What is Deep Learning

Tuesday, December 17, 2024 12:33 PM

For neural networks to work properly you need two things

- Data
- Processing Power

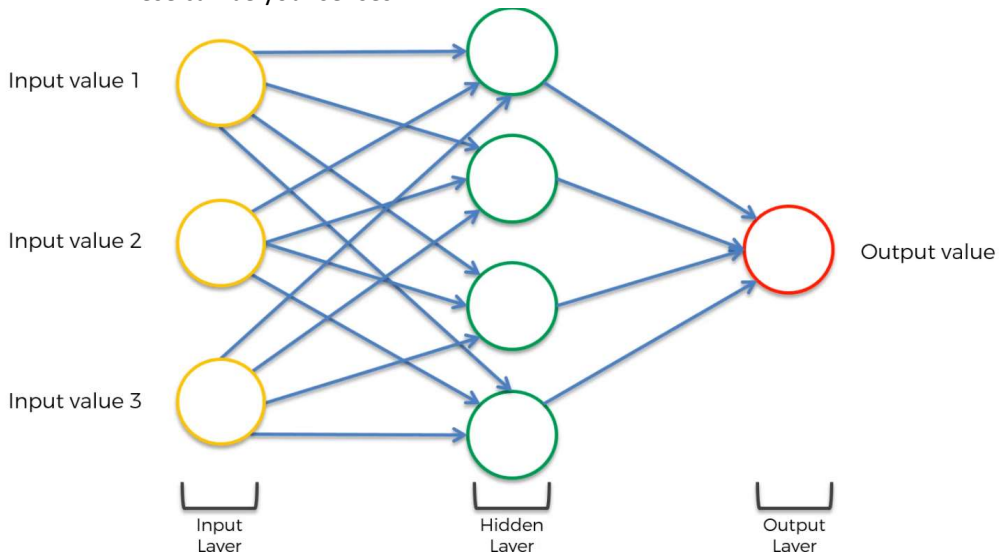
https://www.superdatascience.com/blogs/the-ultimate-guide-to-convolutional-neural-networks-cnn

What is Deep Learning

- We are trying to mimic how the human brain operates, and try and recreate it

How do we recreate this

- We create an artificial network with nodes, or neurons
- We have the input layer
- We also have an output layer
- In between we have a hidden layer
    - These can be your senses



Shallow/Deep Learning

- Shallow learning is when we only have a few hidden layers
- Deep learning is when you have many hidden layers

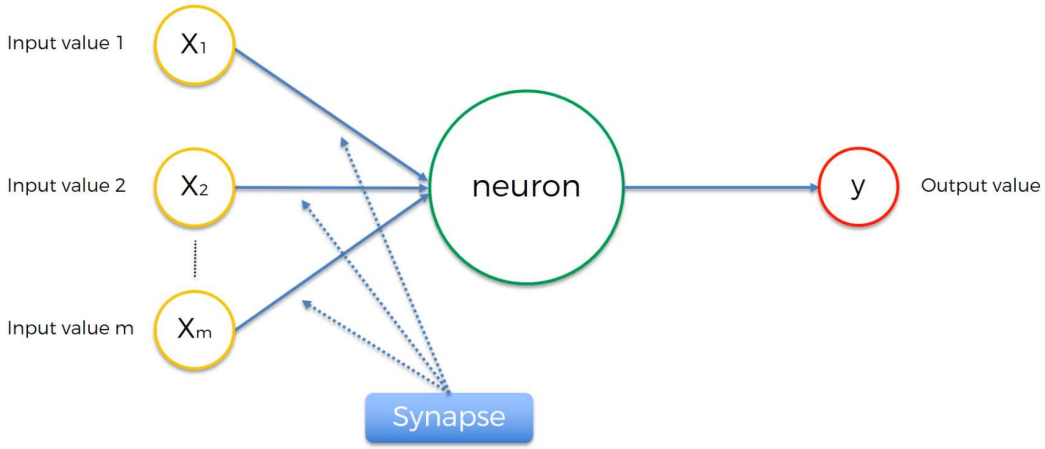
Artificial Neural Networks

Tuesday, December 17, 2024 7:45 PM

We want to mimic how the human brain works

Neuron (node)

- Has input values and an output value
- Yellow = Input layer
- Green = Neuron
- Red = Output Value



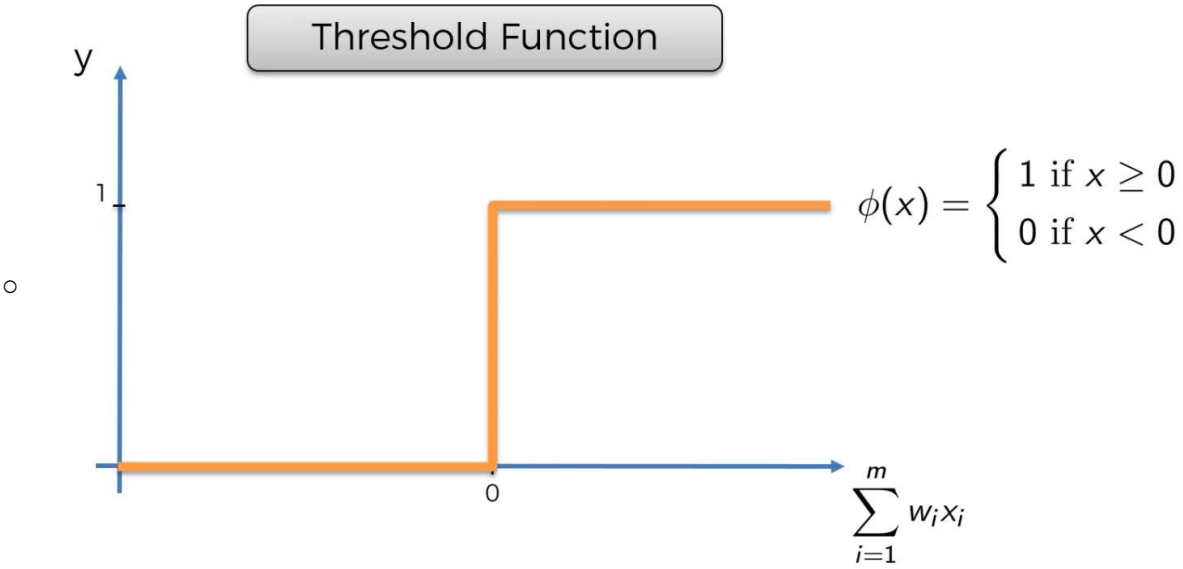
- Inputs are Independent Variables
  - They are all for one single observation
  - We need to standardize or normalize them
- Output Value
  - Continuous (Price)
  - Binary (Yes / No)
  - Categorical

- Synapse
  - They each have weights
  - By adjusting the weights

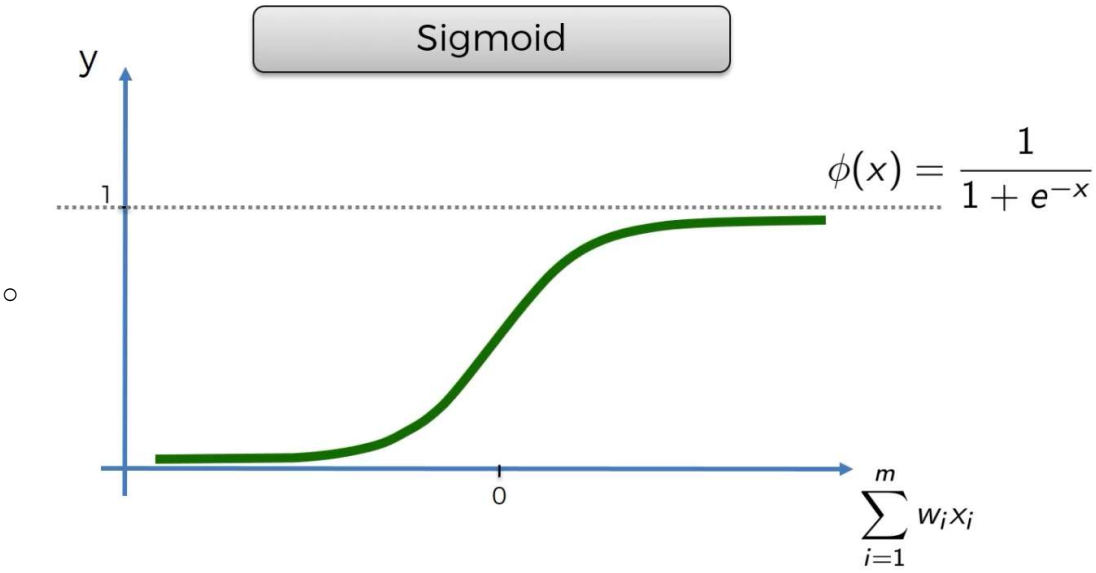
- Neuron
  - Gets the weighted sum of all input values
  - Activation Function applied to weighted sum
    - Either pass the signal or wont pass the signal

Activation Functions

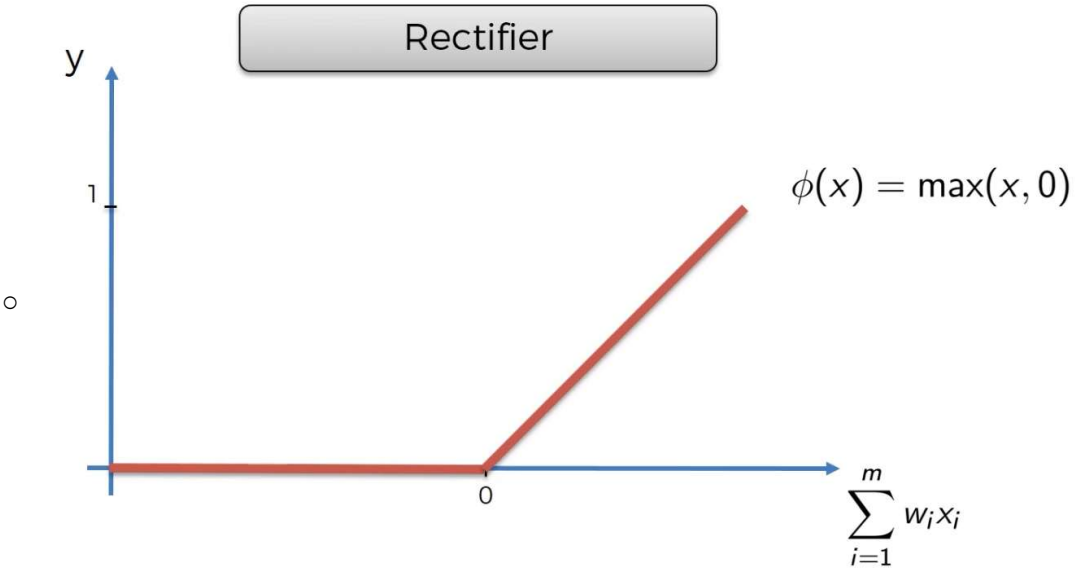
- Threshold Funciton
  - X axis there are the weighted sum
  - Y axis is 0-1
  - If the value is less then 0, then it passes 0
  - If the value is more than 0, then it passes 1



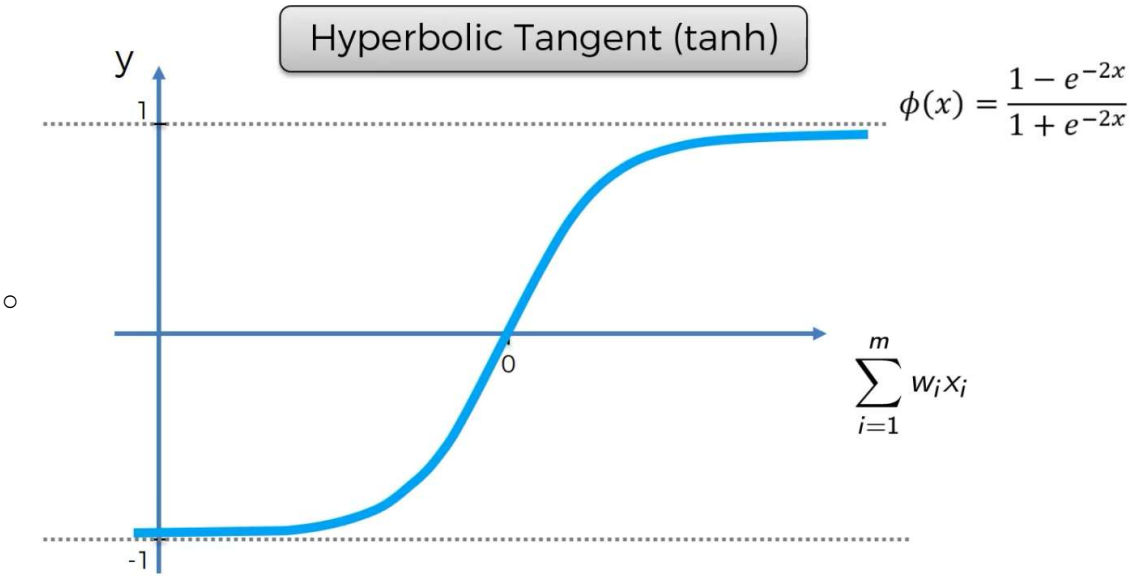
- Sigmoid Function
  - Used in logistic regression. The function is smooth
  - Used in the output layer, especially trying to predict probability



- Rectifier
  - One of the most popular functions

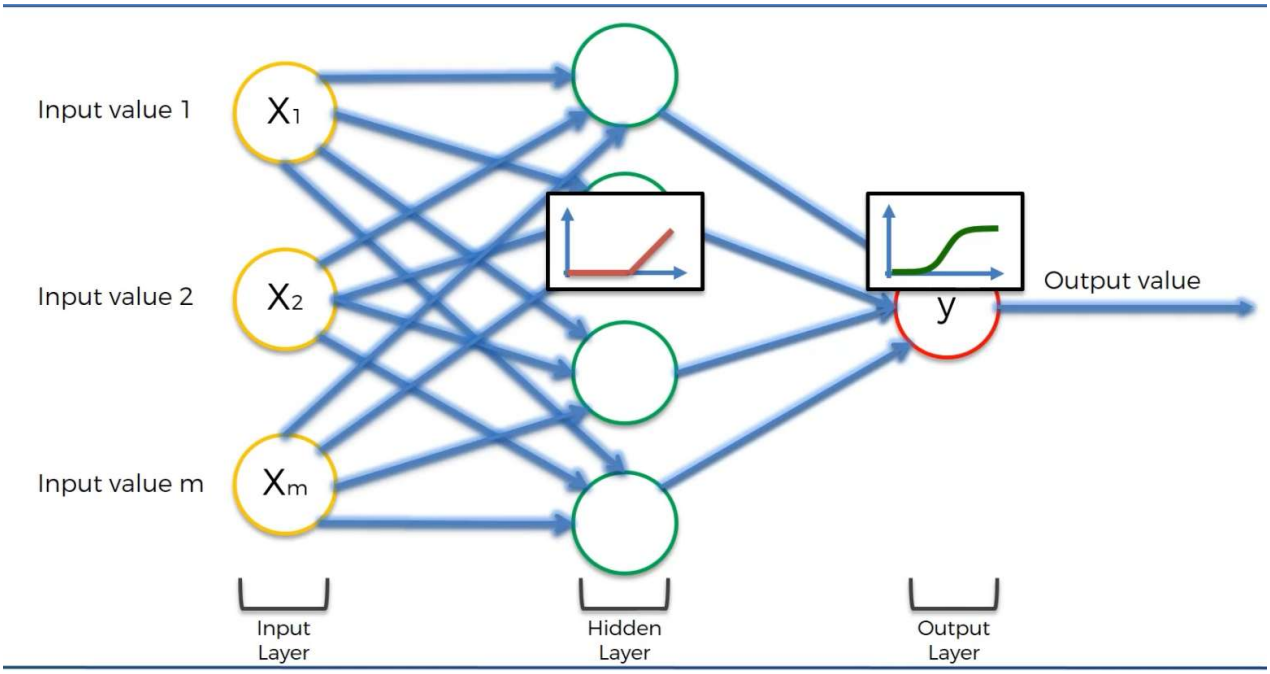


- Hyperbolic Tangent (tanh)
  - Similar to sigmoid, but goes from -1 to 1



Assuming the Dependent variable is binary, what threshold functions could you use?

- Threshold
- Sigmoid (probability of y = 1)

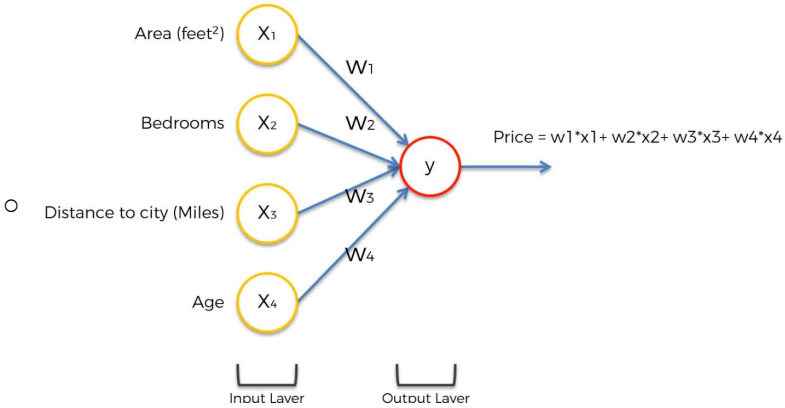


- Hidden layer we apply the rectifier function
- Output layer we apply the sigmoid function

How do NNs work?

- For this example we are going to use property values

- 4 parameters about the property
  - Area, Bedrooms, Distance to City, Age
- Very simple case



  - Even without hidden layers, we already have an interpolation that could work for other machine learning models

- Adding hidden layer gives us extra power
  - We can pass all input layer neurons into the first hidden layer
    - The hidden layer neuron can only fire up when certain criteria is made
  - Certain hidden layer neurons can only consider certain input values

- For example, the three parameters turn into one new parameter if the activation function is fired up

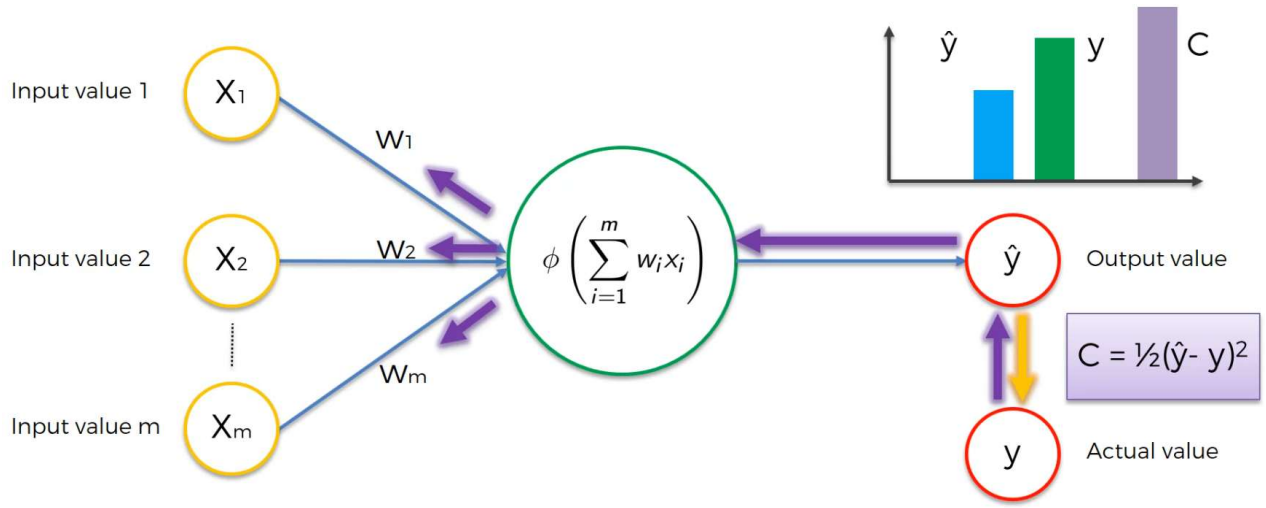
- Having more neurons allows the NN to look for very specific things in combination
  - A single ant cannot build an ant hill, but multiple can

How do NNs Learn?

- Given a file of cats and dogs, how to determine which pictures are cats and which are dogs
  - The NN will take the files of cats and dogs and learn what each look like

Cost Function / **Back Propagation**

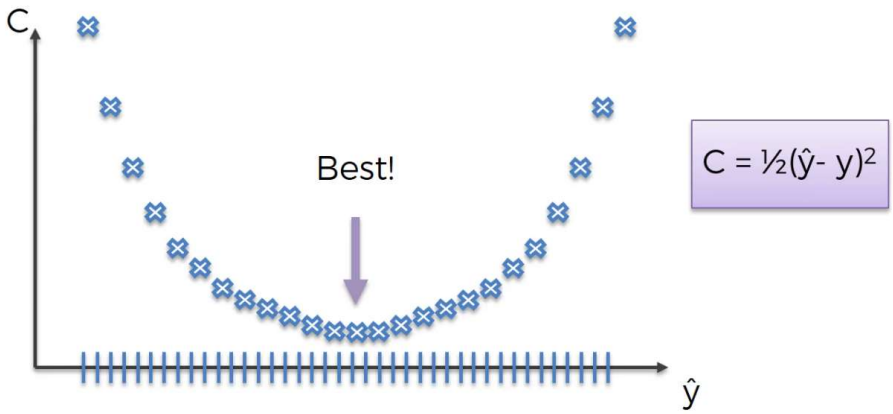
- Basically calculates the error between y^ and y (predicted value and actual value)
- The cost function then travels backward across the neural network and updates the weights on the input values
  - Goal is to minimize the cost function



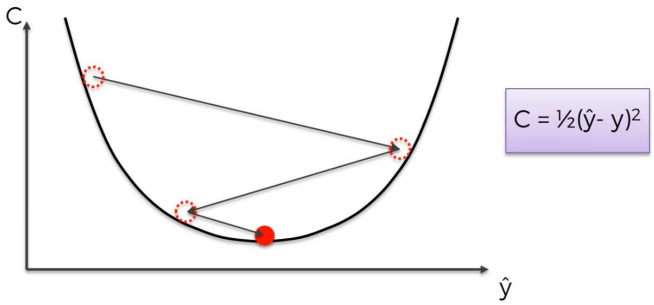
- Training on one row, we want to get the weights just right so the cost function is minimized
- Training with multiple rows, you get y^ for each row, Now we can calculate the cost function from all of the y^. Using this cost function we can update the weights

Gradient Descent

- In order for a neural network to learn, back propagation needs to occur
- Brute Force method



    - Curse of Dimensionality
        - The more dimensions to calculate, the higher computation time
    - Brute force is too slow
- Gradient Descent
    - Given a cost function, look at the slope at the specific point
    - If needed, roll the ball down, and opposite if we need to roll the ball up



Stochastic Gradient Descent
- If the data wasn't a parabola, then this is a solution
- Adjust the weight row by row
    - Avoids finding local minimum

Back propagation
- Used to train the model and adjust the weights
- We are able to adjust all the weights at the same time

Training the ANN with Stochastic Graident Descent
- STEP 1: Randomly initialize the weights to small numbers close to 0
- STEP 2: Input the first observation of your dataset in the input lear, each feature in one input node
- STEP 3: Forward-Propagation: from left to right, the neurons are activated in a way that the impact of each neuron's activation is limited by the weights. Propagate the activations until getting the predicted result y.
- STEP 4: Compare the predicted result to the actual result. Measure the generated error
- STEP 5: Back-Propagation: From right to left, the error is back-propagated. Update the weights

according to how much they are responsible for the error. The learning rate decides by how much we update the weights

- STEP 6: Repeat steps 1 to 5 and update the weights after each observation (Reinforcement Learning), or Repeat Steps 1 to 5 but update the weights only after a batch of observations (Batch Learning)
- STEP 7: When the whole training set passed through the ANN, that makes an epoch. Redo more epochs

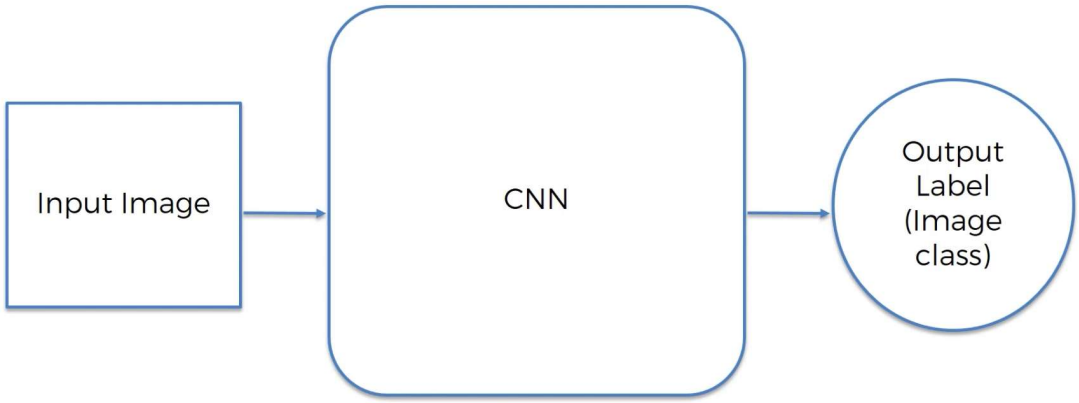
ALWAYS APPLY FEATURE SCALING WHILE BUILDING ARTIFICIAL NEURAL NETWORKS

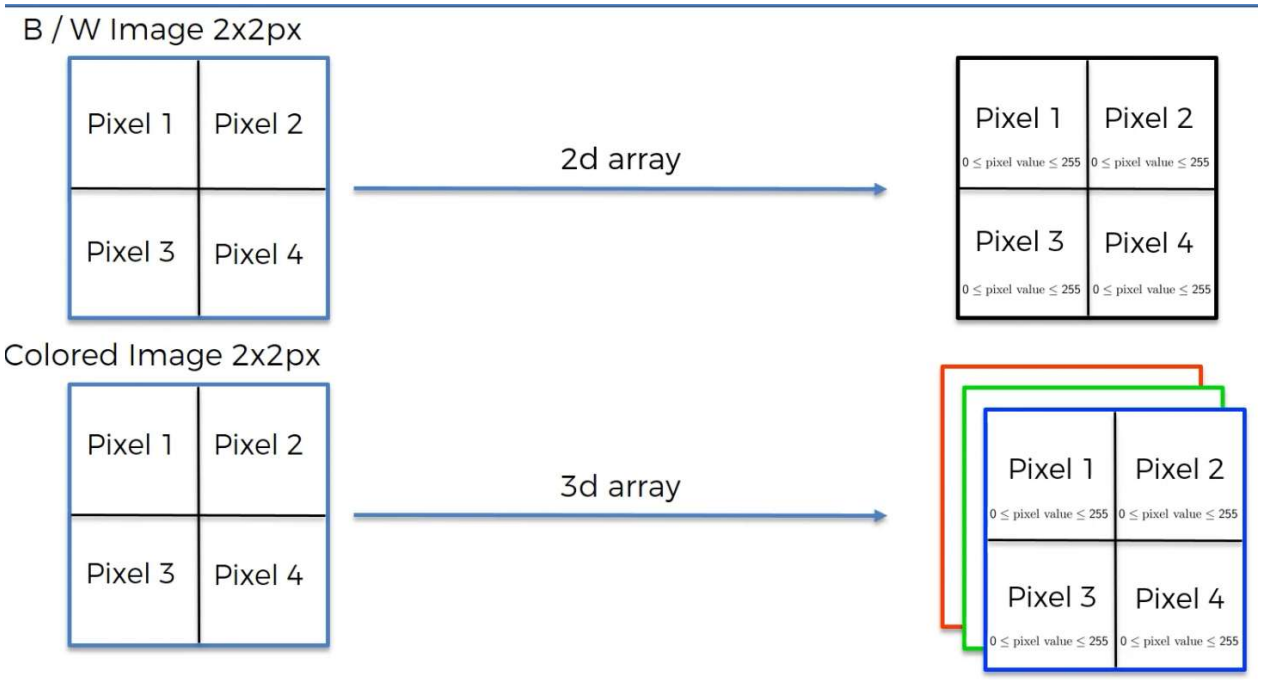
https://www.superdatascience.com/blogs/the-ultimate-guide-to-artificial-neural-networks-ann

Convolutional Neural Networks

Thursday, December 26, 2024 3:36 PM

The way computers process images is very similar how humans process images

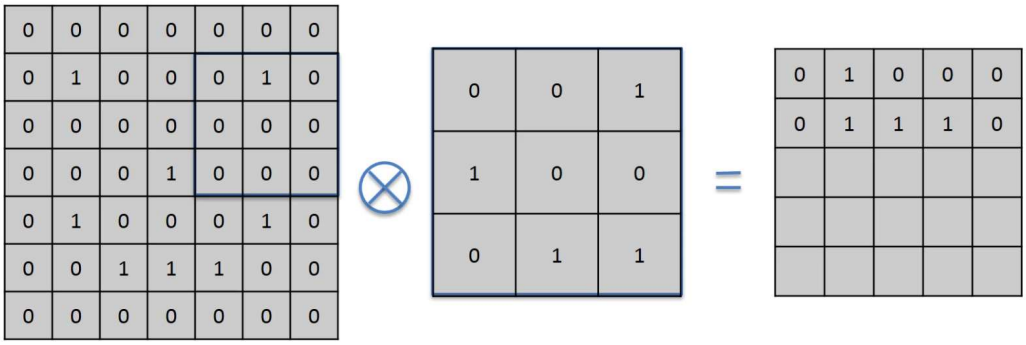




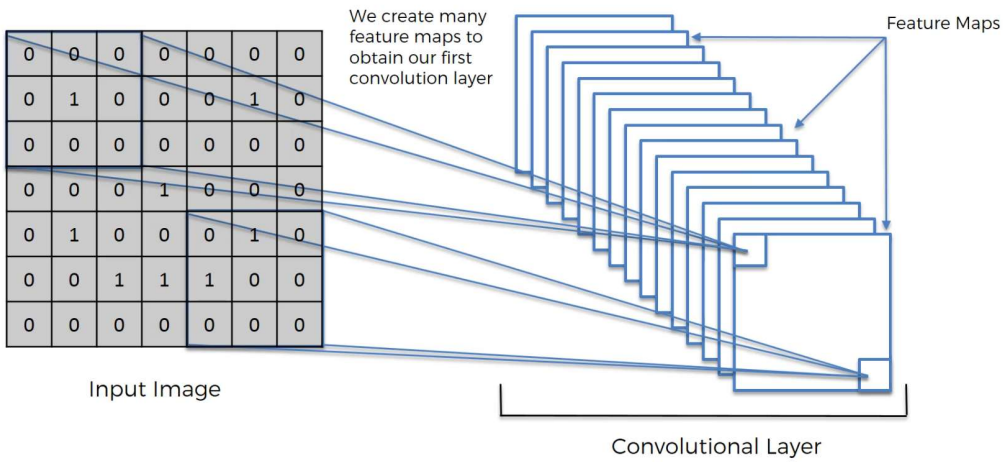
STEP 1: Convolution
STEP 2: Max Pooling
STEP 3: Flattening
STEP 4: Full Connection

STEP 1: Convolution

- $(f * g)(t) \stackrel{\text{def}}{=} \int_{-\infty}^{\infty} f(\tau)\, g(t-\tau)\, d\tau$
- Convolution Function

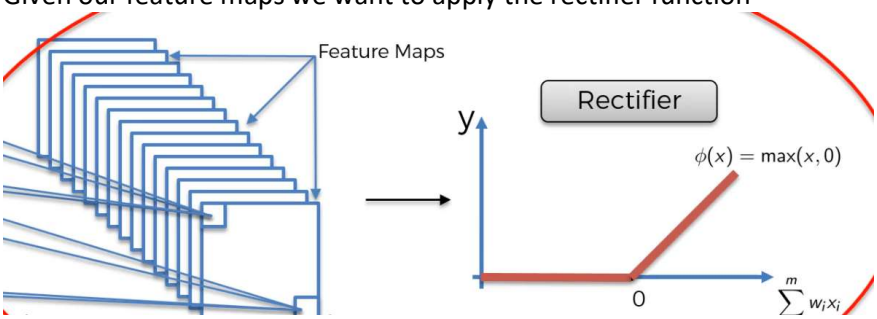
- Feature Detector
  - Mainly 3x3, but can be different
  - Using the feature detector go through the entire input image and AND the values
  - 

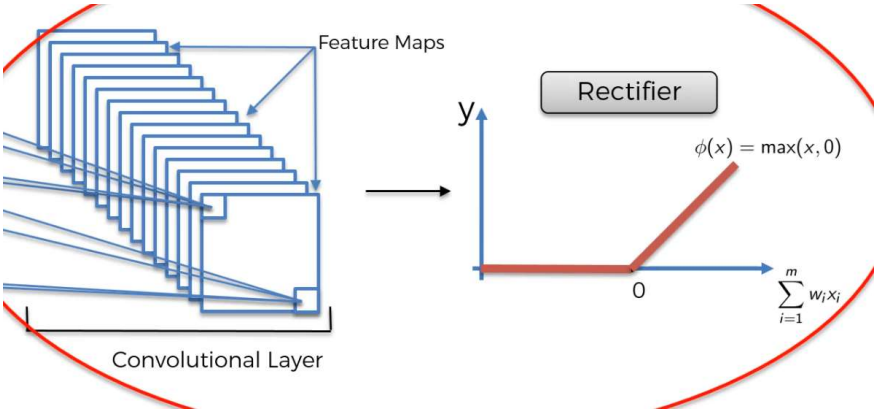
  - This generates a feature map
    - Also can be called an activation map
  - The highest value that can exist in the feature map is the amount of 1s in the feature detector

- Feature Maps
  - We create multiple feature maps because we search for multiple features
  - This is to obtain our first convoluted layer
  - 

- Primary purpose of convolution is to find features within the image, and put them into a feature map
- Feature map maintains the spatial relationships between pixels

STEP 1 (b) - ReLU Layer (Rectified Linear Unit)

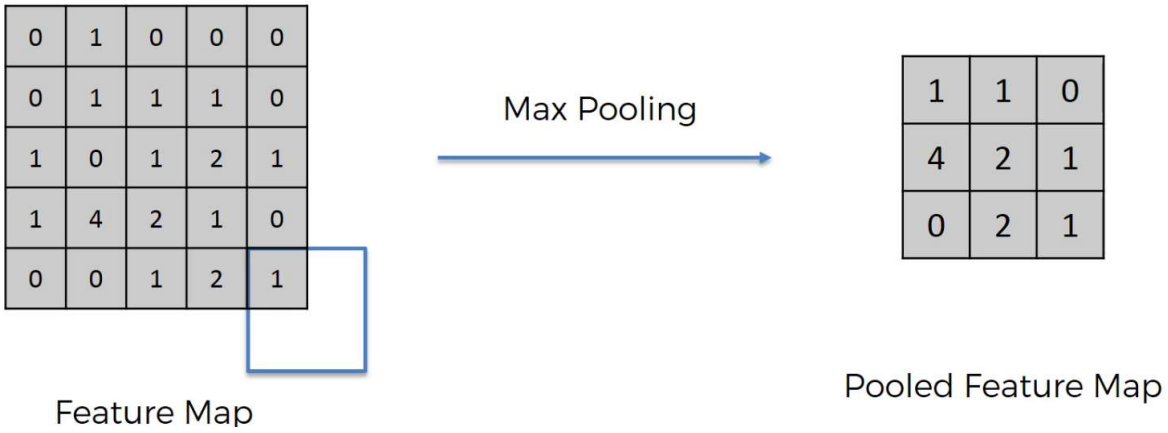
- Given our feature maps we want to apply the rectifier function
- 

- 

- Basically this removes all the black from the feature maps
    - This breaks up the linearity of the image

STEP 2: Max Pooling

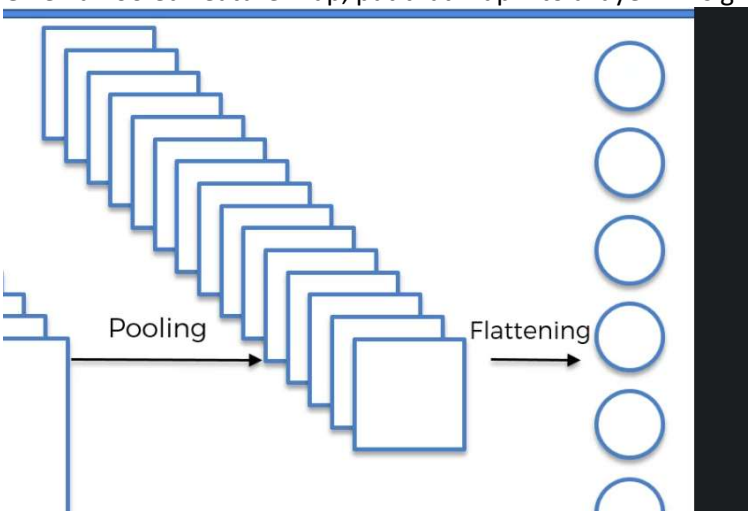
- Given different images of the same thing, how can we get the neural network to recognize the same object
- Given a feature map, we can apply pooling (we will do max pooling)
- We create a Pooled Feature Map.
    - This is a smaller feature map, but just cut out the irrelevant indexes

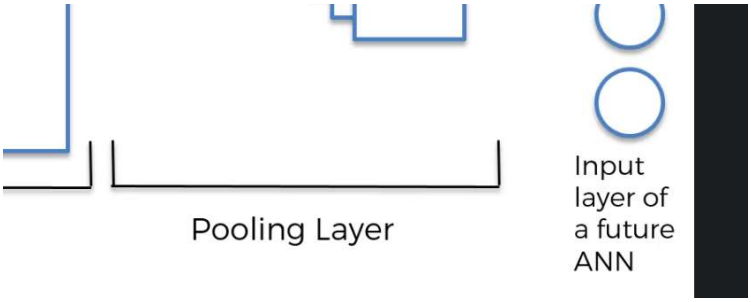
- 

- Removing information (pooling) ensures overfitting wouldn't happen

STEP 3 - Flattening

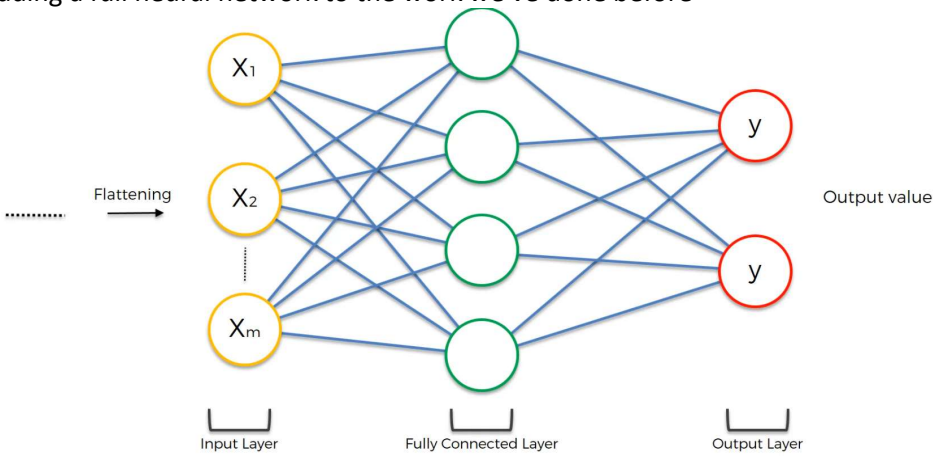
- Given a Pooled Feature Map, put that map into a layer. This gives us a large vector layer

- 

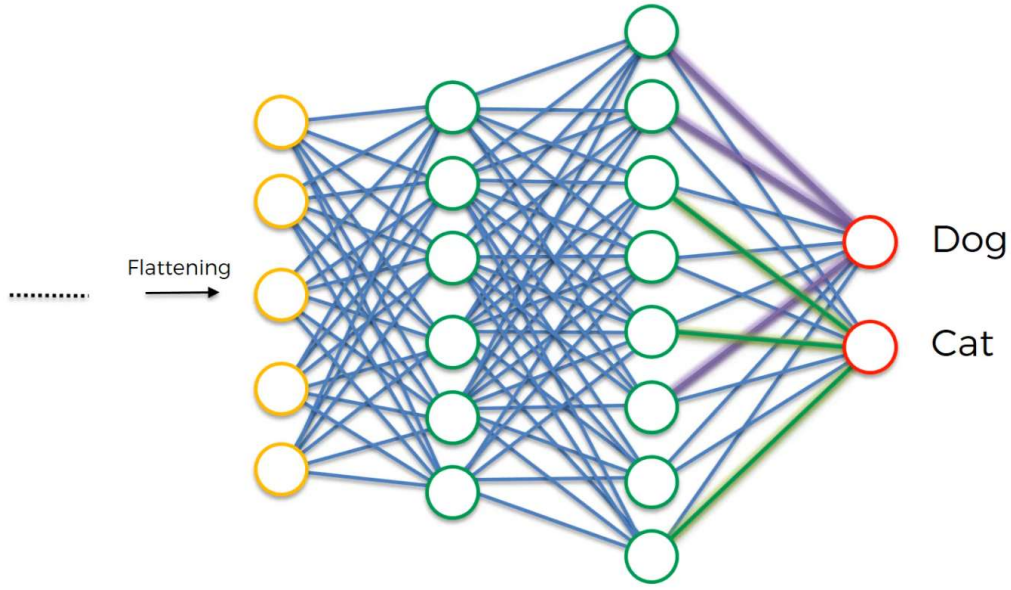


STEP 4: Full Connection

- Adding a full neural network to the work we've done before
-



- When you are doing classification you have multiple output layers
- When an output is incorrect, instead of cost function, it is called a cross entropy function
    - Optimizing the weights, and back propagated throughout the network
    - The features matrices are also adjusted to improve results
-



- The Dog or Cat will pay more attention to those features that have a higher value

Summary

- Started with an input image, which we apply multiple layers to the image
    - Convolutional Layer
    - Pooling Layer
        - Make sure we have spatial invariance in our image. Also reduces the size of images
    - Flattening

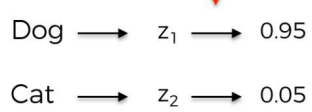
        - Flatten into one long vector of values
    - Fully connected neural networks

Softmax & Cross-Entropy

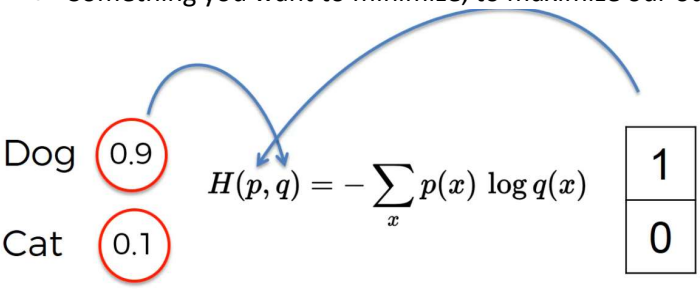
- The way the outputs of the neural network add up to 1, or even has the highest value is Softmax equation

$$f_j(z) = \frac{e^{z_j}}{\sum_k e^{z_k}}$$

-



- A cross-entropy function is a better cost function
    - Something you want to minimize, to maximize our output



- Classification Error
    - Measures the error rate (if you got 2 out of 3 correct, it is a 1/3 error rate)
- Mean Squared Error
    - Sum of squared errors
    - Better for regression
- Cross Entropy
    - Just seems more accurate than Mean Squared Error
    - Only for classification