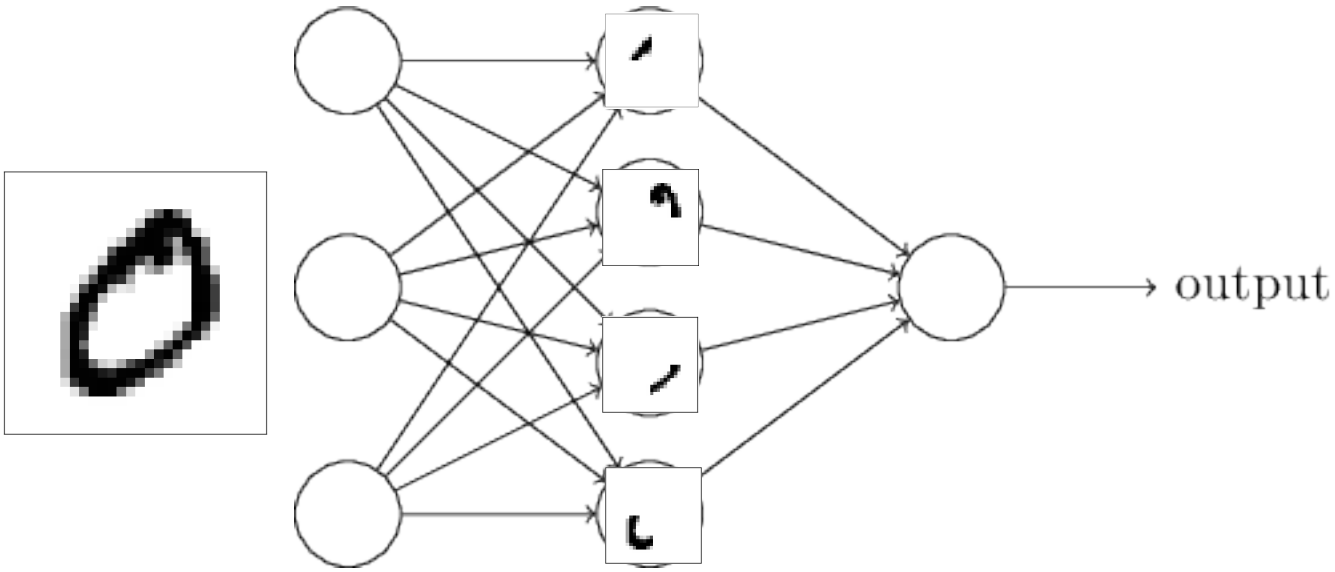
What is a neuron?

- Perceptron



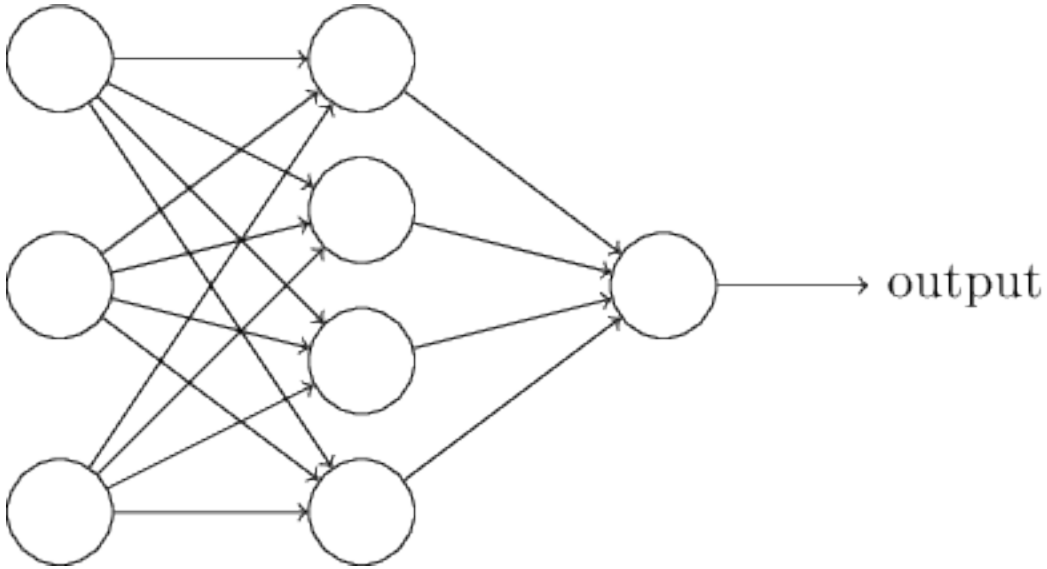
$$\text{input } 00 \Rightarrow (-2) * 0 + (-2) * 0 + 3 = 3 \Rightarrow \text{output } 1$$
$$\text{input } 11 \Rightarrow (-2) * 1 + (-2) * 1 + 3 = -1 \Rightarrow \text{output } 0$$

Architecture of Neural Netowrks

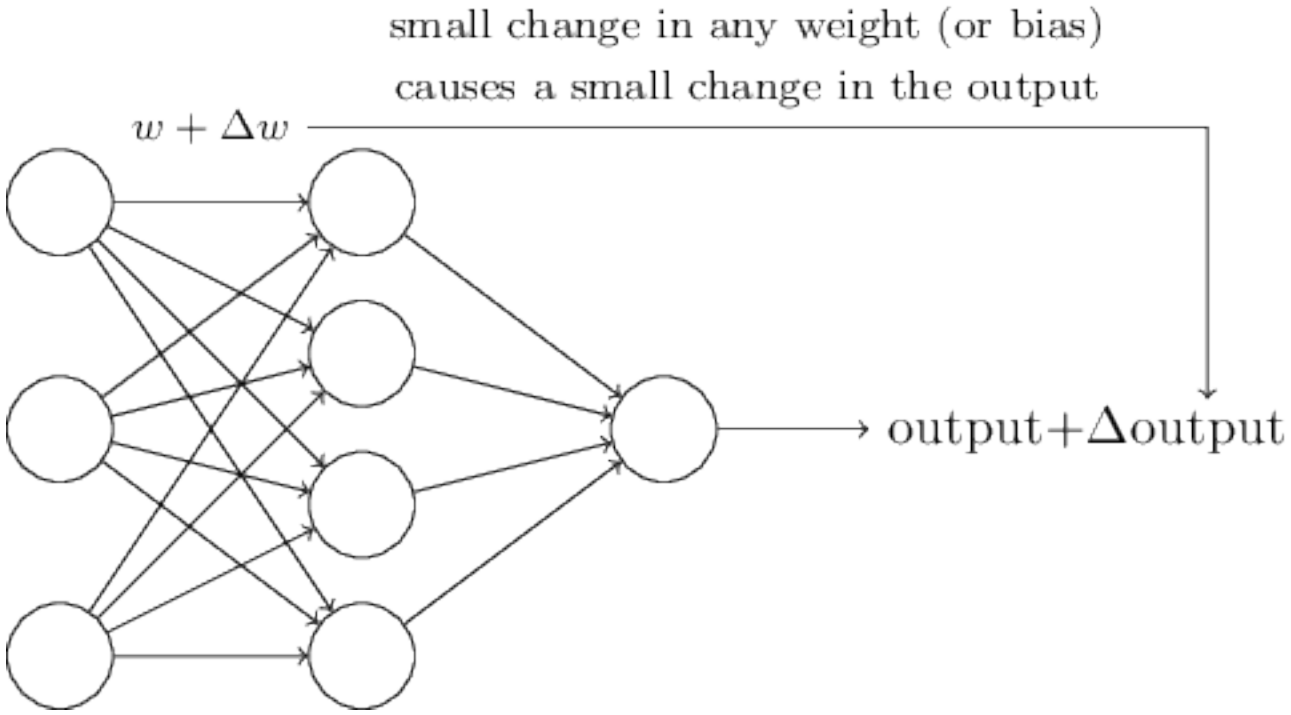


Architecture of Neural Netowrks

- Need certain features to be able to learn
  - At extreme values, needs to give 0 or 1 ✔

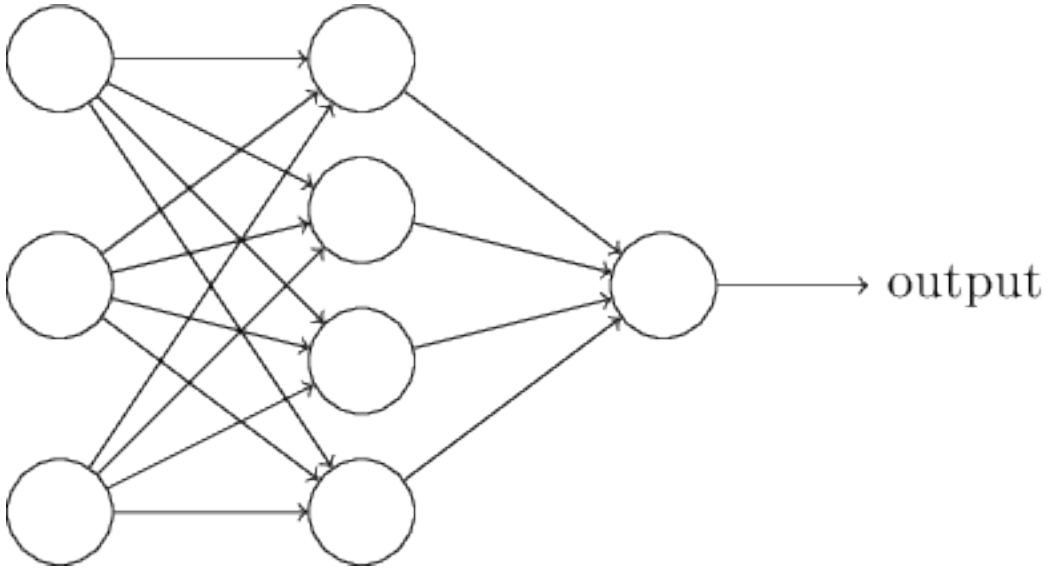


Architecture of Neural Netowrks



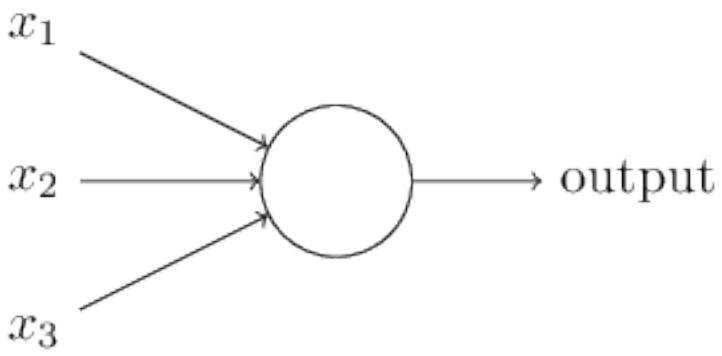
Architecture of Neural Netowrks

- Need certain features to be able to learn
  - At extreme values, needs to give 0 or 1 ✔
  - Small changes in weights leads to small changes in output ✗



‘Better’ Neuron

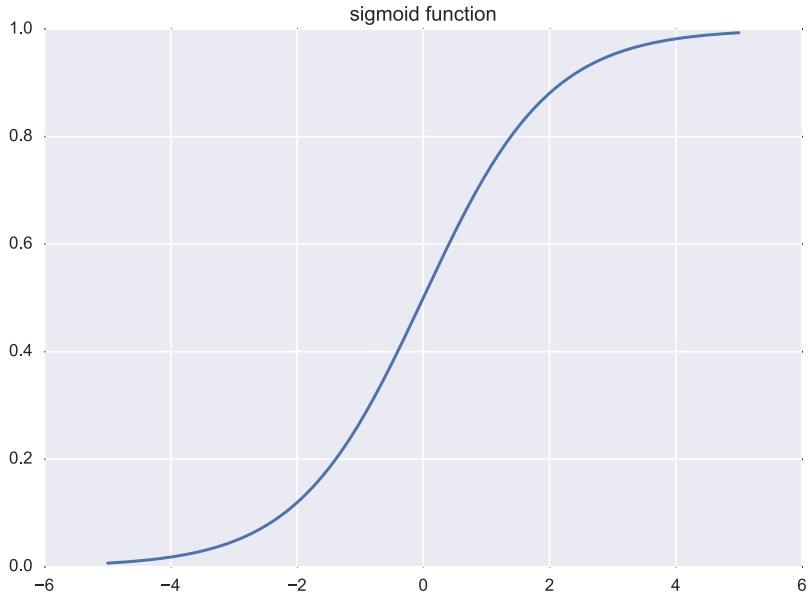
- Sigmoid



$$\sigma(z) \equiv \frac{1}{1+e^{-z}} = \frac{1}{1+\exp(-\sum_j w_j x_j - b)}$$

‘Better’ Neuron

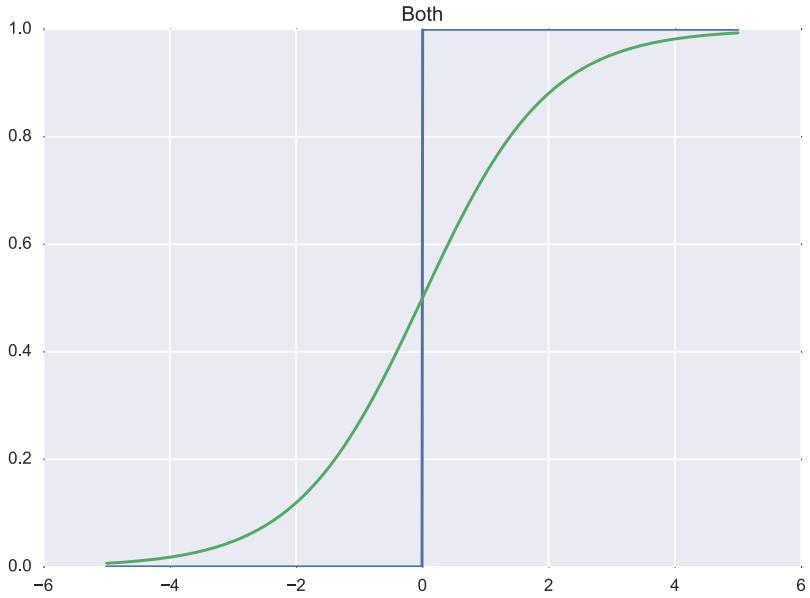
- Sigmoid



$$\sigma(z) \equiv \frac{1}{1+e^{-z}} = \frac{1}{1+\exp(-\sum_j w_j x_j - b)}$$

‘Better’ Neuron

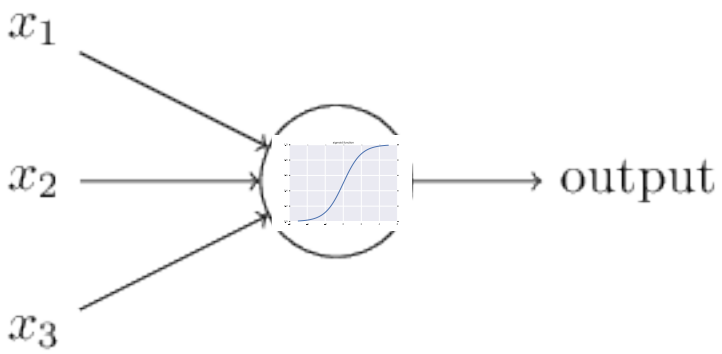
- Sigmoid



$$\sigma(z) \equiv \frac{1}{1+e^{-z}} = \frac{1}{1+\exp(-\sum_j w_j x_j - b)}$$

‘Better’ Neuron

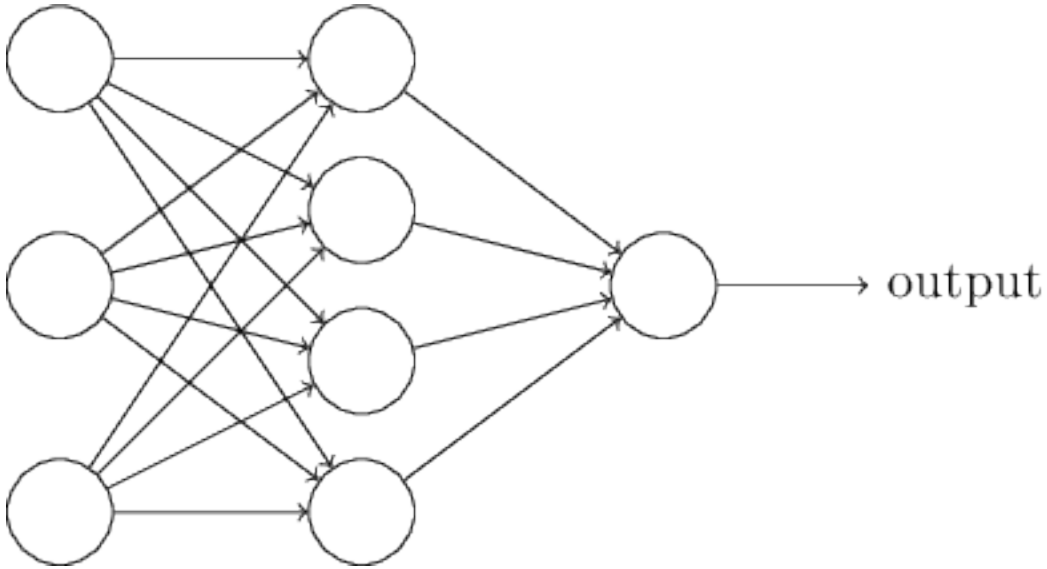
- Sigmoid



$$\sigma(z) \equiv \frac{1}{1+e^{-z}} = \frac{1}{1+\exp(-\sum_j w_j x_j - b)}$$

Architecture of Neural Netowrks

- Need certain features to be able to learn
  - At extreme values, needs to give 0 or 1 ✔
  - Small changes in weights leads to small changes in output ✔



Cost Function

$$C = \frac{1}{n} \sum_{x} C_x$$

Cost Function

- Quadratic Cost Function

$$C(w,b) \equiv \frac{1}{2n} \sum_x \|y(x) - a\|^2.$$

Cost Function

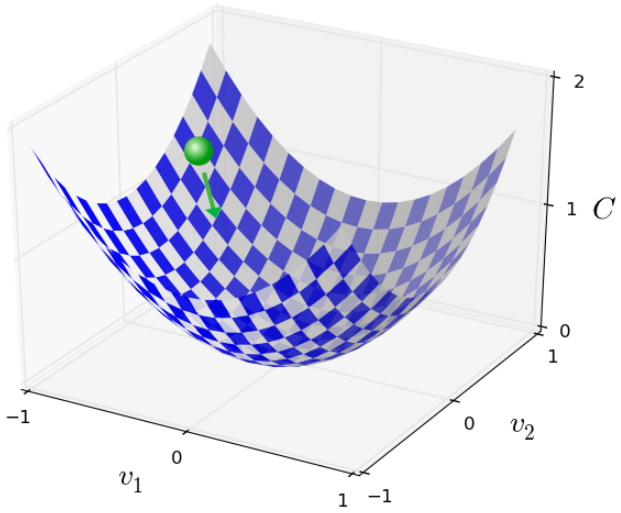
- Quadratic Cost Function

$$C(w,b) \equiv \frac{1}{2n} \sum_x \|y(x) - a\|^2.$$

- New Goal: Minimize the cost function!
- But how? Need a learning algorithm!

Gradient Descent

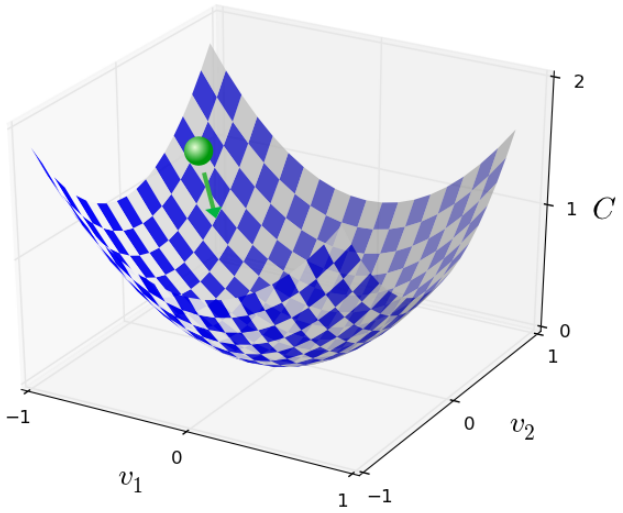
$$\Delta C \approx \frac{\partial C}{\partial v_1}\Delta v_1 + \frac{\partial C}{\partial v_2}\Delta v_2.$$



Gradient Descent

$$\Delta C \approx \nabla C \cdot \Delta v$$

$$\Delta v = -\eta \nabla C$$



$$\sigma(z) \equiv \frac{1}{1 + \exp(-\sum_j w_j x_j - b_j)}$$

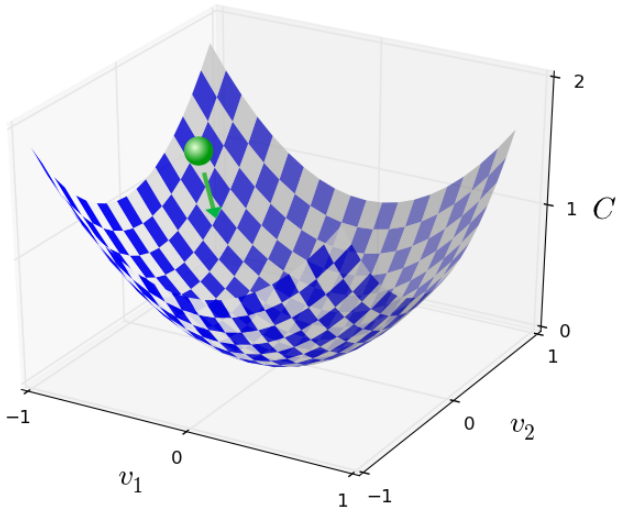
Gradient Descent

$$\Delta C \approx \nabla C \cdot \Delta v$$

$$\Delta v = -\eta \nabla C$$

$$w_k \rightarrow w_k' = w_k - \eta \frac{\partial C}{\partial w_k}$$

$$b_l \rightarrow b_l' = b_l - \eta \frac{\partial C}{\partial b_l}$$



Gradient Descent

$$C = \frac{1}{n} \sum_{x} C_x$$

Gradient Descent

$$\nabla C = \frac{1}{n} \sum_x \nabla C_x$$

Stochastic Gradient Descent

$$X_1, X_2, \ldots, X_m$$

$$\frac{\sum_{j=1}^{m} \nabla C_{X_j}}{m} \approx \frac{\sum_x \nabla C_x}{n} = \nabla C$$

$$\nabla C \approx \frac{1}{m} \sum_{j=1}^{m} \nabla C_{X_j}$$

Stochastic Gradient Descent

$$X_1, X_2, \ldots, X_m$$

$$w_k \to w_k' = w_k - \frac{\eta}{m} \sum_j \frac{\partial C_{X_j}}{\partial w_k}$$

$$b_l \to b_l' = b_l - \frac{\eta}{m} \sum_j \frac{\partial C_{X_j}}{\partial b_l}$$

Interchangeable Parts

- Activation functions
  - How the weights + input determine the output
  - Perceptron, Sigmoid (logistic) function…
- Cost/Loss/Objective functions
  - A way to measure how well we're fitting the desired behavior
  - Quadratic cost (Mean squared error)…
- Learning Algorithm
  - How our network minimizes the cost function
  - Gradient Descent, Stochastic Gradient Descent (SGD)…
- Hyper-parameters
  - Various parameters that are tuned by us
  - Learning rate, mini-batch size, epochs of training, layers in neural network, etc...

The hard part has already been done!

- TensorFlow, pyTorch, Theano, etc…

01000111 01101111 01101111 01100100 00100000 01101010 01101111 01110010 01100010 00100001