

Introduction to Neural Networks

Dr. Osita Onyekwelu :

Logistic Regression : Classification algorithm :

Quick Take on Deep Learning : Sub-set of Machine Learning

Uses : Computer Vision

Uses : Natural Language Processing

Uses : Speech Recognition

Deep Learning is extremely computationally expensive.

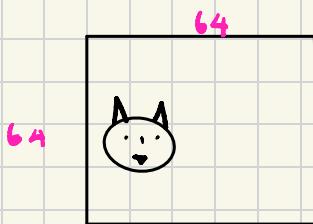
- People have had to find techniques to parallelize the code.
- Data available has been growing exponentially since the internet bubble.
- This type of algorithm is well known when there is a platter of data.
 - models are very flexible.
 - The more data available the better the algorithm can learn salient features within the data.
- Algorithms:

I) Logistic Regression

Goal : Predict class in image (Binary Classification)

If there is a cat in the image, we want to output a number close to 1
If there is no cat in the image, we want to output 0.

For now but we just assume that we are dealing with just one cat in the image and no more.



Note that it is computer science imaging to be flattened/represented as 2-D matrices.

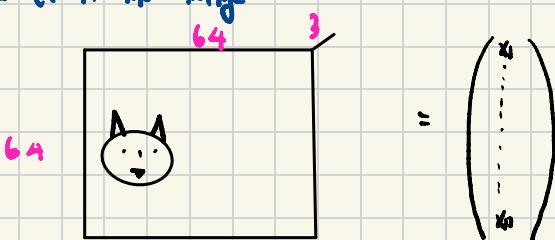
i.e. Assuming the image on the left is a color image of size 64×64 , we have $64 \times 64 \times 3$ numbers to represent those pixels.



RGB Channel

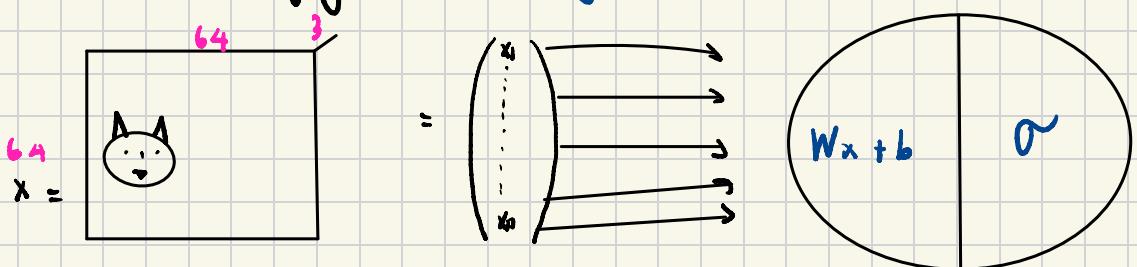
• Every pixel in an image can be represented by 3 numbers.

- One representing the red filter, one representing the blue filter and another representing the green filter.
- We will now flatten this image into a vector before using logistic regression to see if there is a cat in the image.



→ taking all the numbers in this matrix and flatten it into a vector.

- We can now use our logistic regression because we have a vector input.
- We will now take the vector and push them into one operation known as the logistic operation, which has one part which is $Wx + b$, where x is the image, and the second part is going to be the sigmoid.



- The sigmoid function is a function that takes a number between $-\infty$ and ∞ , and maps it between 0 and 1. Very common for a classification problem.

A diagram showing the sigmoid function and its mathematical derivation. On the left, a circle is divided vertically, with the left half labeled $Wx + b$ and the right half labeled σ . An arrow points from this circle to the right, where the sigmoid function is defined as $\hat{y} = \sigma(W^T x + b)$. This equation is then expanded using the definition of the sigmoid function, $\sigma(z) = \frac{1}{1 + e^{-z}}$, to show that $(\sigma^T x) = \sum \Theta_i x_i$ and $(W^T x + b) = \sum \Theta_i x_i + \Theta_0$.

Question : What is the shape of the matrix W ?

Answer :

$$\hat{y} = \sigma(Wx + b)$$

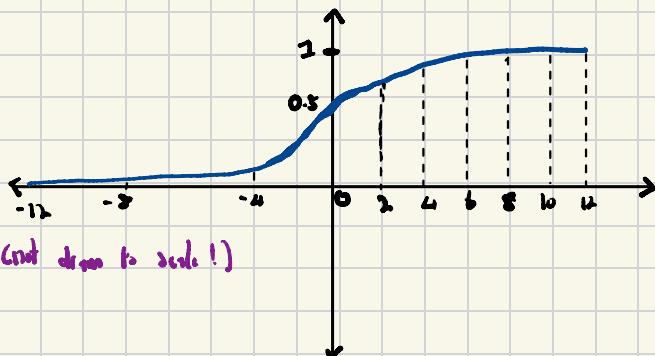
(1×1) ($60 \times 60 \times 3, 1$)
 (12, 288)

∴ We have a **Row vector** as our **parameter**.

Once we have the model, we need to train it. The process of training is that

- i) initialize parameters W, b (find the right parameters)
ii) Then optimise them. Find the optimal W, b .
iii) After finding the optimal $\{W, b\}$, we train to predict.
 $\Rightarrow \hat{y} = \sigma(Wx + b)$ to predict.

And this is a sigmoid function:



$$\begin{aligned}\sigma(x) &= \frac{1}{1 + e^{-x}} \\ &= \frac{e^x}{1 + e^x} \\ &= 1 - \sigma(-x)\end{aligned}$$

- Note that finding the optimal W means defining the loss function, which is the objective.
- In machine learning, it is often the case such that while you have a function, that you know you want to find (output function) but you do not know the value of its parameters. In order to find them

You are going to use a proxy, that is going to be your loss function. If you manage to minimize the loss function you will find the right parameters.

- We define a loss function that is the Logistic loss or

$$J = -[y \log(\hat{y}) + (1-y) \log(1-\hat{y})]$$

originating from Maximum Likelihood Estimation, starting from a probabilistic model.

- The idea is now how do we minimize this function, using a gradient descent algorithm. \Rightarrow iteratively compute the derivative of the loss function with respect to our parameters.

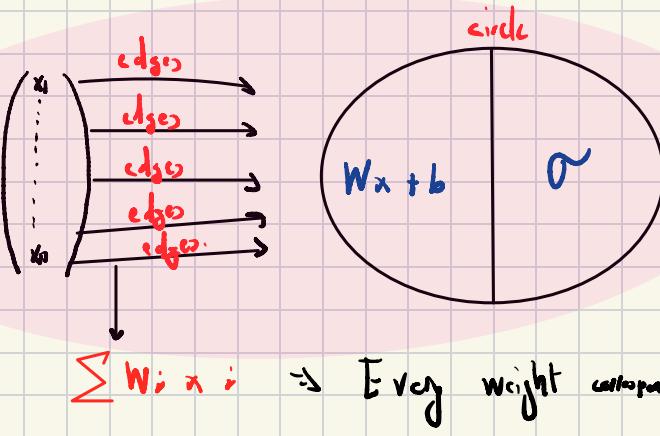
$$\begin{cases} w = w - \alpha \frac{\partial J}{\partial w} \\ b = b - \alpha \frac{\partial J}{\partial b} \end{cases}$$

- At every step we will update them to make this loss function go smaller.
- At every point you compute the derivative of the loss with respect to your parameters.
- Note that: $\sigma' = \sigma(1-\sigma)$

Q: How many parameters does the Logistic Regression have?

A: 12,783 weights + 1 bias = 12,784 parameters.

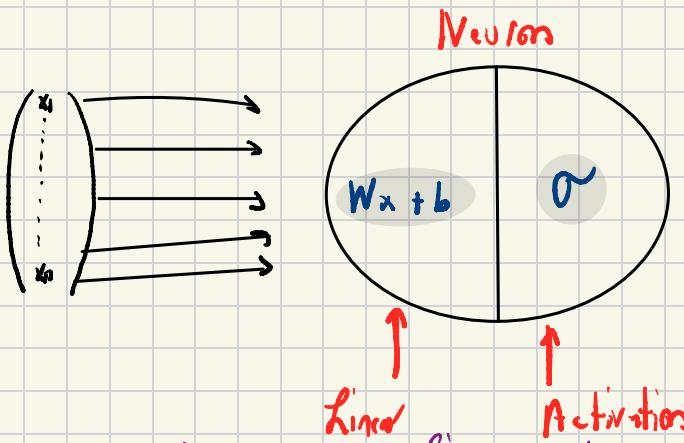
Note that every circle has a bias & every edge has a weight!



2 Important Equations to know :

$$\textcircled{1} \quad \text{Neuron} = \text{Linear} + \text{Activation}$$

(A neuron is an operation that has 2 parts, 1) Linear 2) Activation)



Take the output of the linear part & put it into the activation (sigmoid) function in this way however it can be other functions).

$$\textcircled{2} \quad \text{Model} = \text{architecture} + \underset{w, b}{\text{parameters}}$$

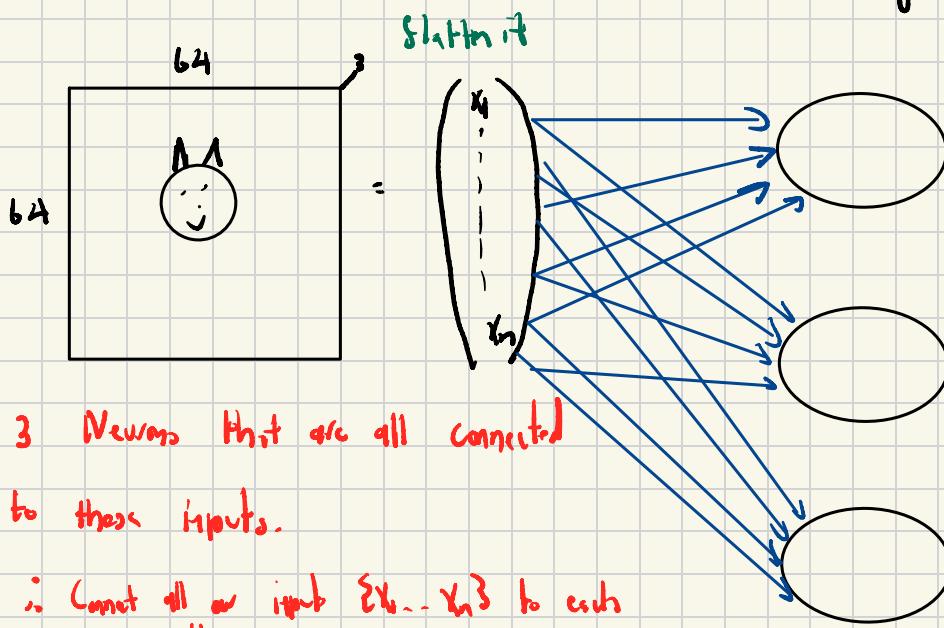
Here we are trying to train a logistic regression in order to be able to fit

When people say "we have shipped a model" in industry, what they are saying is that we have found the right parameters with the right architecture.

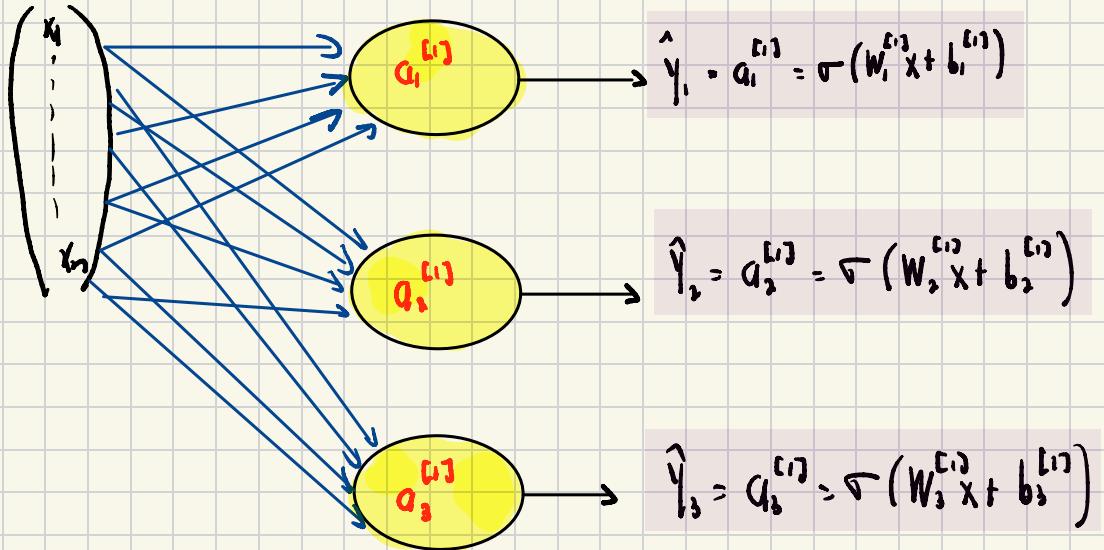
For this decision, we will modify the architecture extensively (adding neurons etc) and the parameters will always be called $w \neq b$. They will become bigger and bigger because we have more data. We will need more parameters to understand what a cat is.

Goal 2: Find Cat/Human/Igloo in images
3 types of animals.

≈ Put 2 more circles/neurons & do the drawing.



We will use a specific set of Notation here:



The square brackets represent what we will call the **layer**. If you look at this network, it looks like there is just one layer currently present. In this one layer, the neurons do not communicate with each other.

- The index that is the subscript to the "a" is the number identifying the neuron inside the layer.
- Thus we have \hat{y} , and instead of being a single number as it was before, is known as a vector of size 3.

Question: How many neurons does this network have?

Answer: 3 x what we had before because we added 3 more neurons.

Note that we are dealing with Binary Classification with images and labels.

- A **label** for an image, which has a cat would probably be a vector with a one and two zeros.
- One should represent the network as follows. With the following label.

$$\begin{array}{l}
 \text{P/cat} \leftarrow \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} p(\text{cat}) \\
 \text{P/him} \leftarrow \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} p(\text{him}) \\
 \text{P/guitar} \leftarrow \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} p(\text{guitar})
 \end{array}
 \quad \left. \right\} \text{Assume we use this scheme to represent the data set.}$$

∴ We train this network using the same technique here such that you initialize all of your weights & biases with a starting value, optimize the loss function by varying $y = \dots$ to gradient descent, and predict.

- The neuron a_1 will be responsible for detecting cats because we decided to label our data set in such a manner. Thus, after training, the neuron a_1 is naturally going to detect cats.
- If we changed the labeling scheme & said the second entry would correspond to the cat, then after training, you will detect that the neuron a_2 is responsible for detecting a cat... etc
- The network will evolve depending on how you label your data set
- Do you think that this network could still be robust to different angles in the same picture?

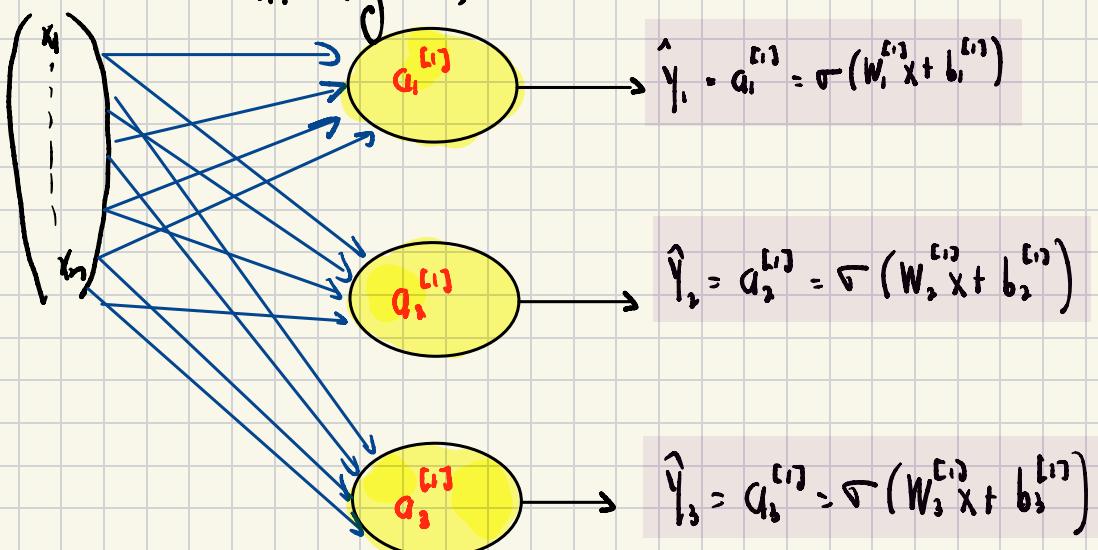
- This cat now has a friend that is a him.



$$= \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} \begin{matrix} p(\text{cat}) \\ p(\text{him}) \\ p(\text{ignore}) \end{matrix}$$

- The network only sees $(1, 1, 0)$ is an image. It does not see which "1" corresponds to the cat or the him. With neural networks you do not need to tell them everything. If you have enough data they will figure them out.
- Even though you will have multiple of those animals either together or alone in a picture, the neuron will know what it is looking for and will understand that a specific "1" corresponding to a specific animal and yes, it is going to be robust. You just need a lot of data of course to achieve this.

- This system is also going to robust because the 3 neurons are communicating with each other & thus we can train them



independently from one another. Further, the Sigmoid in y_2 does not depend on the Sigmoid in y_1 , and the Sigmoid in y_3 does not depend on the Sigmoid in y_2 .

⇒ That we can have $\begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$ as an output. So you can think

of it as 3 Logistic Regressions. So We will not call this a Neural Network yet.

You can also think of this as a 3-Neuron Network but NOT a Neural Network

Question: Is there a case such that we have a constraint that leads to only one possible outcome? Such that there is no 'but and then', it is either a cat or a dog.

Answer: Think about health care. There are many models that are made to determine whether or not someone has a skin disease based on cell microscopic images. So you want to classify a specific disease over a large number of diseases.

i.e. The model we have will still work but it will not be optimal because it's going to take longer to train.

i.e. one disease might be extremely rare and one of the neurons is never going to be trained

i.e. You are working in a Zoo & there is 1 lion & 1 tiger. The third neuron is never going to be trained

is you would want to start with another model whereby
you put the constraint such that "There is only one
disease that we want to predict."

Then let the model train with all the Neurons **Training Together**
by **sharing interactions between them.**
