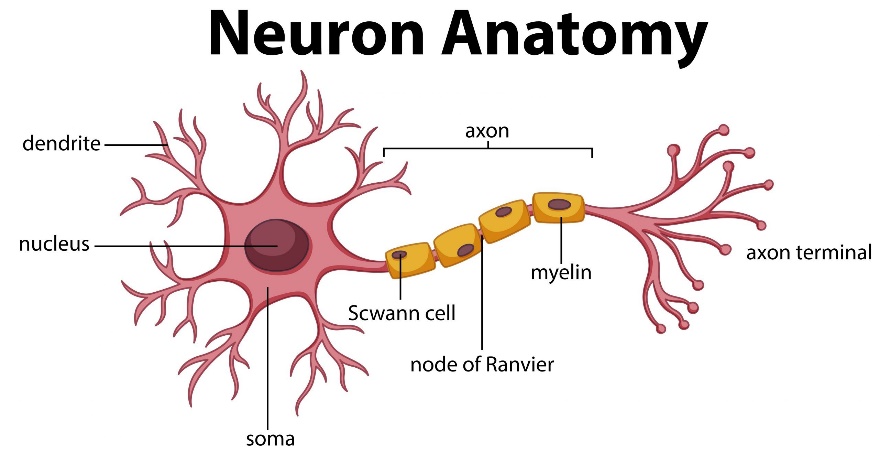
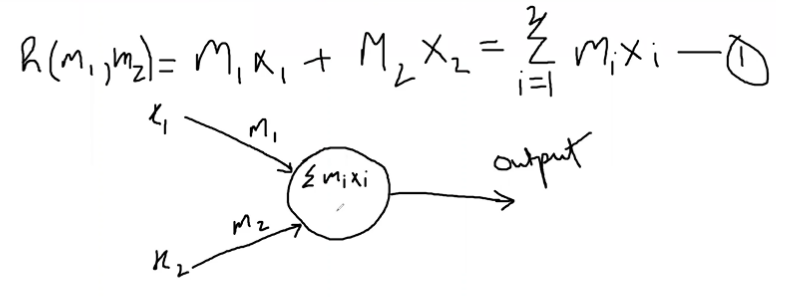
A neuron is a single unit in brain that does some computations and passes result to other neurons. Dendrite and axon terminals are connections to other neurons. It receives initial electrical signal from say sense organs, computes it and passes results.

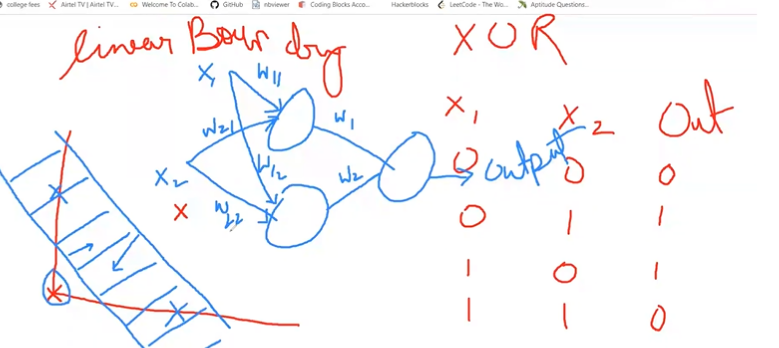


Artificial Neurons / perceptron started in 50s.

Each input Xi has a attached weight to it Wi.



Capable of drawing a linear/straight line. Xi are inputs, Mi are weights, perceptron has the function h(mi). The arrows are called synapses.



Can use multiple perceptron to get complex regions. Use activation function to get non linear decision boundaries like can use sigmoid functions.

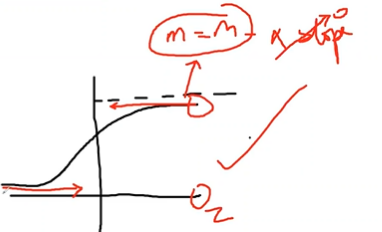
There are layers. Starting or first layer, hidden layers, output layer.

Forward propagation : data signals move starting to output layer. Feed -> NN -> Output

Back propagation : Calculate cost/loss, to modify weights move back.

Can use Gradient decent also like in Regression and all.

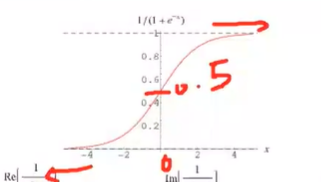
Vanishing gradient : in sigmoid function at the edges, the slope = 0, so modify func m = m-LR\*slope will never change m. This is called vanishing gradient.



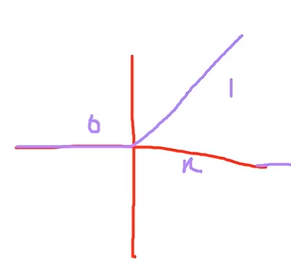
For this, initialize weight around [-0.5 to 0.5]. Also we can use some ither functions.

Functions :-

Sigmoid :



Relu (used commonly) : y=x , x>0 ; y=0 , x<= 0



tan (h) (also common)



Regularization

To avoid overfitting use regularization. Say some xi overpowers other, so we reduce its Mi .

J = (1/N) sum (Yp-Y)^2 - lambda (sum(mixi))^2

This is called L2 type regularization.

L1 = lambda\* ( sum(mixi)) (no squaring)

CNN Convolution Neural Network

In deep NN or Artificial NN, there was overfitting and img passed as a 1d array so a pixels relationship with just left and right pixel is considered, not with the pixel above or below it. Also, suppose we train for getting numbers, if the img is tilted, then it will not be able to detect it at all as it knows only straight no.

Basically any kind of img transition : translation motion, rotation, resize, illumination, flip.

A cat should be recognized wherever it is in img, of whatever size and how much edited img.

This all is overcome by CNN.

It is a powerful deep network widely used in image related tasks like image recognition, segmentation, computer vision etc.

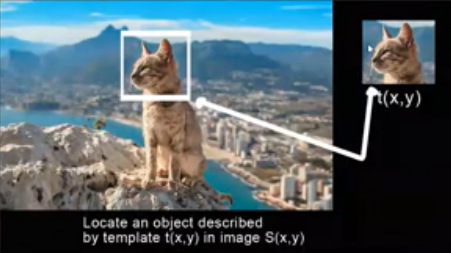


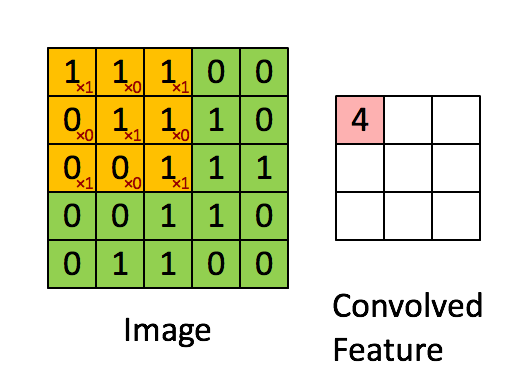
Softmax activation function. 1d array as dimension of the layer.

CNN have kernels, feature maps.

the kernel is nothing but a filter that is used to extract the features from the images. The kernel is a matrix that moves over the input data, performs the dot product with the sub-region of input data, and gets the output as the matrix of dot products.

The feature maps of a CNN capture the result of applying the filters to an input image. I.e at each layer, the feature map is the output of that layer. The reason for visualising a feature map for a specific input image is to try to gain some understanding of what features our CNN detects.

 the cat box is traversed all the way through the matrix , and if it looks similar it is predicted, this is a feature map.

 here kernel is 3x3. Img is 5x5. Feature map is pink one. Kernel = [101,010,101]

Feature map size = img size – kernel size + 1 = 3 here.

Kernels can of different types, to get blur, to get edges etc.

Stride : movement of the kernel or the filter. Like we were moving 1 to right in a step, 1 down, so stride=(1,1). But in general use stride = (2,2). It means how many pixels we skip each time.

Padding : say have 5x5 images, stuffing its surrounding with 0 is called padding. If we pad with (2,2), then matrix becomes 7x7. The boundary is two 0s.

Same convolution will have same nxm, then we do padding

IMG nxm X CONV 🡪 IMG nxm

But if nxm is reduced it is called Valid Convolution, in that case don’t do padding.

Pooling layers : pooling done after Convolution operation

2 Types :

Average: slides an (f,f) window over the input and stores the average value of the window in the output.

max pooling : slides an (f,f) window over the input and stores the max value of the window in the output.

Helps reduce computation by discarding 75% of neurons ( for stride=2, filter=(2,2))

Makes feature detectors more robust

No parameters for learning, just hyperparameters like filter\_size

CNN Architecture

Input img (RGB 3 channels) 🡪 convolution generates Feature Map 🡪 apply function RELU 🡪 do max pooling to reduce the size of the img 🡪 convert to 1d array and pass to NN (dense layer)

[ Input -> convolution layer -> ReLu -> Pooling ] -> Softmax

