Introduction to Machine Learning

Darcy Meyer
09/30/2021

BOSTON UNIVERSITY
MACHINE INTELLIGENCE
COMMUNITY

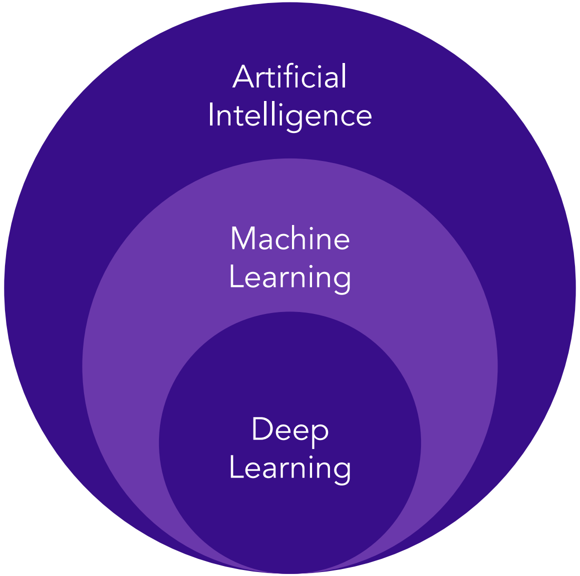
Attendance

https://forms.gle/1YDi83J3rvaWUanM8

What is Machine Learning?

Machine Learning is the field of study that gives computers the capability to learn without being explicitly programmed

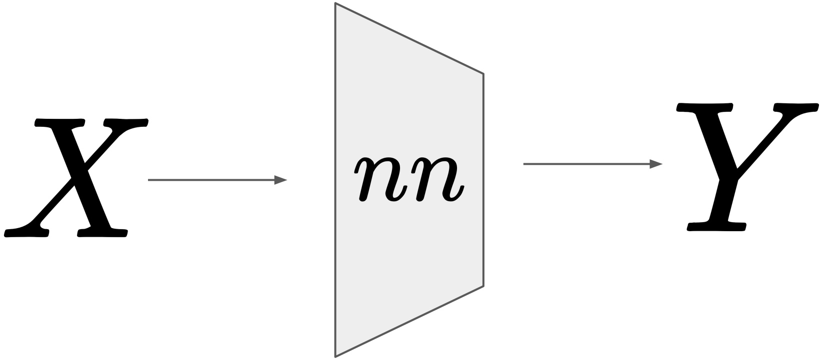
What is Deep Learning



Deep learning is a subset of machine learning

What is Deep Learning

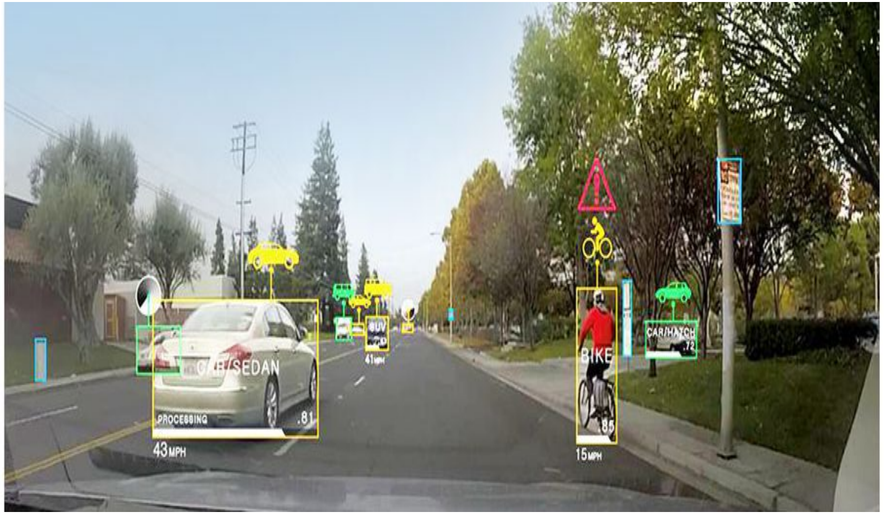
Deep learning learns from data using a class of functions known as Neural Networks



A neural network maps an input to an output

Applications of Deep Learning

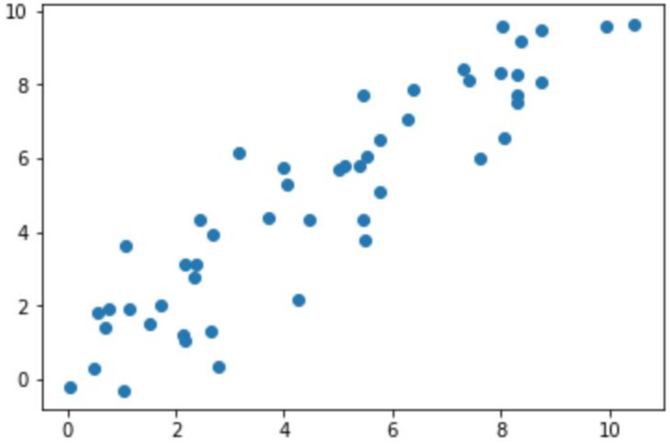
1. Cool things using deep learning
    a. Computer Vision
        i. Tesla recognizing items on a street
    b. Text generation
        i. An algorithm was trained to create a similar Shakespeare piece
    c. Image recognition
        i. Classifying what a certain picture contains
        ii. Facebook photo tagging
    d. Many more…



Learning from data

We have some data D

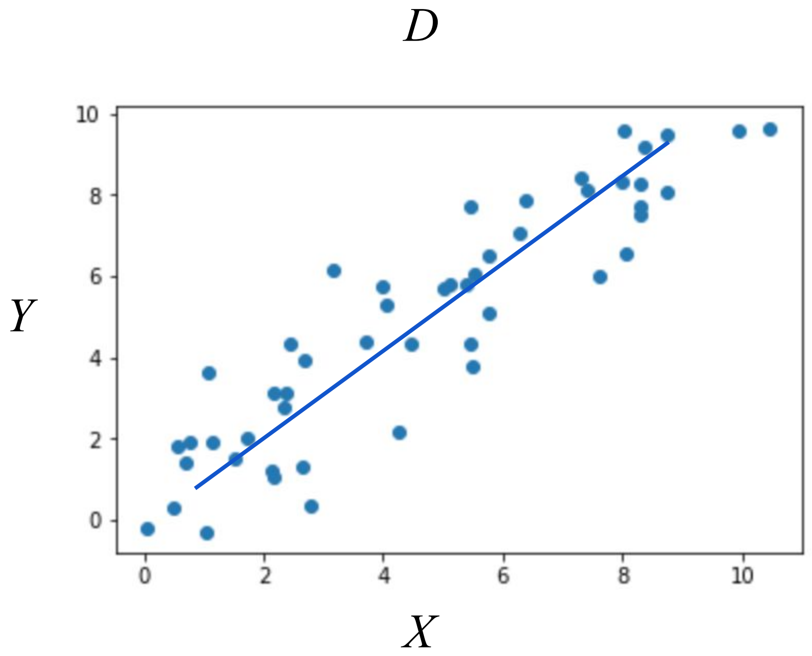
D



Y

X

Make an assumption about D



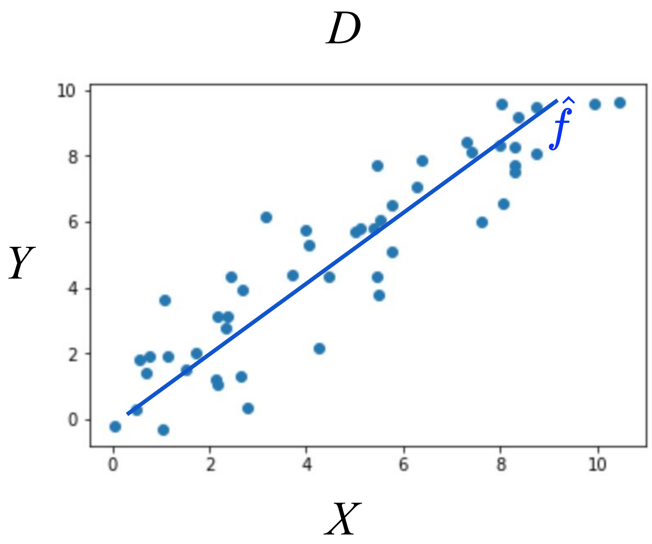
$$y = b + mx$$

$$\downarrow$$

$$\hat{f} = \theta_0 + \theta_1 x$$

What is learning?

The approximation of some unknown function f based on some data D.



$$f : X \to Y$$

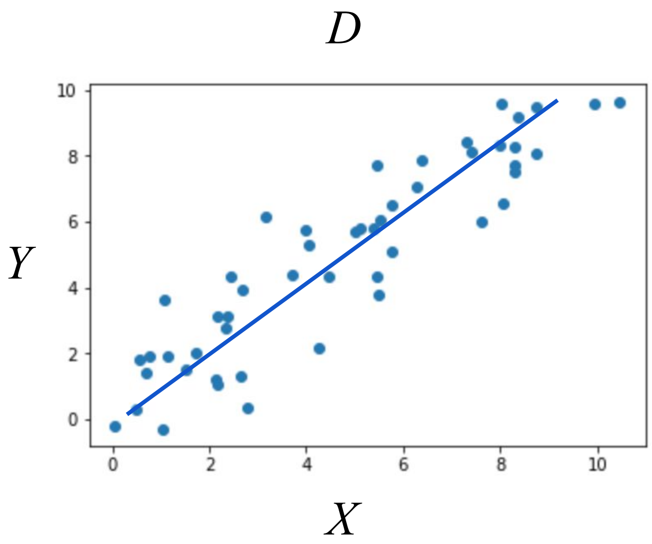
$$\hat{f} = \theta_0 + \theta_1 x$$

How do we set the parameters?
How do we know what assumptions to make?

Linear Regression

We have some data D

The approximation of some unknown function f based on D.



$$f : X \to Y \text{ or } y = b + mx$$

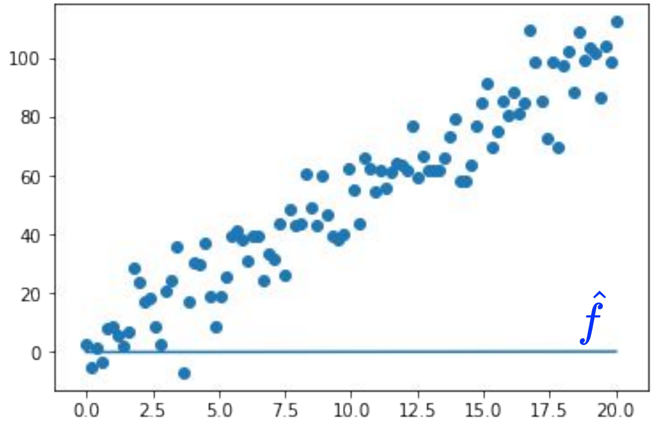
$$\downarrow$$

$$\hat{f} = \theta_0 + \theta_1 x$$

Initially, the theta parameters act as an arbitrary estimate for the parameters we are trying to learn.

Initial Guess

In this example we can instantiated our guesses (θ_0, θ_1) to be values close to zero. For example, the guesses in the example below are -0.01 and 0.01.



First Guess/Initialization:

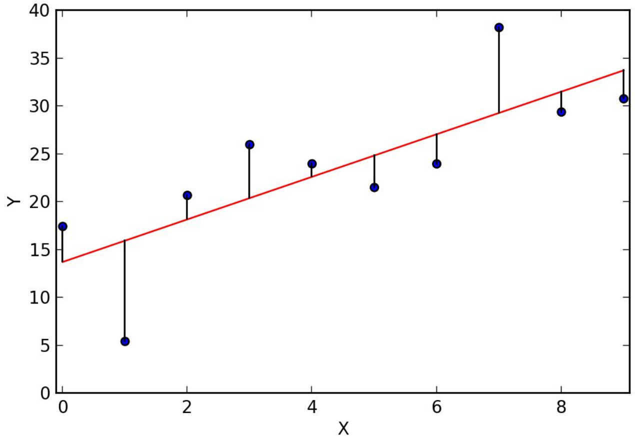
$$\hat{f} = -0.01 + 0.01x$$

Computing the cost

In order to train our neural network, we need some way to tell us how far off its estimate was from the actual value.

We define the cost function, $J(\hat{y}, y)$ as the sum of losses, $\sum_{i=0}^{m} L(\hat{y}, y)$

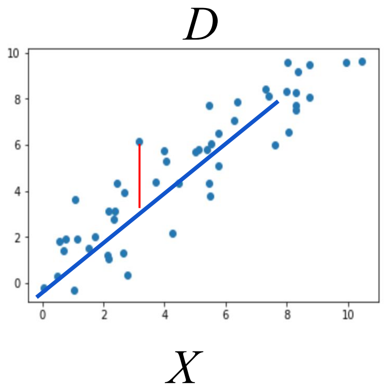
a. Loss = Error for a single training example
b. Cost = Sum of all Losses
c. Y is our actual point
d. Y hat is our estimated point



Compute the cost

Calculate difference between predicted output and actual data

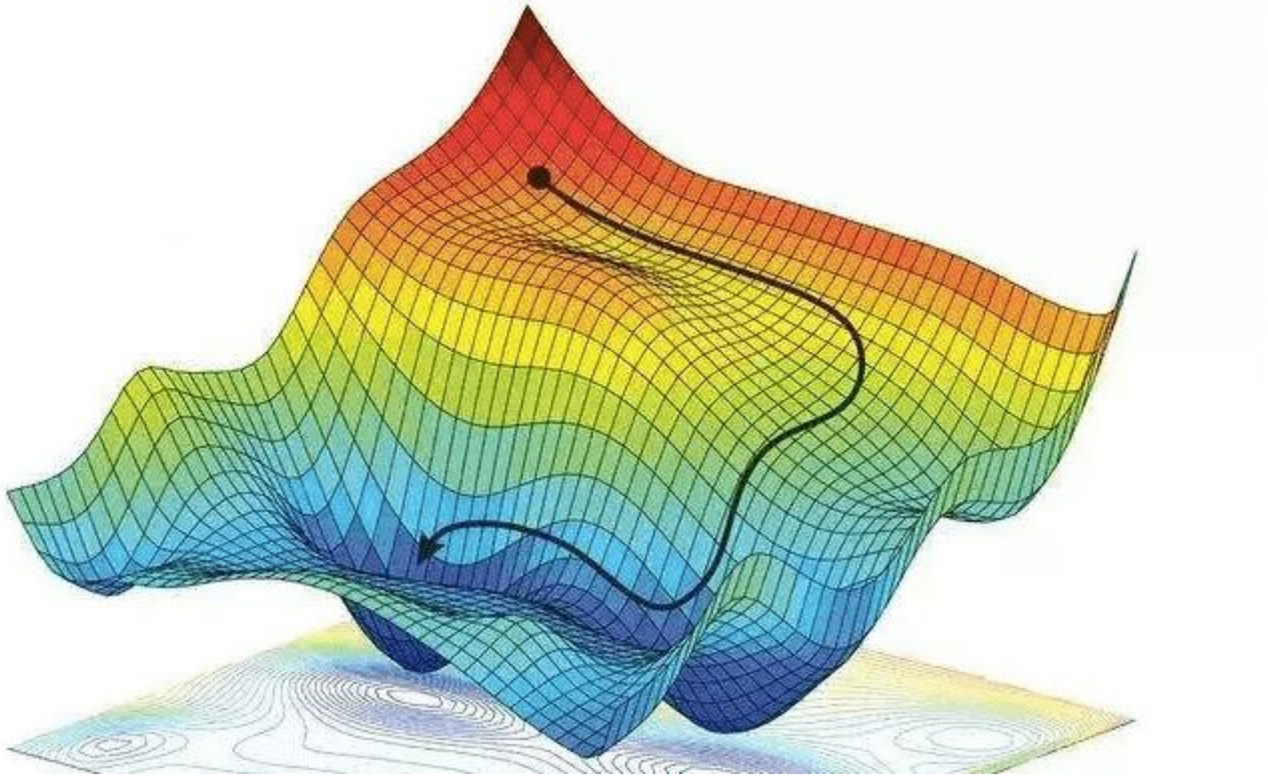
$$J(\hat{y}, y) = \frac{1}{2n} \sum_{i}^{n} (y_i - \hat{y}_i)^2$$



Where i is the ith training example and n is the number of training examples

Intuition: if y hat is far away from y, the square will be large

Cost function for gradient descent



Update Rule

To find the slope, we compute the derivative of the cost (gradient) with respect to a single parameter.

Scalar learning rate

$$\theta_j := \theta_j - \alpha \overbrace{\frac{\partial}{\partial \theta_j} J(\theta)}^{\text{Also written } \nabla J(\theta)}$$

Individual weights

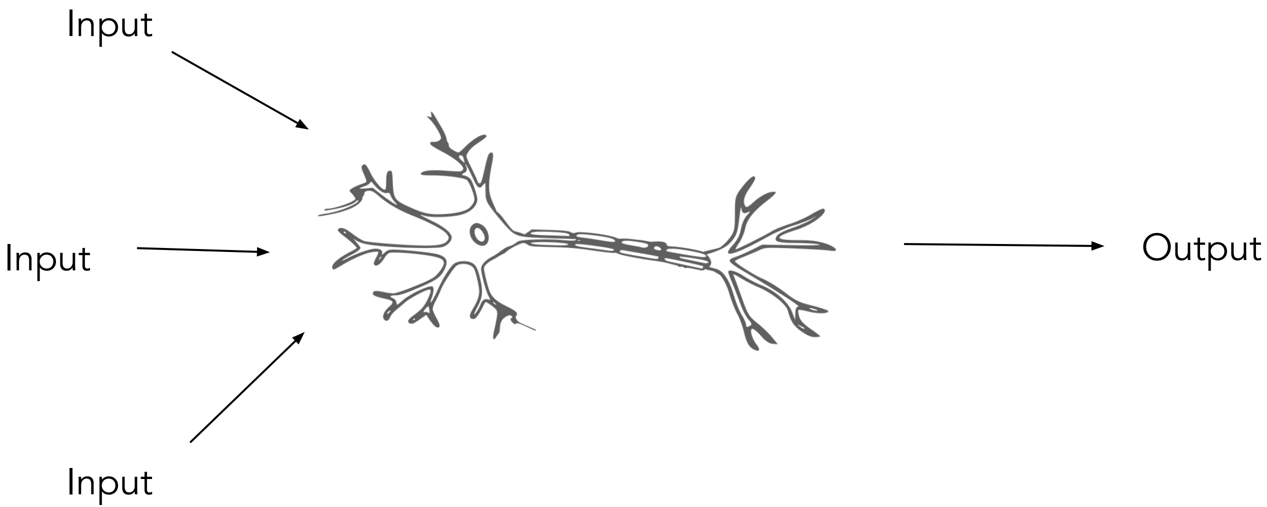
Vector of weights

Cost/objective/loss function

Linear Classifier

A Single Neuron

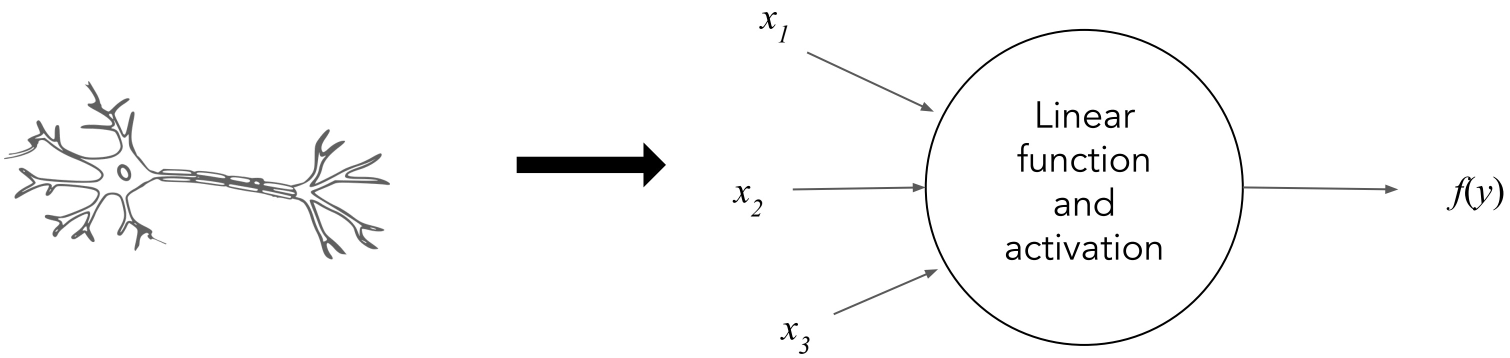
- Detects electrical pulses as inputs, and outputs one (or more) signal based on the inputs



Modeling a Single Neuron

- Want a function that takes n inputs and produces one output
  - Choose a simple linear function: $y = a_0 + a_1x_1 + a_2x_2 + \dots + a_nx_n$
- We want the output to be in a certain range (e.g. between 0 and 1)
  - Apply an "activation function" to y, e.g. $f(y) = 1/(1+e^{-y})$ (sigmoid function)

This is called a "perceptron"



What can we use it for? A Linear Classifier

Linear Classifier: given some input data, output "yes" or "no"

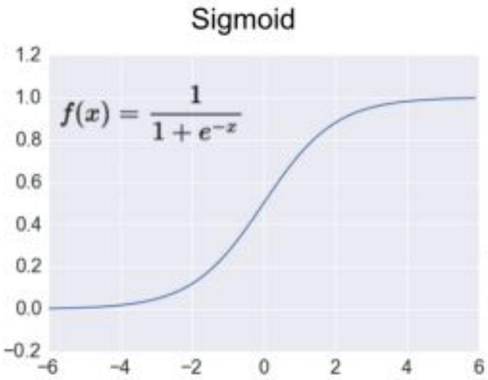
- Input data is a set of numbers $x_1, x_2, \ldots x_n$
- $z = a_0 + a_1x_1 + a_2x_2 + \ldots + a_nx_n$ (where a_i are the weights of this model)
- Apply an activation function $f(z) = 1/(1+e^{-z})$ (squeeze outputs between 0 and 1)
- Interpret ~1 as "yes" and ~0 as "no"

Note that y is a dot product between $\boldsymbol{a}$ and $\boldsymbol{x}$ (first element of $\boldsymbol{x}$ (x_0) is 1)

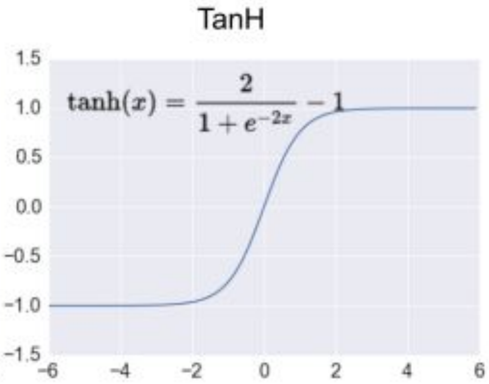
Activation Functions

Activation Functions model nonlinear data by taking inputs and comparing them to a threshold. This allows us to model non-linear data.

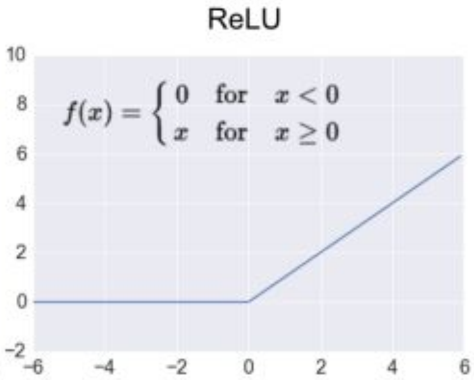
Sigmoid: output is between 0,1



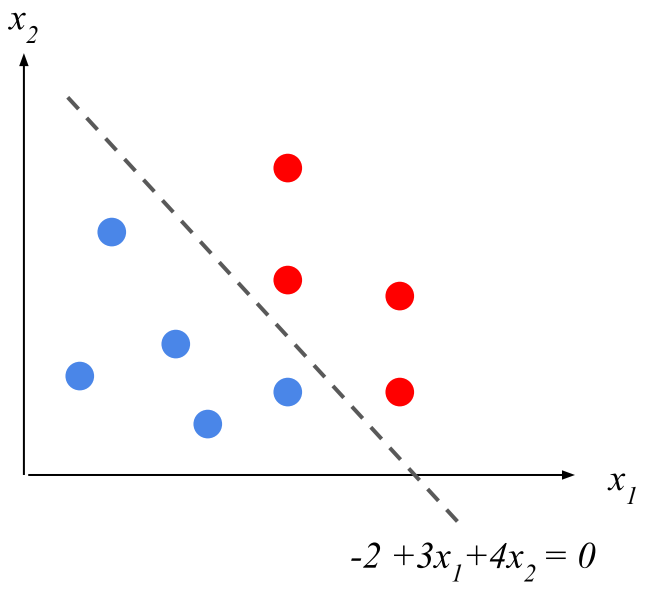
Tanh: output is between -1,1



ReLu: output is positive real numbers

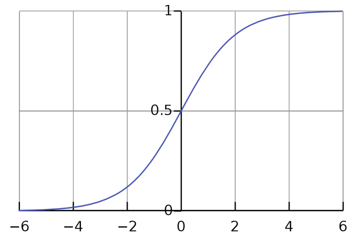


Linear Classifier - Example



Parameters: $a_0 = -2$, $a_1 = 3$, $a_2 = 4$

- $z = a_0 + a_1 x_1 + a_2 x_2$
- $y = f(z) = 1/(1+e^{-y})$



- y around 1 → red
  y around 0 → blue

Training a Linear Classifier

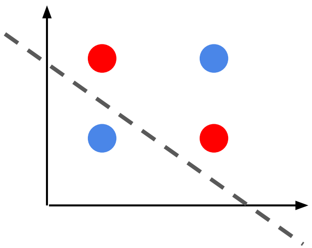
Given some examples of labeled data points, how do we find a_0, a_1, … ?

Gradient Descent

- Start with random parameters, use them to predict an output
- Compare the predicted output to the correct output by computing a cost
- Find the partial derivative of the cost with respect to each parameter
- Update the parameters to decrease the cost
- Repeat many times

Limitations of Linear Classifiers

- What if we want to model something more complicated?
  - Data isn't linearly separable
  - Want to approximate any arbitrary function

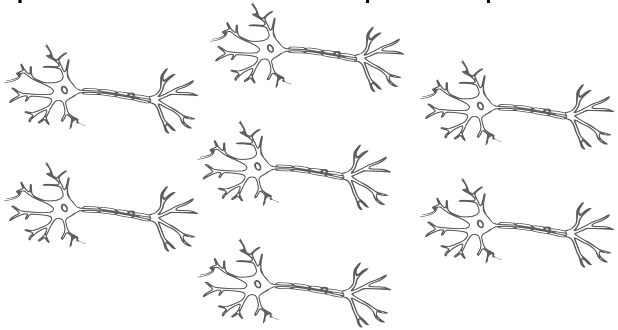


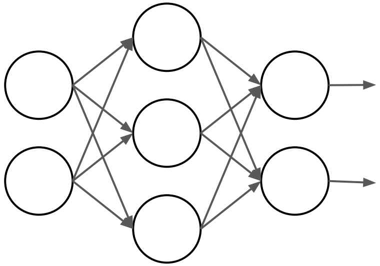
Dogs?

Neural Networks

Neural Networks

Use outputs from some perceptrons as inputs to more perceptrons



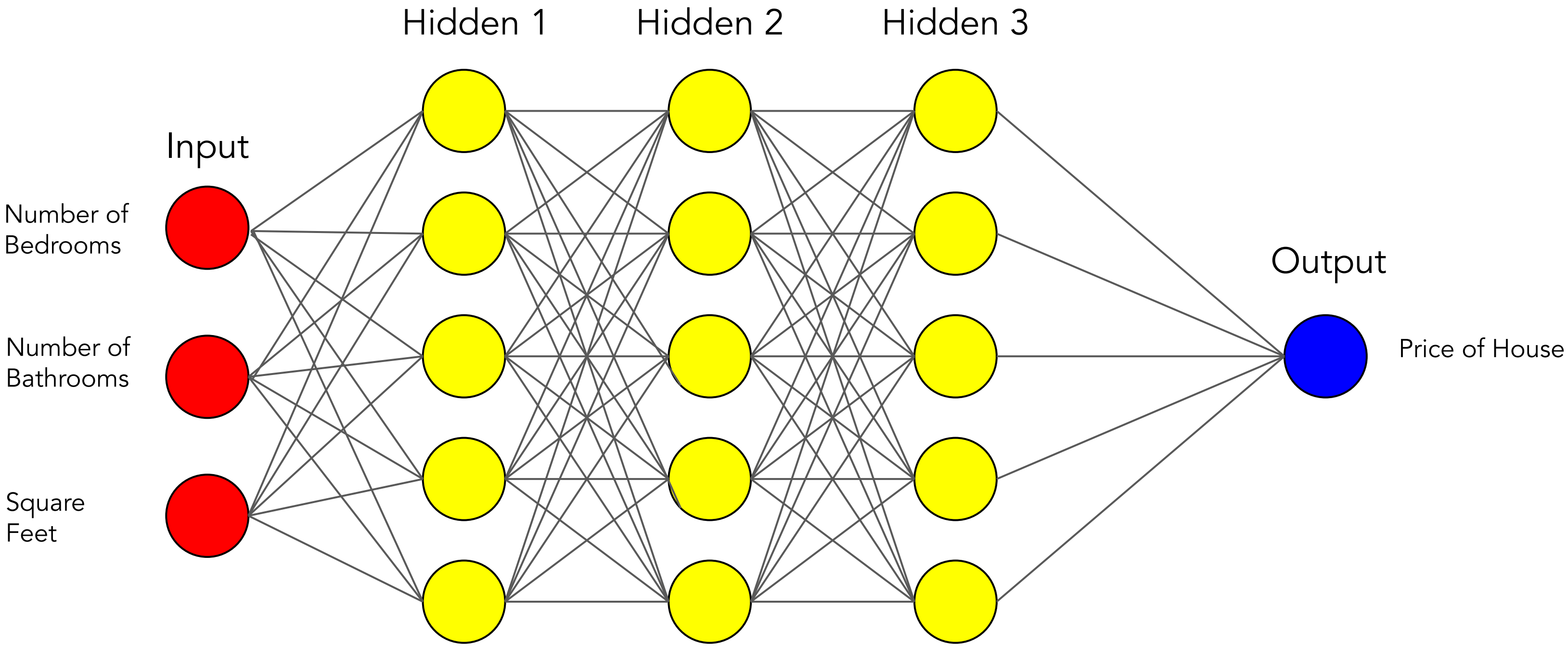


- Each single perceptron can detect one simple thing (one “feature”)
- Many perceptrons assembled together can detect complex things
- This is a “multilayer perceptron” or neural network

Steps to Train a NN

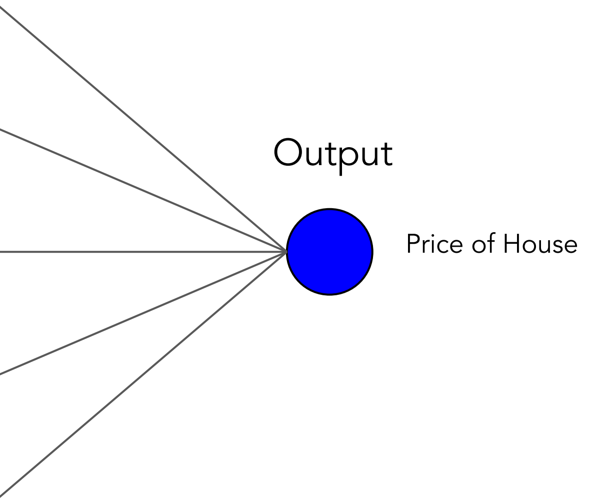
Forward propagation

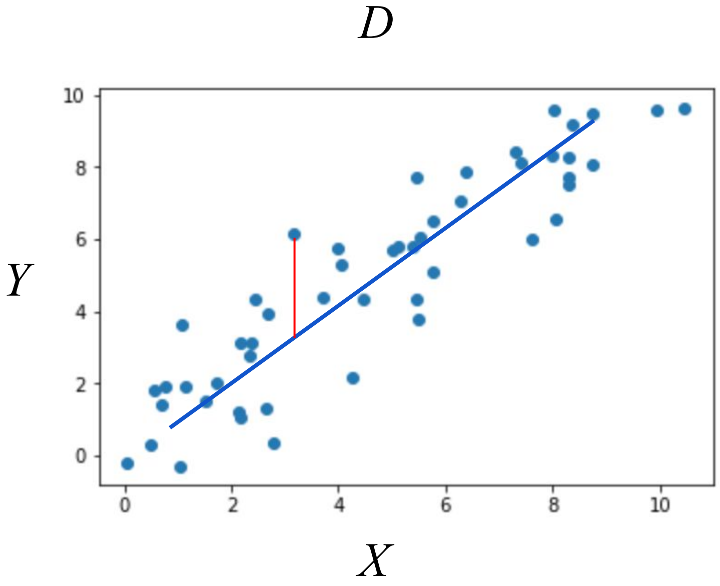
Push example through the network to get a predicted output



Compute the cost

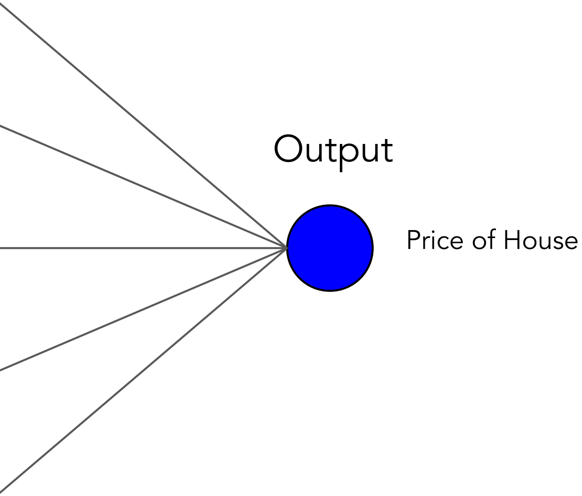
Calculate difference between predicted output and actual data





Compute the cost

Calculate difference between predicted output and actual data



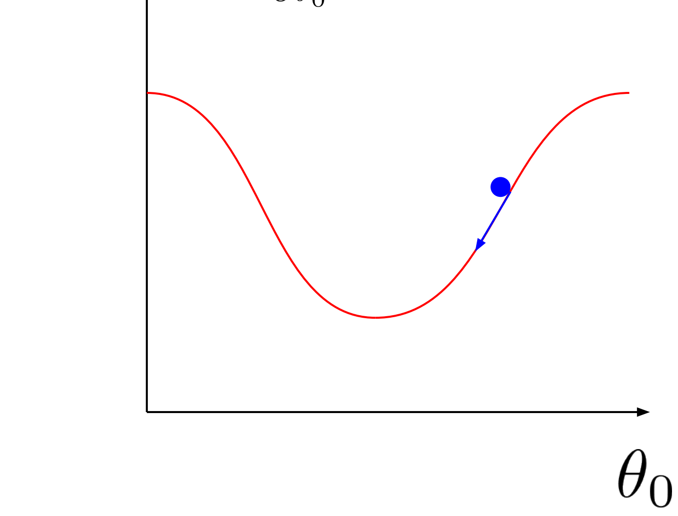
$$J(\theta) = \frac{1}{2m}\sum_{i}^{m}(y_i - \hat{y}_i)^2$$

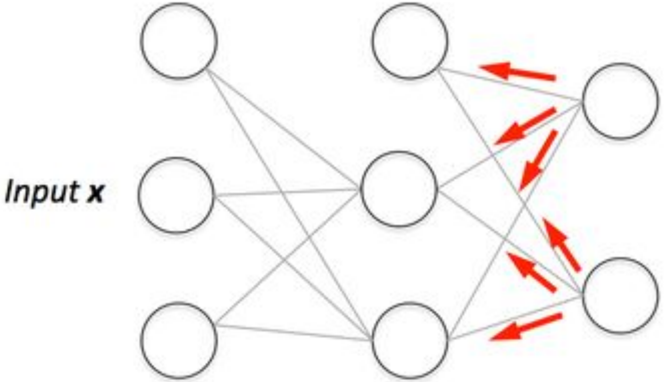
Where i is the ith training example and m is the number of training examples

Backward propagation - “Update”

Push back the derivative of the error and apply to each weight, such that next time it will result in a lower error

$$-\frac{\partial}{\partial\theta_0}J(\theta) =$$





Programming Exercise

https://bit.ly/2Y6RMhS

Eboard positions available!

https://forms.gle/aV12v3iJVMnRb1xo6