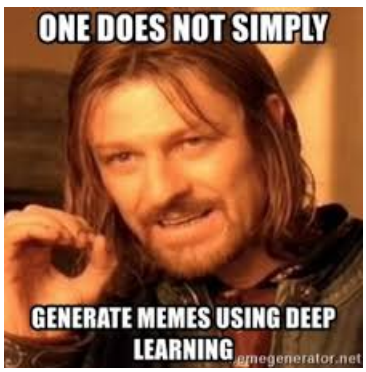
What is Machine Learning?

Actually it's something like this.</big>

Source:xkcd

Why Bother to learn?

Because...

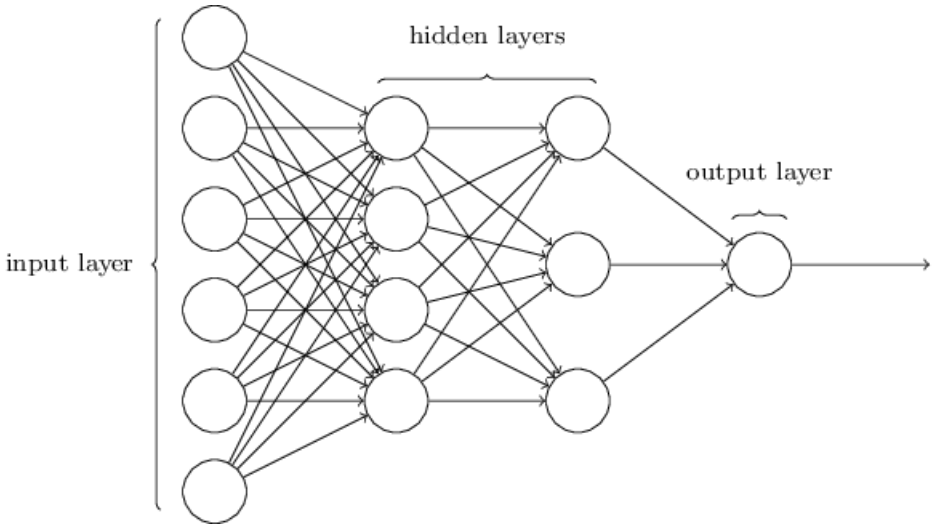


Source: Google Images

Neural Networks and Deep Learning

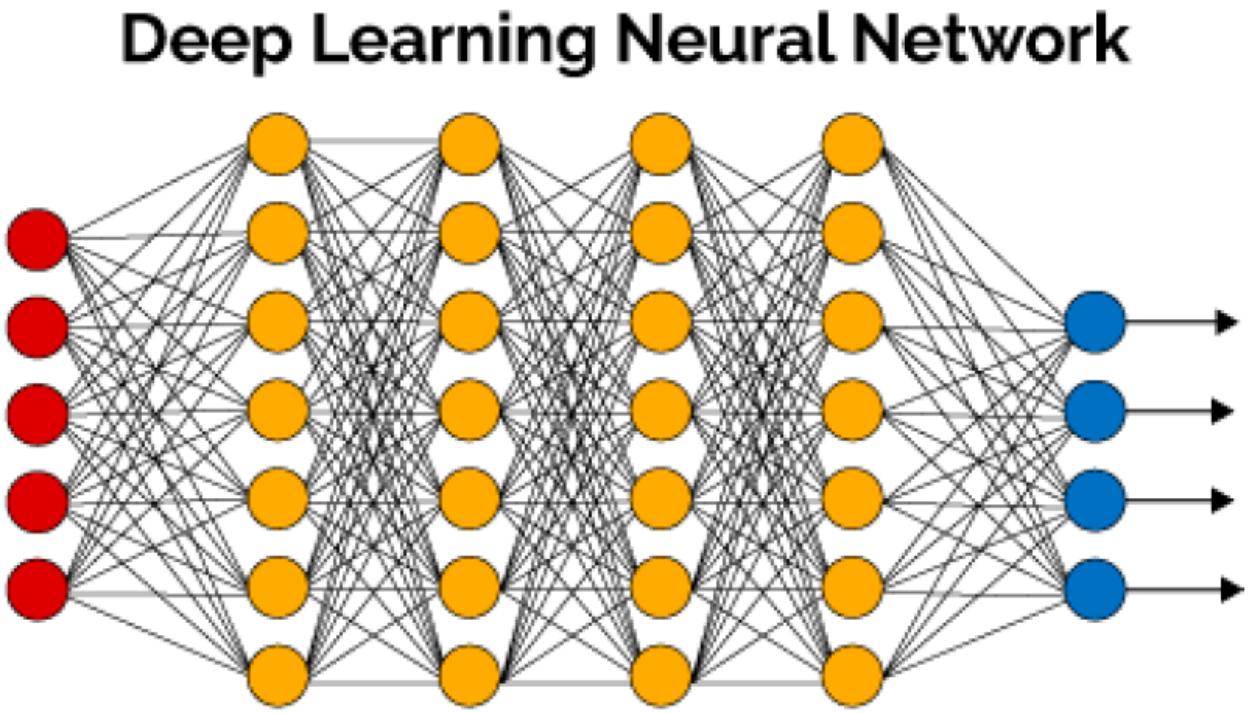
What is a Neural Network?

Let's skip the definitions for which there's wikipedia at your disposal
For now I'll say they are universal function approximators
Have a look at this

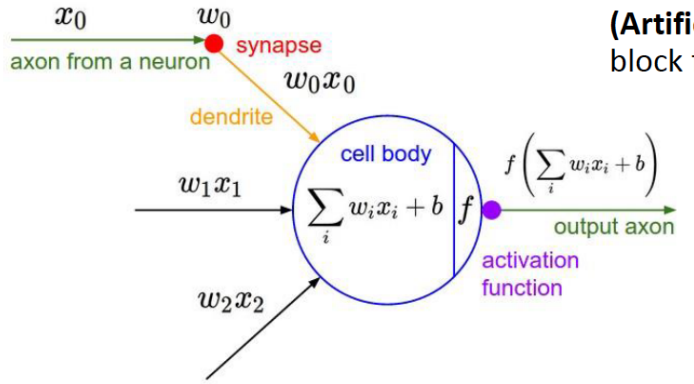


Looks Familiar?

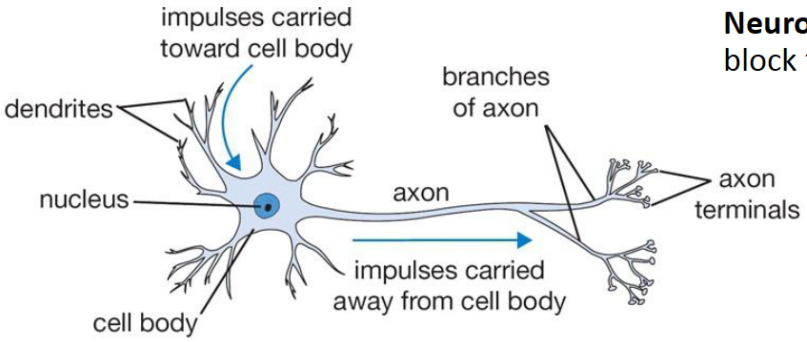
Lets go deep.



Neuron: Biological Inspiration for Computation



(Artificial) Neuron: computational building block for the "neural network"

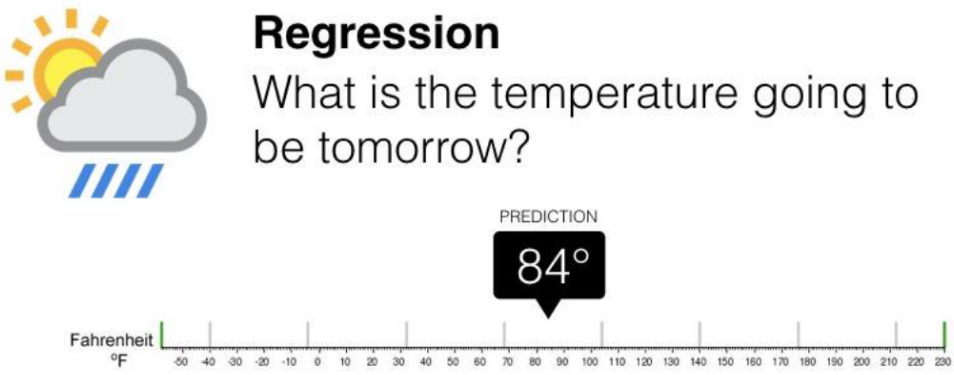


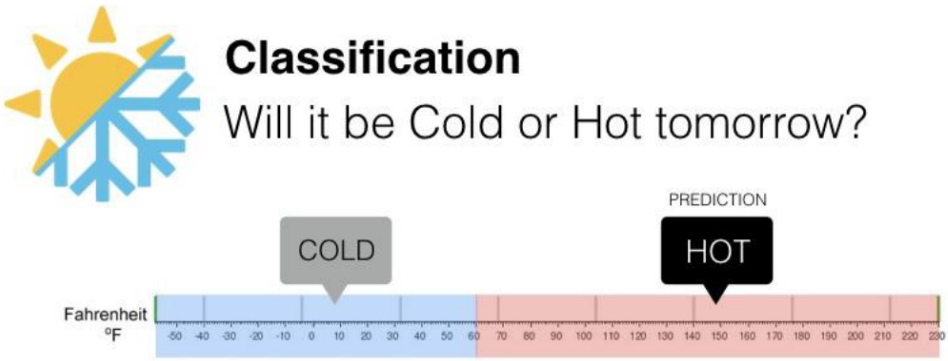
Neuron: computational building block for the brain

Supervised Learning with Neural Network

Input (X)	Output (y)	Application	
Home Features	Price	Real Estate	**Standard Neural Networks**
Ad, User info	Click on Ad? (0/1)	Online Advertising	**Standard Neural Networks**
Image	Object (1,2,----,100)	Photo Tagging	**CNN**
Audio	Text Transcript	Speech Recognition	**RNN**
English	Chinese	Machine Translation	**RNN**
Image, Radar Info	Position of other cars	Autonomous Driving	**Custom/Hybrid**

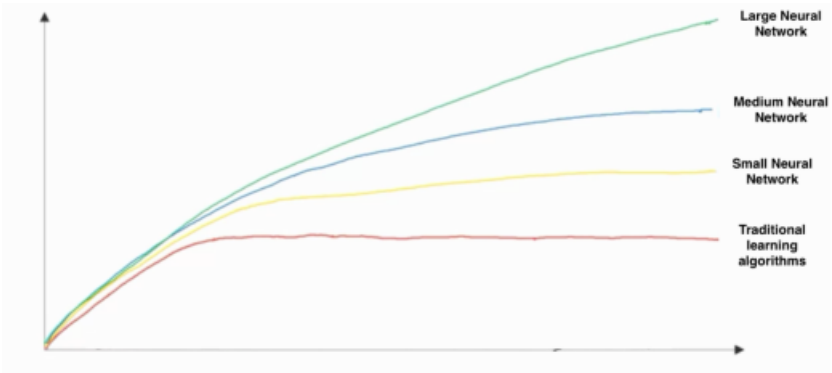
Regression vs Classification





**Wait a minute, What did you say?
Supervised, Unsupervised, What's that?** ![suvsus.png](images/suvsus.png)

Why is DeepLearning taking off?



The vertical axes of the diagram you can see the performance of an algorithm (e.g. it's prediction accuracy) and at the horizontal axes you can see the amount of data (Labelled Data).

You can also see that the performance of traditional learning algorithms (logistic Regression, SVM's etc.) increases at the beginning with an increase of the amount of data but that it plateaus at a certain level and stops improving it's performance.

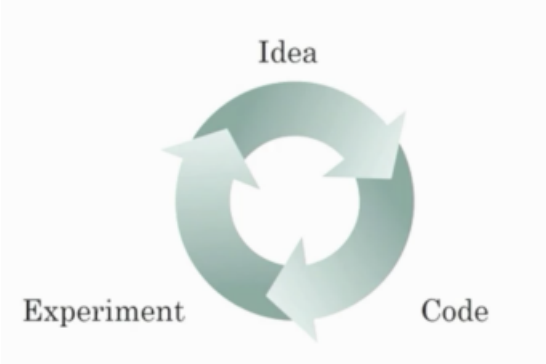
The thing is that we have accumulated huge amounts of data over the last decades where our traditional learning algorithms can't take advantage of, which is where Deep Learning comes into play.

Large Neural Networks (e.g. Deep Learning) are getting better and better the more data you put into them. Andrew NG, a leading AI scientist, said that the three main forces which improve Neural Networks are:

1. Data
2. Computation
3. Algorithms

The recent breakthroughs in the development of algorithms are mostly due to making them run much faster than before, which makes it possible to use more and more data. For an example, a big advancement came from switching from a Sigmoid function (left picture) to a rectified-Linear-Unit function (right picture).

The other reason why fast computation is important is that, the below cycle must be faster.



Bringing more data to a model is almost always beneficial. **Deep Learning approches improve with more data**

Deep Learning Representation

2 Deep Learning representations

For representations:

· nodes represent inputs, activations or outputs

· edges represent weights or biases

Here are several examples of Standard deep learning representations

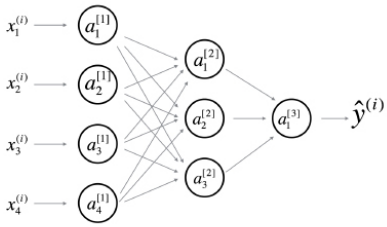


Figure 1: Comprehensive Network: representation commonly used for Neural Networks. For better aesthetic, we omitted the details on the parameters ($w_{ij}^{[l]}$ and $b_i^{[l]}$ etc...) that should appear on the edges

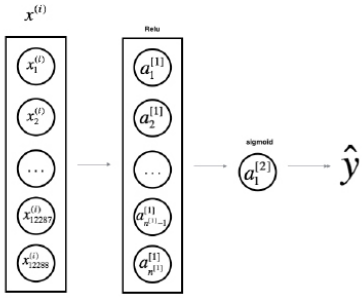


Figure 2: Simplified Network: a simpler representation of a two layer neural network, both are equivalent.

Deep Learning

Aka. Here comes the exciting Part

- **What is it:**
  Extract useful patterns from data.
- **How:**
  Neural network + optimization
- **How (Practical):**
  Python + TensorFlow & friends
- **Hard Part:**
  Good Questions + Good Data
- **Why now:**
  Data, hardware, community, tools, investment
- **Where do we stand?**
  Most big questions of intelligence have not been answered nor properly formulated

Exciting progress:

- Face recognition
- Image classification
- Speech recognition
- Text-to-speech generation
- Handwriting transcription
- Machine translation
- Medical diagnosis
- Cars: drivable area, lane keeping
- Digital assistants
- Ads, search, social recommendations
- Game playing with deep RL

Deep Learning Tools

1. CUDA
2. Theano
3. Caffe2
4. **Tensorflow 2.0**
5. **Pytorch 1.0**

What is Tensorflow?

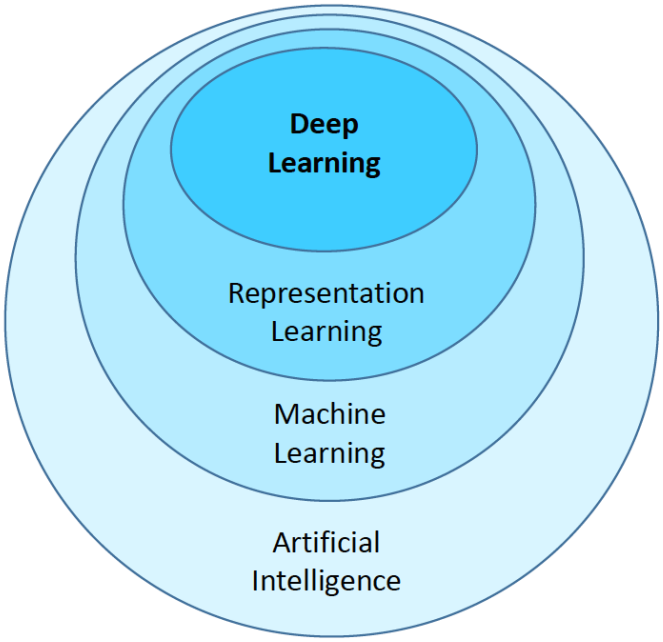
- **What is it:** Deep Learning Library *(and more)*
  - **Facts:** Open Source, Python, Google
- **Community:**
  - 117,000+ GitHub stars
  - TensorFlow.org: Blogs, Documentation, DevSummit, YouTube talks
- **Ecosystem:**
  - Keras: high-level API
  - TensorFlow.js: in the browser
  - TensorFlow Lite: on the phone
  - Colaboratory: in the cloud
  - TPU: optimized hardware
  - TensorBoard: visualization
  - TensorFlow Hub: graph modules
- **Alternatives**: PyTorch, MXNet, CNTK

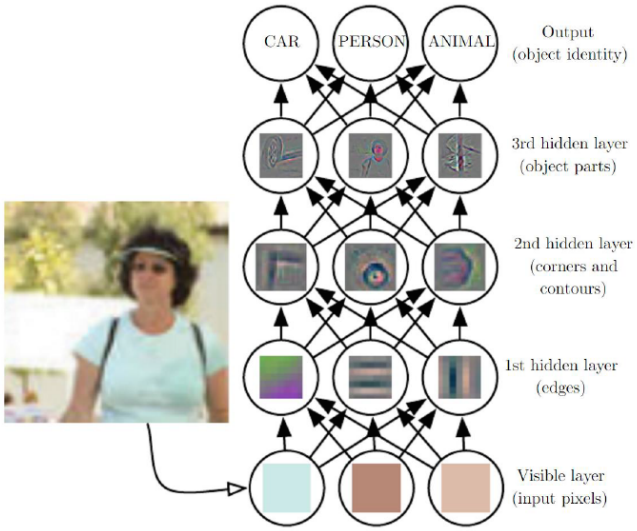
Extras:

- Swift for TensorFlow
- TensorFlow Serving
- TensorFlow Extended (TFX)
- TensorFlow Probability
- Tensor2Tensor

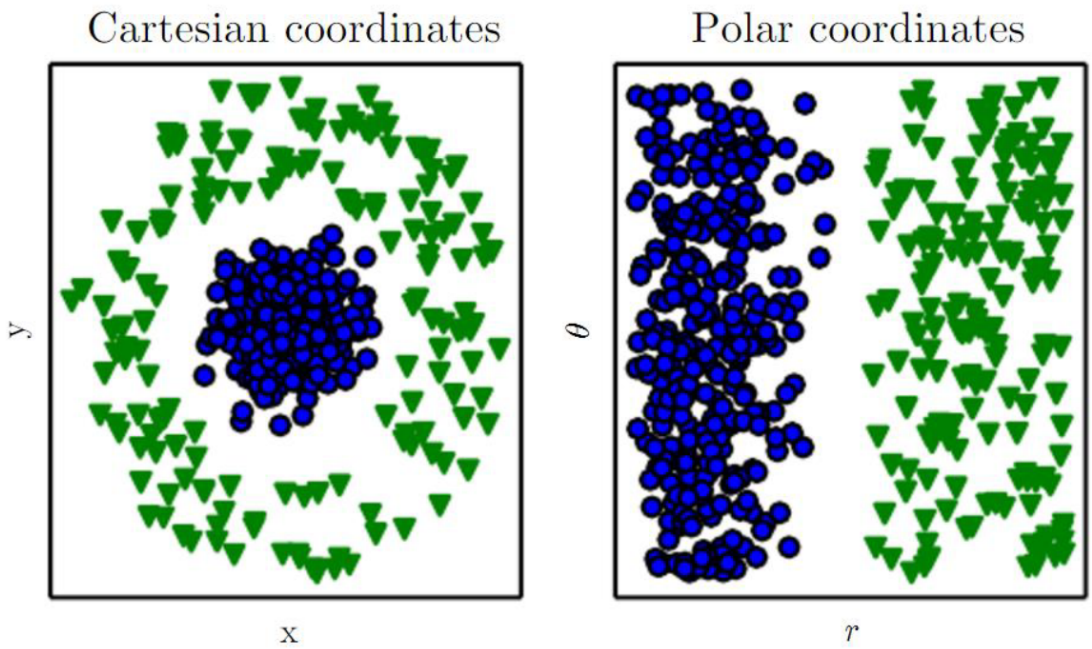
Deep Learning is Representation Learning

(aka Feature Learning)

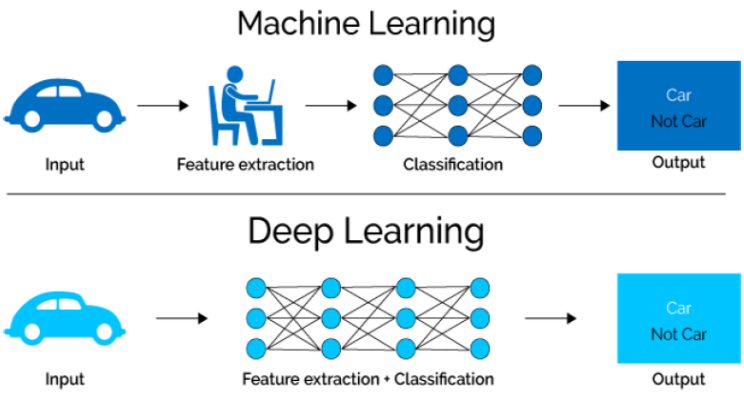

Deep
Learning
Representation
Learning
Machine
Learning
Artificial
Intelligence

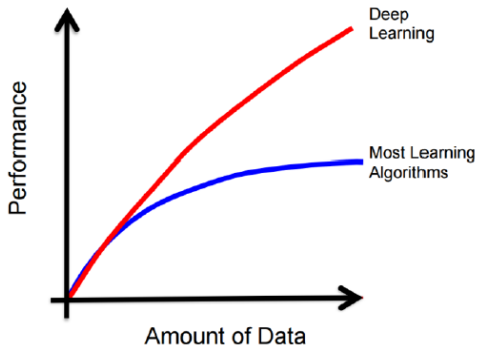

CAR
PERSON
ANIMAL
Output
(object identity)
3rd hidden layer
(object parts)
2nd hidden layer
(corners and
contours)
1st hidden layer
(edges)
Visible layer
(input pixels)

Representation Matters



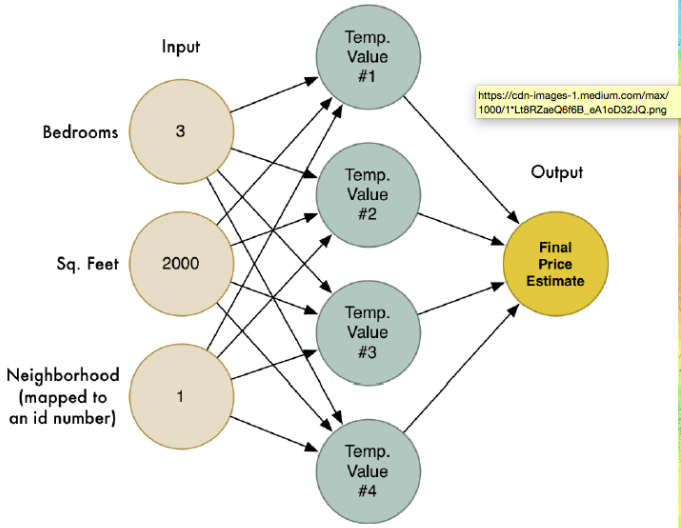
WHY DEEP LEARNING?



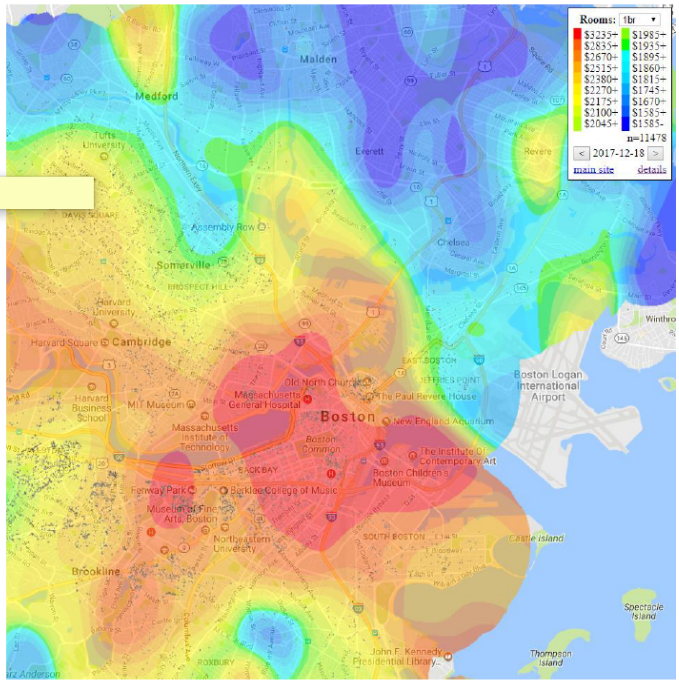


What is a Neural Network?

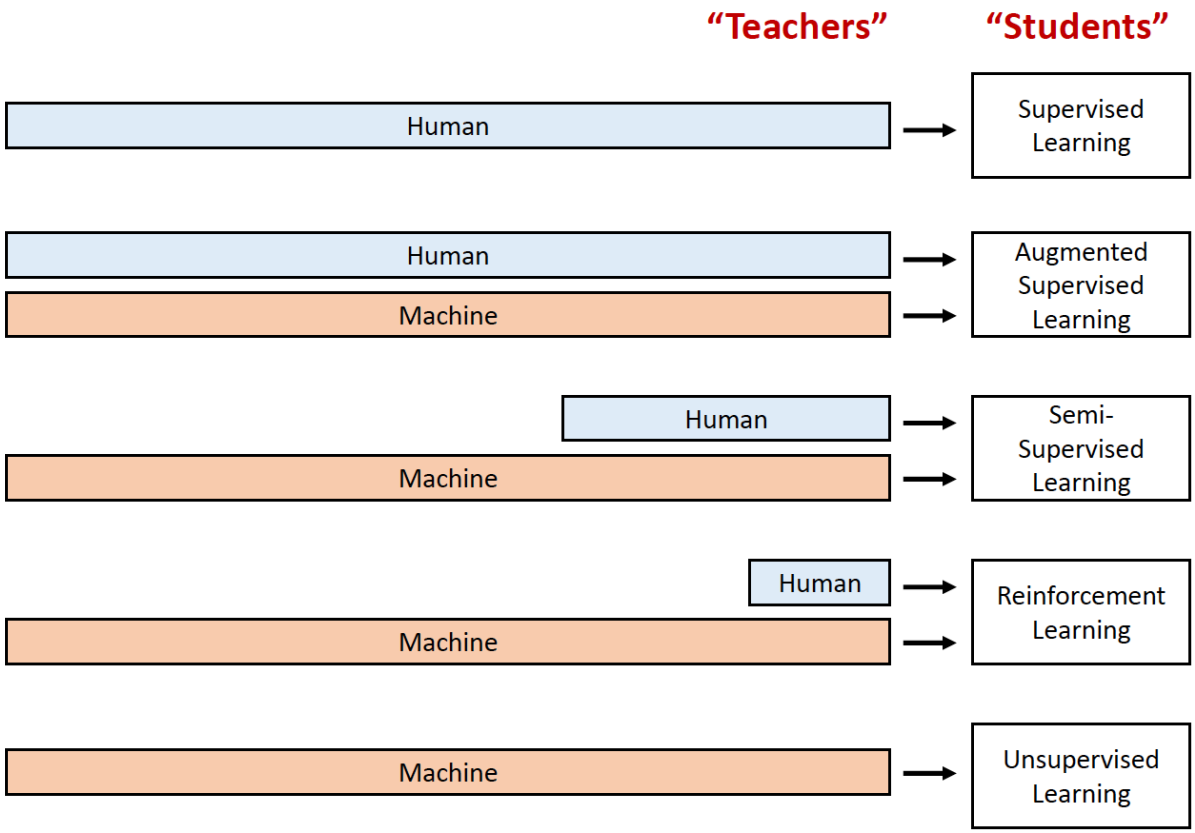
By the Example of House Price Prediction



https://cdn-images-1.medium.com/max/1000/1*Lt8RZaeQ6f6B_eA1oD32JQ.png



DeepLearning for human and machine



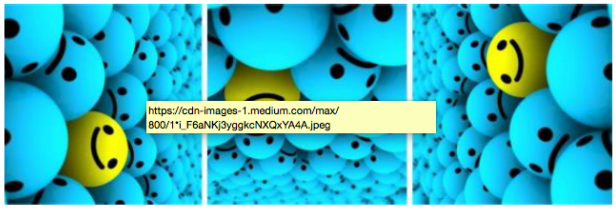
Data Augmentation

Crop:

Flip:

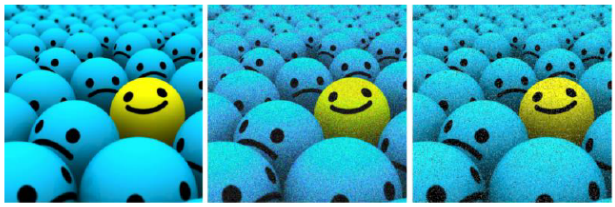
Scale:

Rotate:



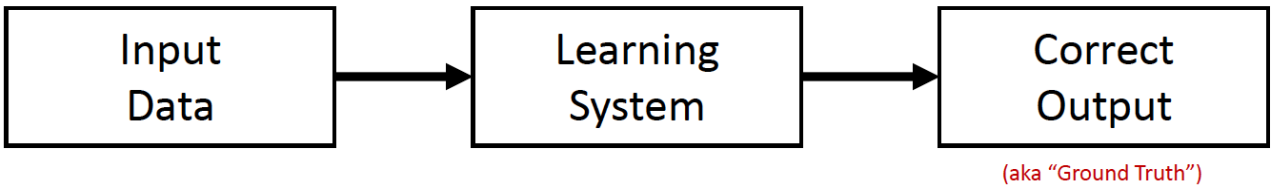
Translation:

Noise:

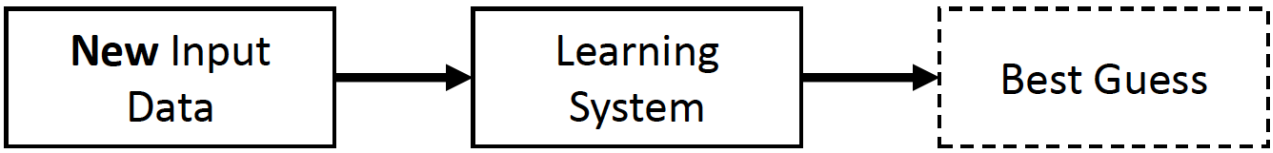


Deep Learning: Training and Testing

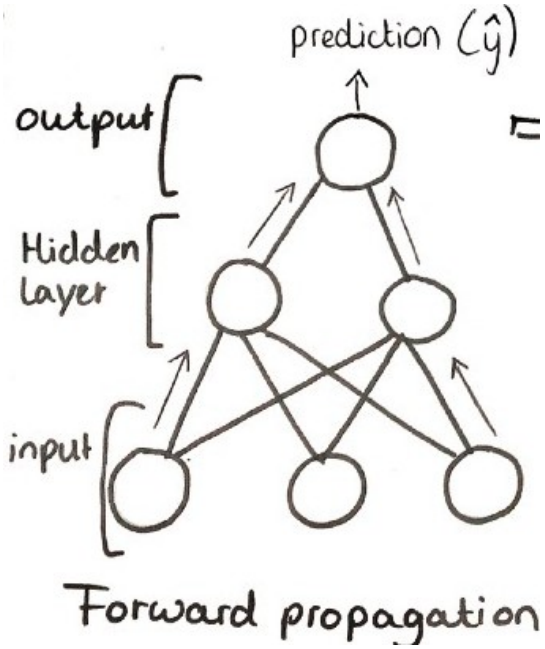
Training Stage:



Testing Stage:

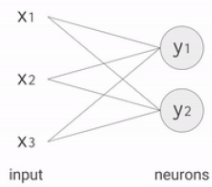


Neuron : Forward pass



Steps Involved in forward propagation!!

Step 1



Step 2

Step 3

$$y_1 = w_{11}x_1 + w_{12}x_2 + w_{13}x_3$$

Step 4

$$y_1 = f(w_{11}x_1 + w_{12}x_2 + w_{13}x_3)$$

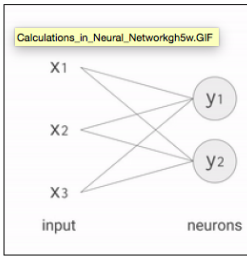
activation function

Step 5

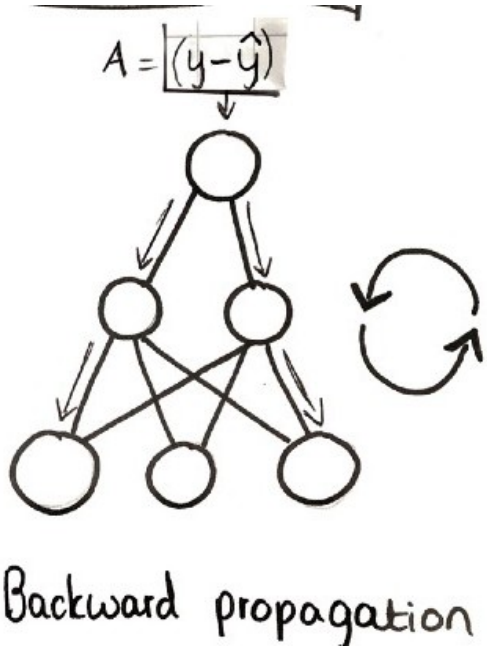
$$y_1 = f(w_{11}x_1 + w_{12}x_2 + w_{13}x_3)$$

$$y_2 = f(w_{21}x_1 + w_{22}x_2 + w_{23}x_3)$$

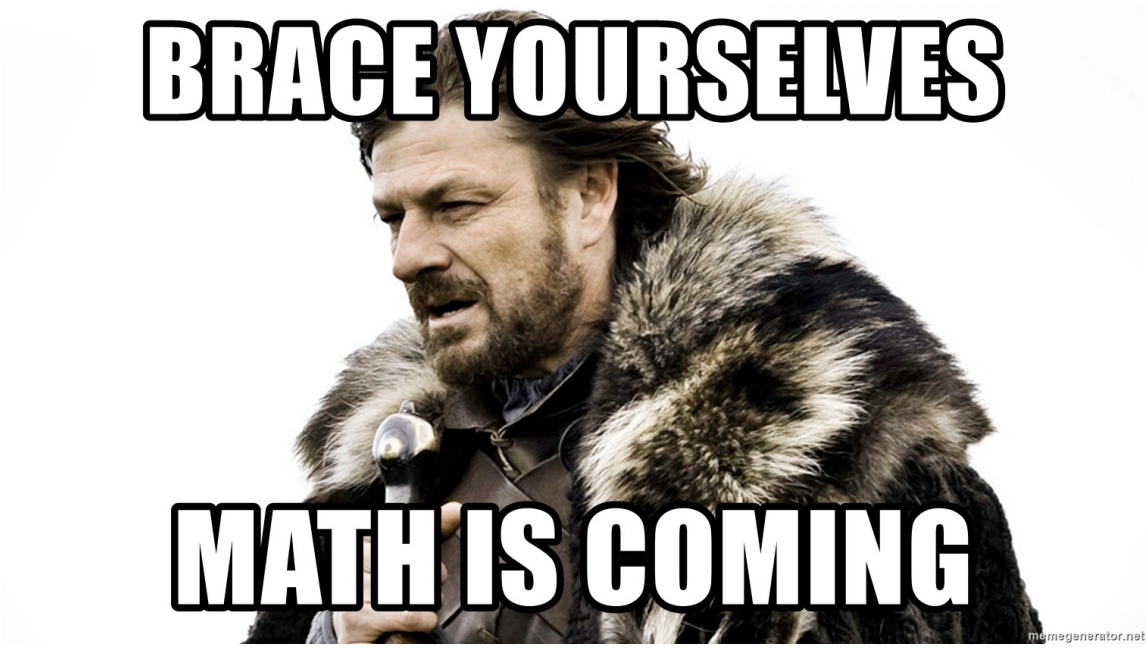
Animated



How Neural Networks Learns : BackPropagation of Errors



Steps Involved



1. **Input** x x**:** Set the corresponding activation a^1 a^1 for the input layer.

2. **Feedforward:** For each $l = 2, 3, \ldots, L$ $l = 2, 3, \ldots, L$ compute $z^l = w^l a^{l-1} + b^l$ $z^l = w^l a^{l-1} + b^l$ and $a^l = \sigma(z^l)$ $a^l = \sigma(z^l)$.

3. **Output error** δ^L δ^L**:** Compute the vector $\delta^L = \nabla_a C \odot \sigma'(z^L)$ $\delta^L = \nabla_a C \odot \sigma'(z^L)$.

4. **Backpropagate the error:** For each $l = L-1, L-2, \ldots, 2$ $l = L-1, L-2, \ldots, 2$ compute $\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l)$ $\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l)$.

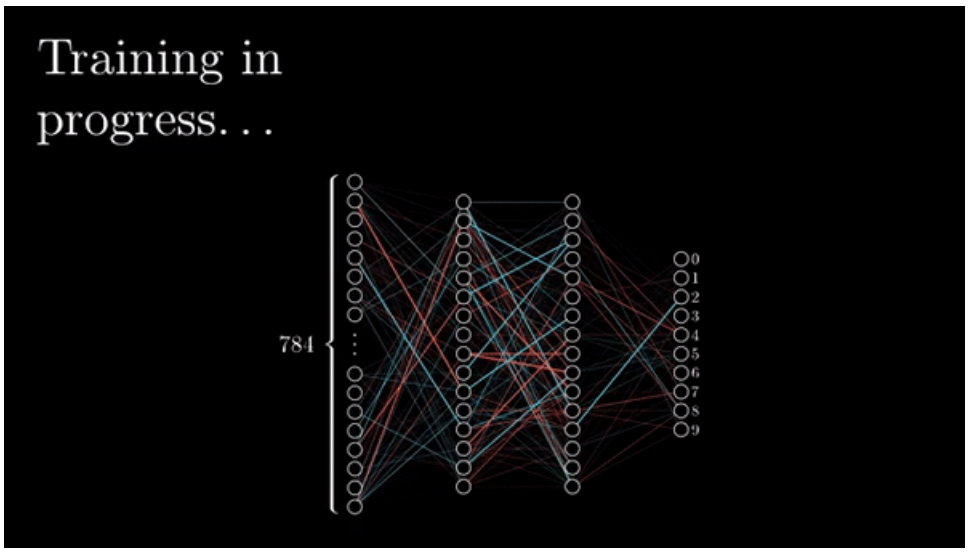
5. **Output:** The gradient of the cost function is given by $\frac{\partial C}{\partial w^l_{jk}} = a^{l-1}_k \delta^l_j$ $\frac{\partial C}{\partial w^l_{jk}} = a^{l-1}_k \delta^l_j$ and $\frac{\partial C}{\partial b^l_j} = \delta^l_j$ $\frac{\partial C}{\partial b^l_j} = \delta^l_j$.

Gradient descent

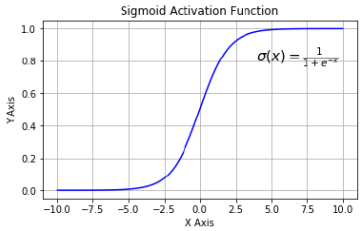
For each $l = L, L-1, \ldots, 2$ $l = L, L-1, \ldots, 2$
update the weights according to the rule $w^l \to w^l - \frac{\eta}{m} \sum_x \delta^{x,l} (a^{x,l-1})^T$ $w^l \to w^l - \frac{\eta}{m} \sum_x \delta^{x,l} (a^{x,l-1})^T$**, and the biases according to the rule** $b^l \to b^l - \frac{\eta}{m} \sum_x \delta^{x,l}$ $b^l \to b^l - \frac{\eta}{m} \sum_x \delta^{x,l}$**.**

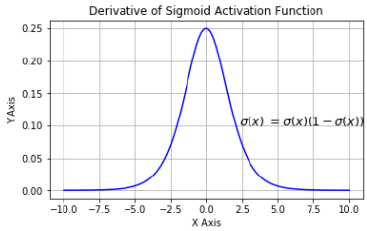
![optimization_problem.png](images/optimization_problem.png)

Backpropagation in action!



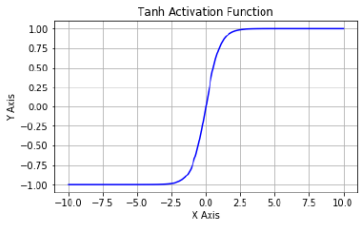
Different types of Activation function

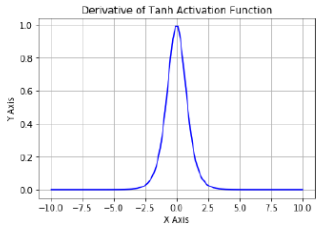




Sigmoid

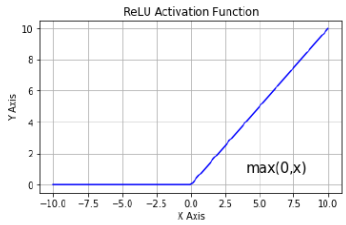
- Vanishing gradients
- Not zero centered

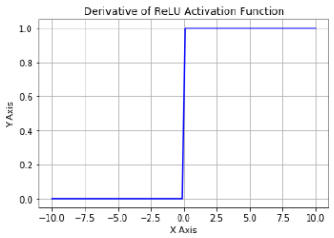




Tanh

- Vanishing gradients

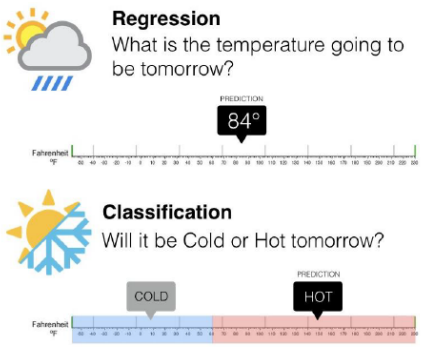




ReLU

- Not zero centered

Loss Function



- Loss function quantifies gap between prediction and ground truth
- For regression:
  - Mean Squared Error (MSE)
- For classification:
  - Cross Entropy Loss

Mean Squared Error

$$MSE = \frac{1}{N}\sum(t_i - s_i)^2$$

Prediction

Ground Truth

Cross Entropy Loss

$$CE = -\sum_{i}^{C} t_i log(s_i)$$

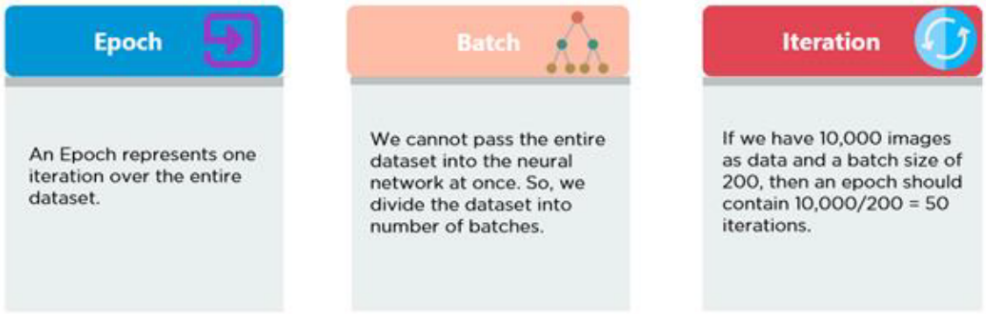
Classes

Prediction

Ground Truth {0,1}

==========

Mini-Batch Size



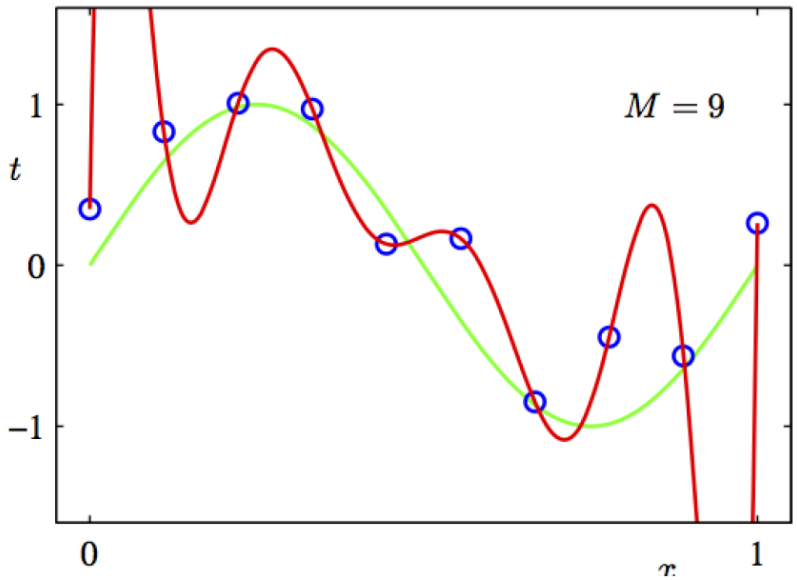
Mini-Batch size: Number of training instances the network evaluates per weight update step.

- Larger batch size = more computational speed
- Smaller batch size = (empirically) better generalization

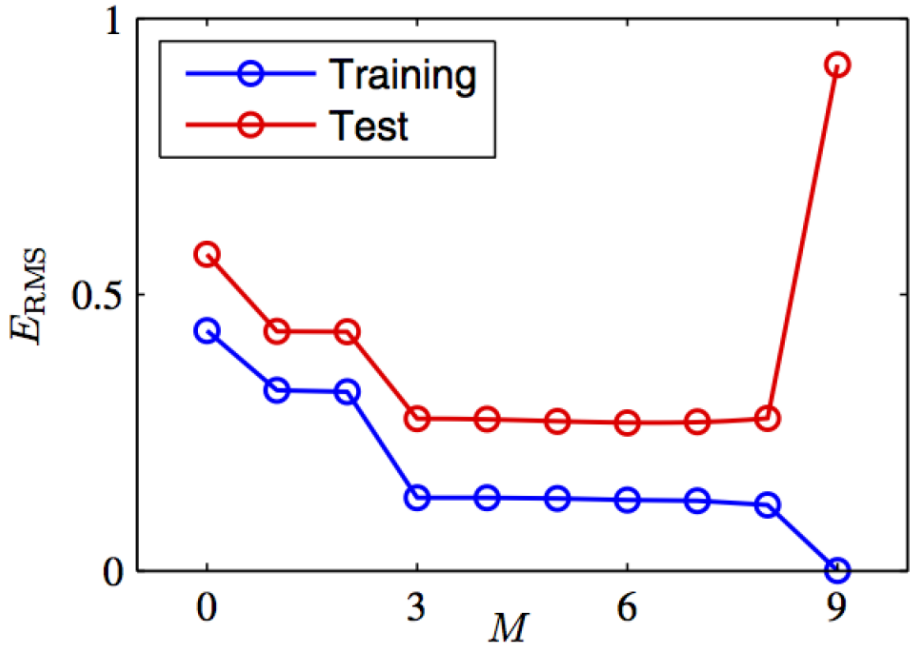
==========

Overfitting

- Help the network **generalize** to data it hasn't seen.
- Big problem for **small datasets**.
- Overfitting example (a sine curve vs 9-degree polynomial):

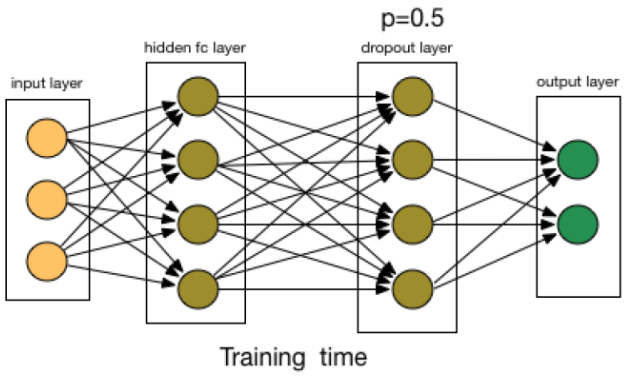


- Overfitting: The error decreases in the training set but increases in the test set.



==========

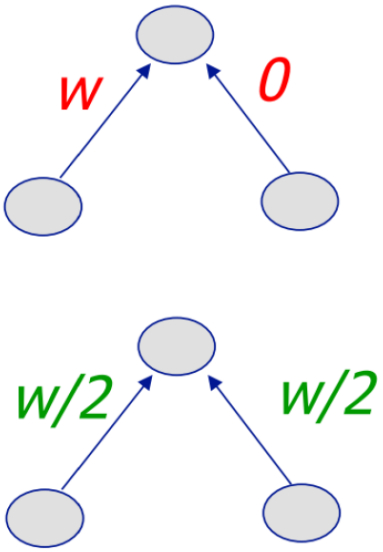
Regularization : 1. Dropout



- **Dropout:** Randomly remove some nodes in the network (along with incoming and outgoing edges)
- Notes:
  - Usually *p >= 0.5* (*p* is probability of keeping node)
  - Input layers *p* should be much higher (and use noise instead of dropout)
  - Most deep learning frameworks come with a dropout layer

2. Weight Penalty

Regularization: Weight Penalty *(aka Weight Decay)*



- **L2 Penalty:** Penalize squared weights. Result:
  - Keeps weight small unless error derivative is very large.
  - Prevent from fitting sampling error.
  - Smoother model (output changes slower as the input change).
  - If network has two similar inputs, it prefers to put half the weight on each rather than all the weight on one.
- **L1 Penalty:** Penalize absolute weights. Result:
  - Allow for a few weights to remain large.

==========

Normalization

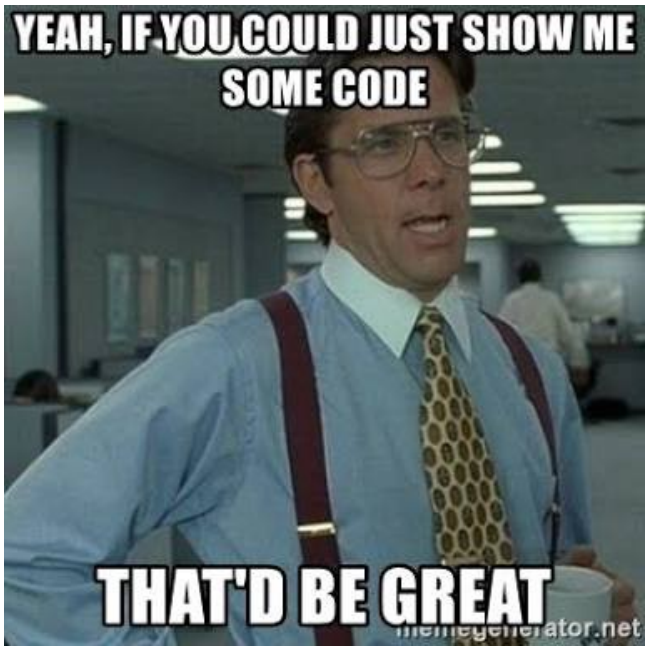
- Network Input Normalization
  - *Example:* Pixel to [0, 1] or [-1, 1] or according to mean and std.

- Batch Normalization (BatchNorm, BN)
  - Normalize hidden layer inputs to mini-batch mean & variance
  - Reduces impact of earlier layers on later layers

- Batch Renormalization (BatchRenorm, BR)
  - Fixes difference b/w training and inference by keeping a moving average asymptotically approaching a global normalization.

- Other options:
  - Layer normalization (LN) – conceived for RNNs
  - Instance normalization (IN) – conceived for Style Transfer
  - Group normalization (GN) – conceived for CNNs

Okay Enough!



Let's Code a neural Network from scratch

In [3]:
```
import numpy as np
#Define the Activation Function Sigmoid and its differential

def sigmoid(t):
    return 1/(1+np.exp(-t))

def difsigmoid(d):
    return d*(1-d)
```

In [91]:
```
#Define a Neural Network Class
class NN:
    def __init__(self,x,y):
        self.input= x
        self.w1 = np.random.rand(self.input.shape[1],4)
        self.w2 = np.random.rand(4,1)
        self.y = y
        self.output = np.zeros(y.shape)

    #Define the forward Pass
    def ff(self):
        self.l1 = sigmoid(np.dot(self.input,self.w1))
        self.l2 = sigmoid(np.dot(self.l1, self.w2))
        return self.l2

    #Define the Backward pass to backpropagate the errors
    def backprop(self):
        d_w2 = np.dot(self.l1.T,2*(self.y-self.output)*difsigmoid(self.out
put))
        d_w1 = np.dot(self.input.T, np.dot(2*(self.y -self.output)*difsigm
oid(self.output), self.w2.T)*difsigmoid(self.l1))
        self.w1 += d_w1
        self.w2 += d_w2
    #Define the training step
    def train(self,X,y):
        self.output = self.ff()
        self.backprop()
```

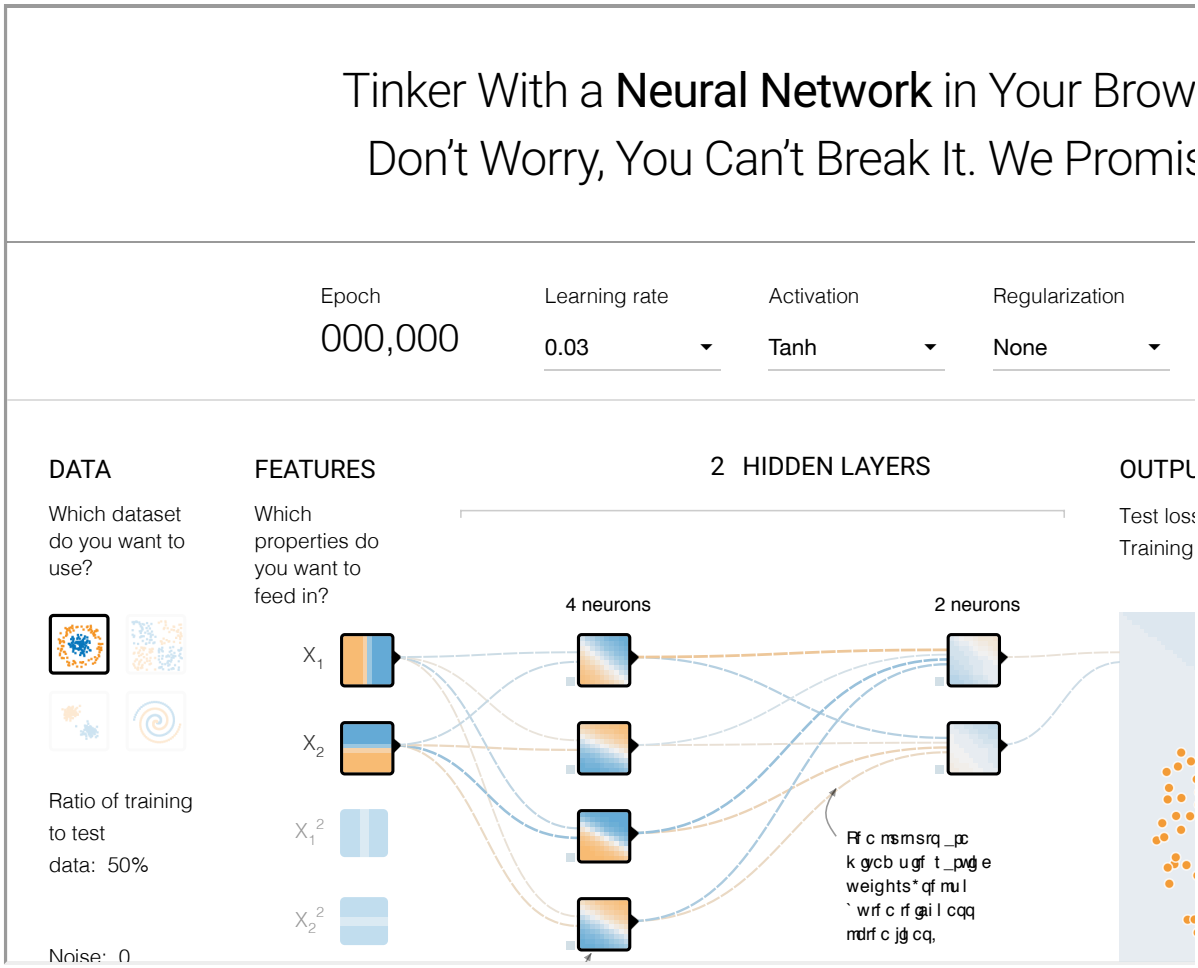
In [92]:
```
X=np.array(([0,0,1],[0,1,1],[1,0,1],[1,1,1]), dtype=float)
y=np.array(([0],[1],[1],[0]), dtype=float)
#Instantiate a neural network from our created Class.
NN = NN(X,y)
#Set our training Epochs
epochs = 2000

for i in range(epochs):
    if i % 1000 ==0:
        print ("for iteration # " + str(i) + "\n")
        l = np.mean(np.square(y - NN.ff()))
        print ("The Loss is: \n" + str(l)) # MSE LOSS
        print ("\n")
    NN.train(X,y)
l = np.around(l*100,decimals=6)
print(f'Final Training Loss is {l} %')
```

```
for iteration # 0

The Loss is:
0.3246295143337549


for iteration # 1000

The Loss is:
0.0012707744468423224


Final Training Loss is 0.127077 %
```

Neural Network Playground (http://playground.tensorflow.org/)

Now its your time to tinker around with Neural Networks

In [1]:
```
%%html
<iframe src="http://playground.tensorflow.org/" width="1000" height="600">
</iframe>
```



In [ ]:
```
```

That was a lot!

Here are some resources to aid you.

1. ### But what is a Neural Network? | Deep learning by 3Blue1Brown (https://www.youtube.com/watch?v=aircAruvnKk)
2. ### What is backpropagation really doing? | Deep learning by 3Blue1Brown (https://www.youtube.com/watch?v=Ilg3gGewQ5U)
3. ### How the backpropagation algorithm works (http://neuralnetworksanddeeplearning.com/chap2.html)
4. ### Using neural nets to recognize handwritten digits (http://neuralnetworksanddeeplearning.com/chap1.html)