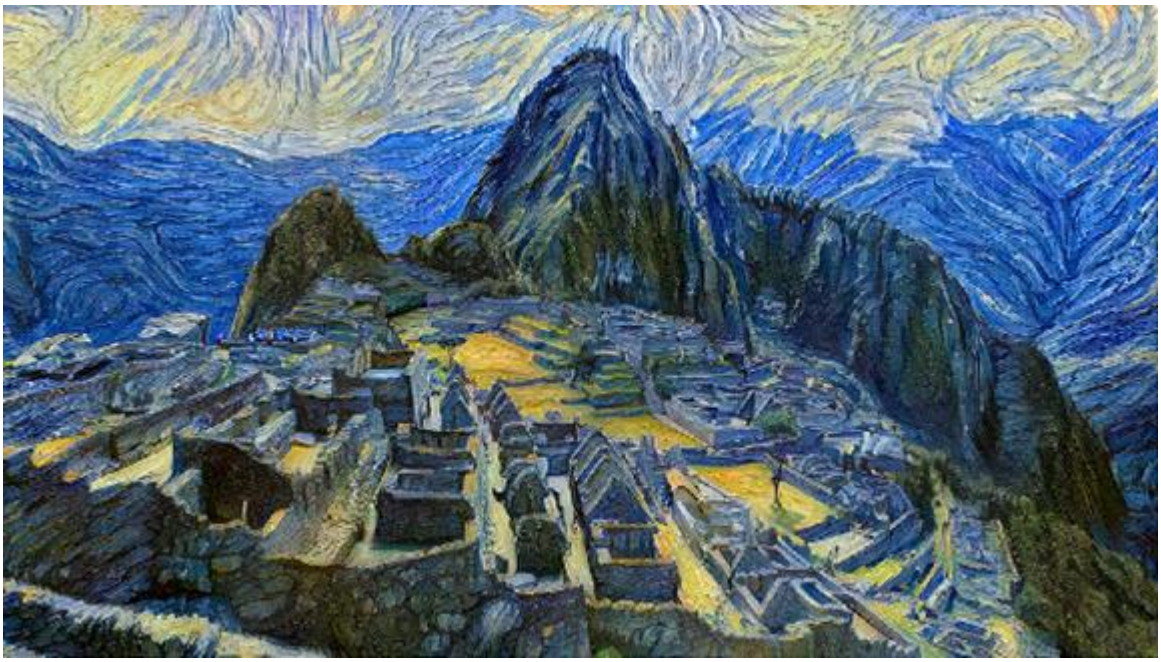
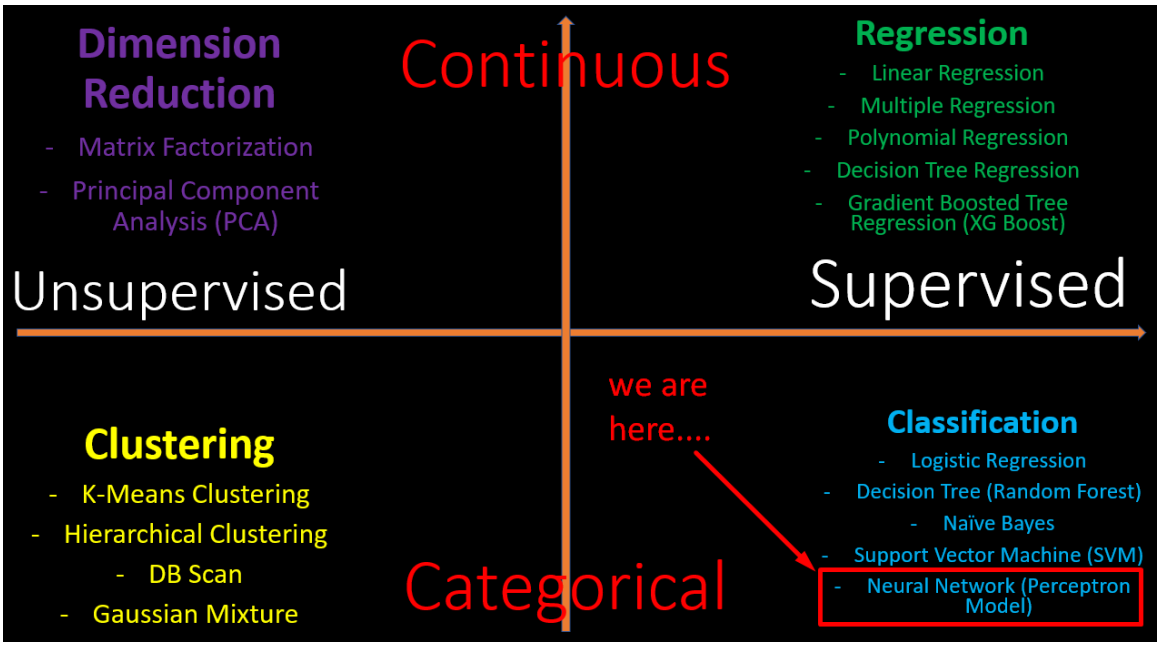
ARTIFICIAL NEURAL NETWORK (ANN)

HOW IT WORKS
DR. ALVIN ANG



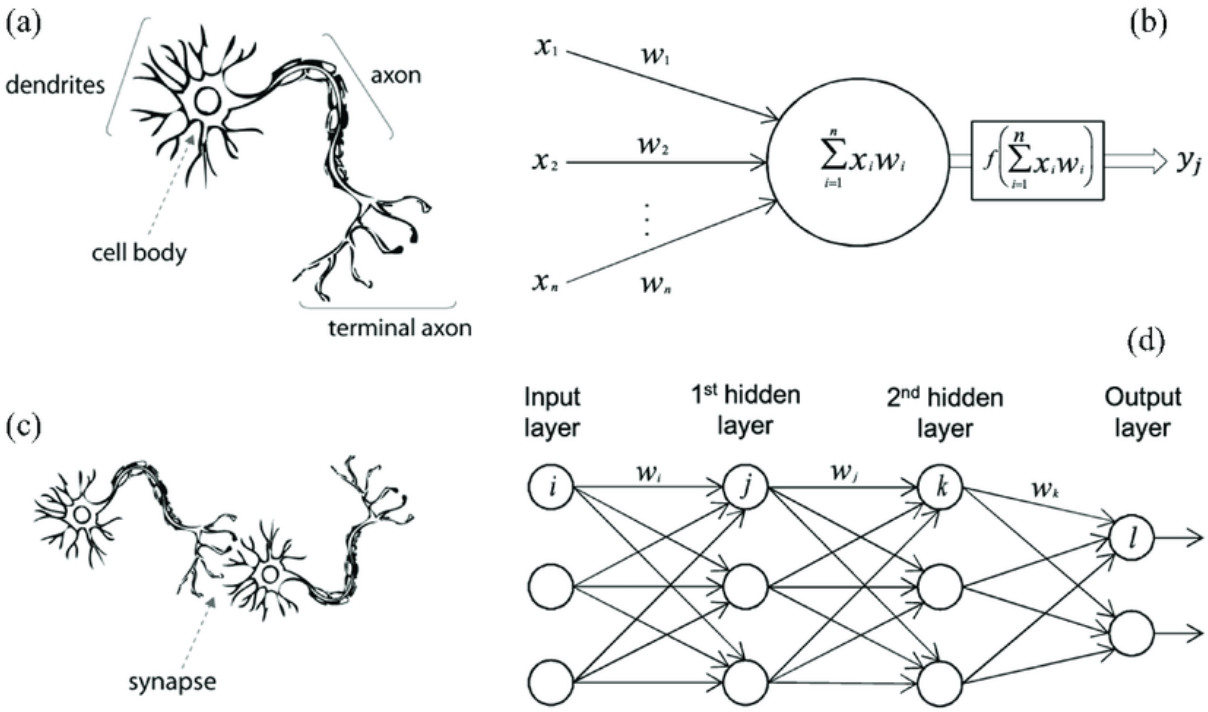
CONTENTS



Above is a table categorizing the different Machine Learning algorithms.

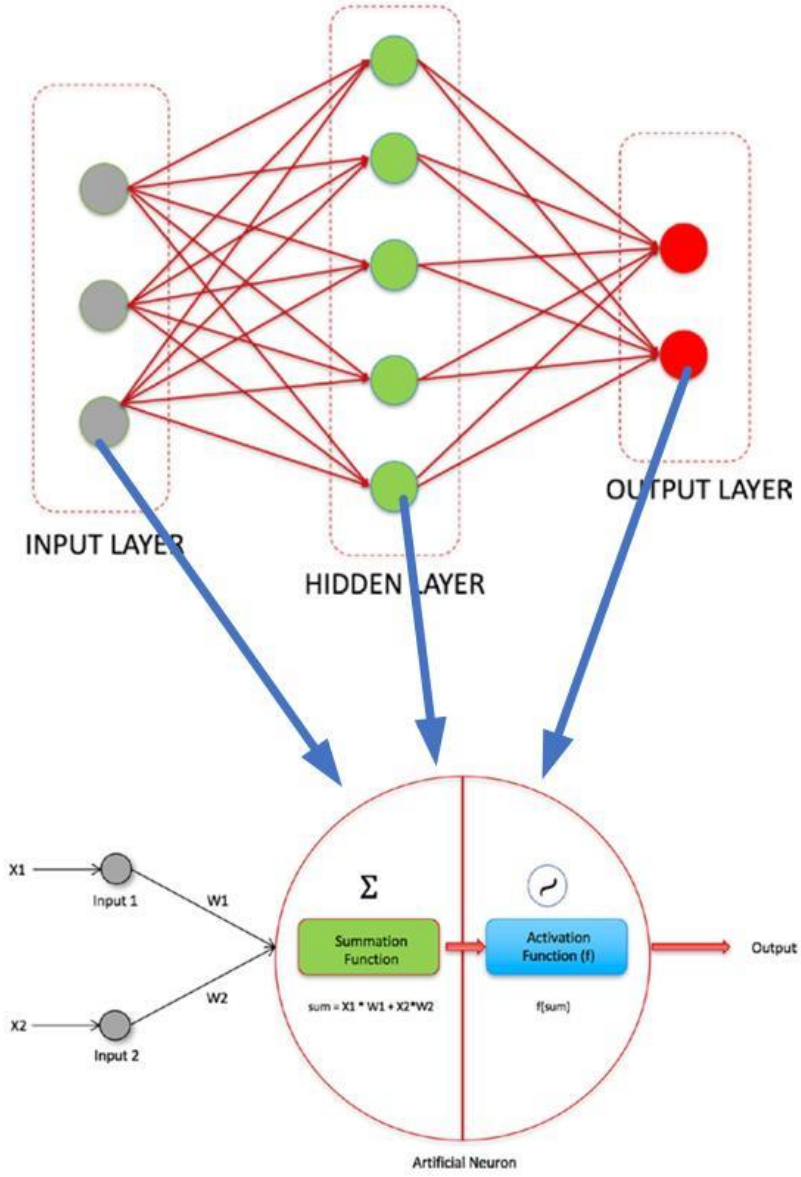
Objective of Neural Network is to predict a CATEGORY.

(actually, it can also be used to predicut Regression….but most literature use it for classifying images like cats vs dogs….so we mainly use it for Classification….)



"Developed by OpenAI, ChatGPT is based on a powerful neural network architecture that has been trained on massive amounts of text data, giving it an unparalleled ability to understand and generate language."

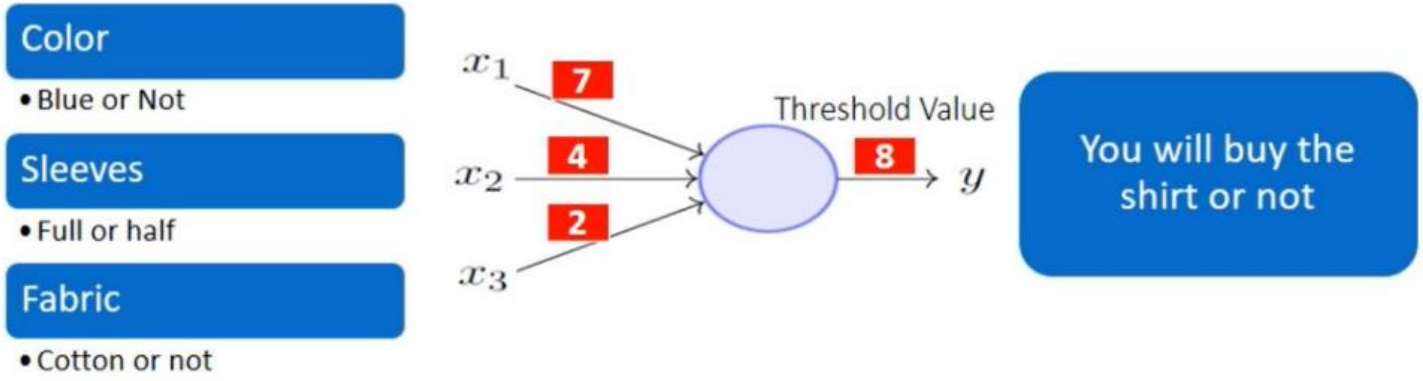
https://www.linkedin.com/pulse/how-does-chatgpt-technology-work-juneconnects/



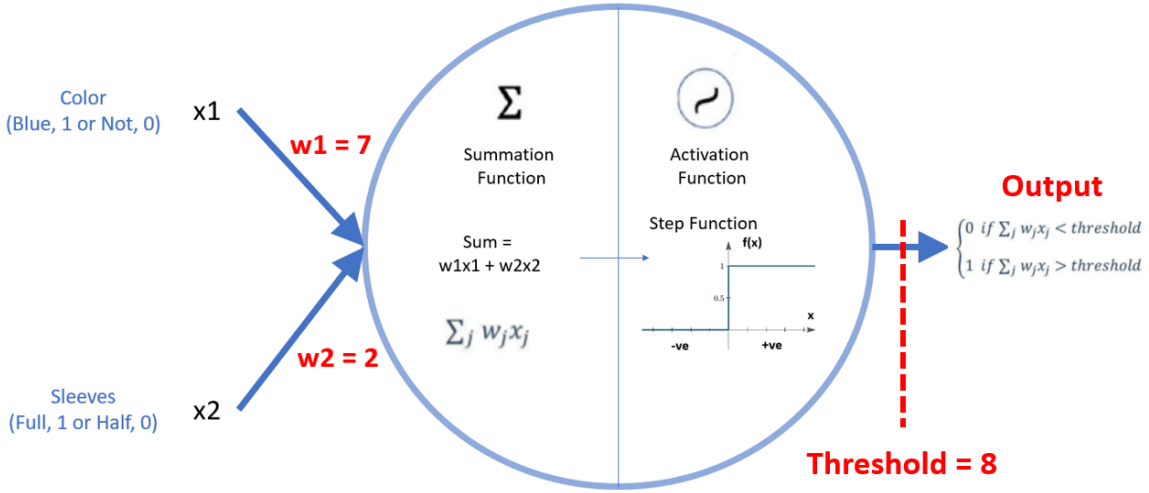
A Neural Network can be made of many layers.

In this example, we presume 3 layers: Input Layer / Hidden Layer / Output Layer.

Every node in the Neural Network is a Perceptron.



Color	Sleeves	Fabric	Calculated Sum	Threshold	Buy / Not Buy
Blue	Half	Non Cotton	7*1 + 4*0 + 2*0 = 7	8	Not buy
Blue	Full	Non Cotton	11	8	Buy
Not Blue	Full	Cotton	6	8	Not Buy



Color	Sleeves	Sum	Threshold	Buy, 1 / Don't Buy, 0
Blue, 1	Half, 0	7*1 + 2*0 = 7	8	Don't Buy
Blue, 1	Full, 1	9	8	BUY
Not Blue, 0	Full, 1	2	8	Don't Buy

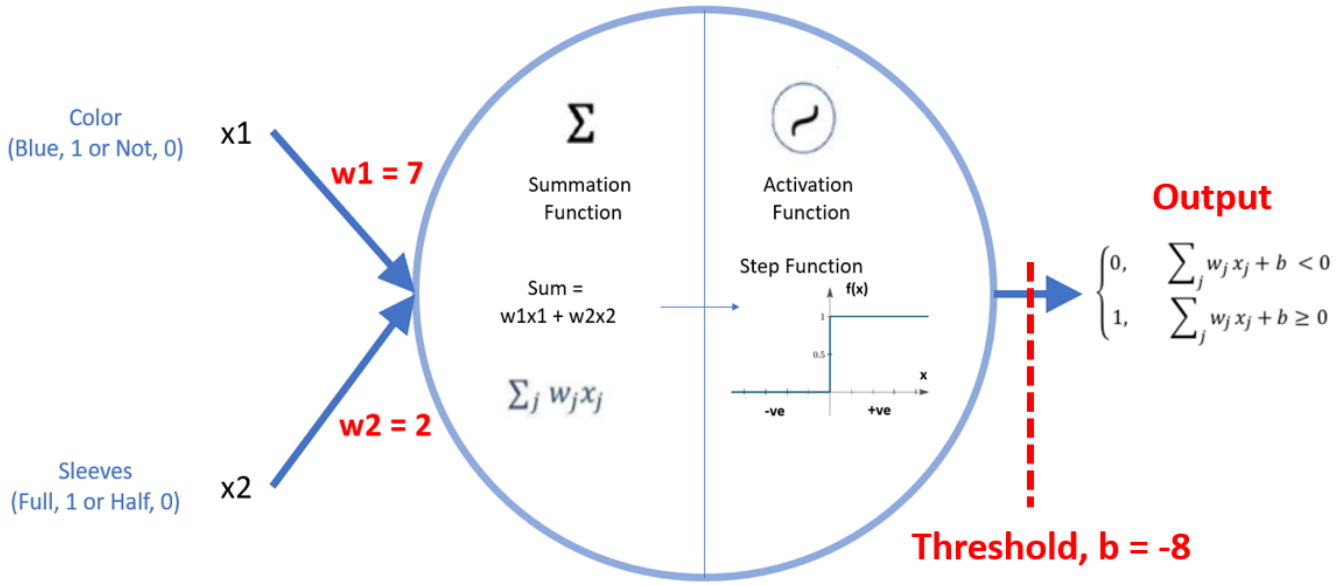
$$output = \begin{cases} 0 \ if \ \sum_j w_j x_j < threshold \\ 1 \ if \ \sum_j w_j x_j \geq threshold \end{cases}$$

We can bring the threshold to the LHS and label it 'b'

b is known as the BIAS (in this case, its our threshold level)

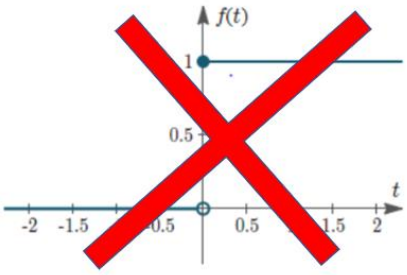
$$Output = \begin{cases} 0, & \sum_j w_j x_j + b < 0 \\ 1, & \sum_j w_j x_j + b \geq 0 \end{cases}$$

Therefore….



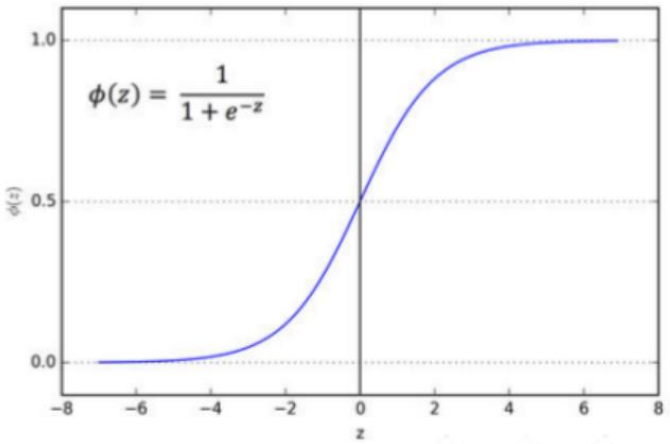
A. BUT STEP FUNCTION CANNOT BE USED IN DEEP LEARNING

Step Function cannot be used as an Activation Function in Neural Networks



Because it will fail during Backpropagation.

(More about this later)



Rather, we may use the Sigmoid Function.

The Step Function cannot be differentiated because the gradients are all 0 throughout.

This prevents Backpropagation (which is necessary because Backpropagation uses Differentiation to help in tuning the Weights and Biases of the NN).

First, let's understand what Forward and Back Propagation generally means (in the subsequent Chapters).

Thereafter, we will do an example of a detailed Forward and Back Propagation to understand this better (in the Appendix).

III. TRAINING PROCESS OF NN (HOW IT MIMICS OUR BRAIN)

A. DO YOU SEE A CAT OR A CROW?

At first glance, you see a Crow (because you mistook the ears for the beak).

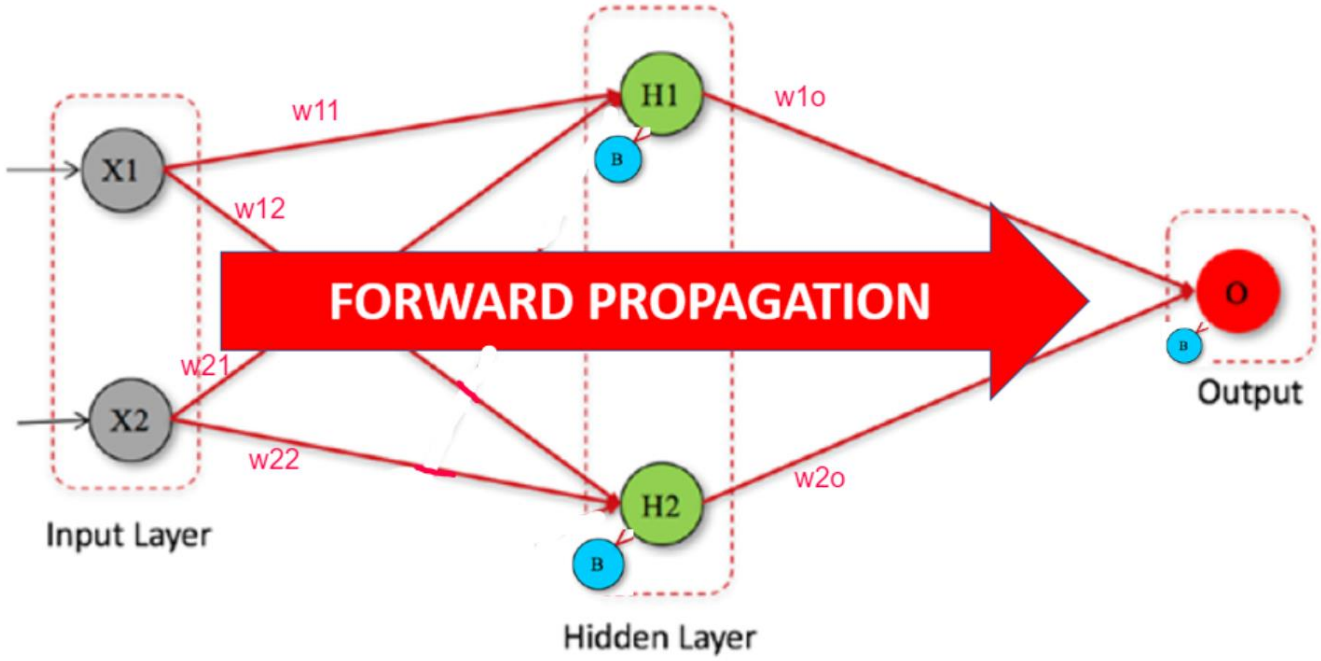
However, upon re-analysing, you realize it's a Cat.

The 'first glance' you took is analogous to ***"Forward Propagation"***, where your eyes fed your brain this image for the very first time.

Subsequently, while re-thinking, you were doing an ***"Error Calculation"***, where your brain is contemplating.

To reconfirm your thoughts, you did a ***"Backward Propagation"***, where your brain told your eyes to relook at the image closely.

B. FORWARD PROPAGATION



Note that the above picture does not depict the "Cat / Crow" photo recognition (as described in the previous section).

This picture is used solely to explain the concept of ***"Forward Propagation"***.

This is because the NN for classifying Cat / Crow will not have the configurations as above.

In Forward Propagation, the sequence goes from Input Layer → Hidden Layer → Output.

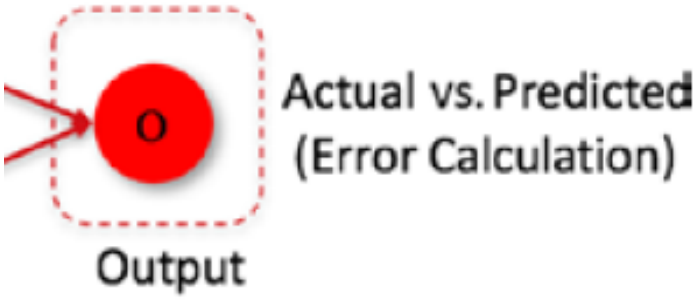
Each layer is an input for the next.

Wxx represent the Weights of each Neuron

B represent the Bias attached to each Neuron.

The Parameters (all Weights and Biases) are randomly chosen and initialized.

C. ERROR / LOSS CALCULATION



Once it reaches the End Layer (Output Layer), Error Calculation is done.

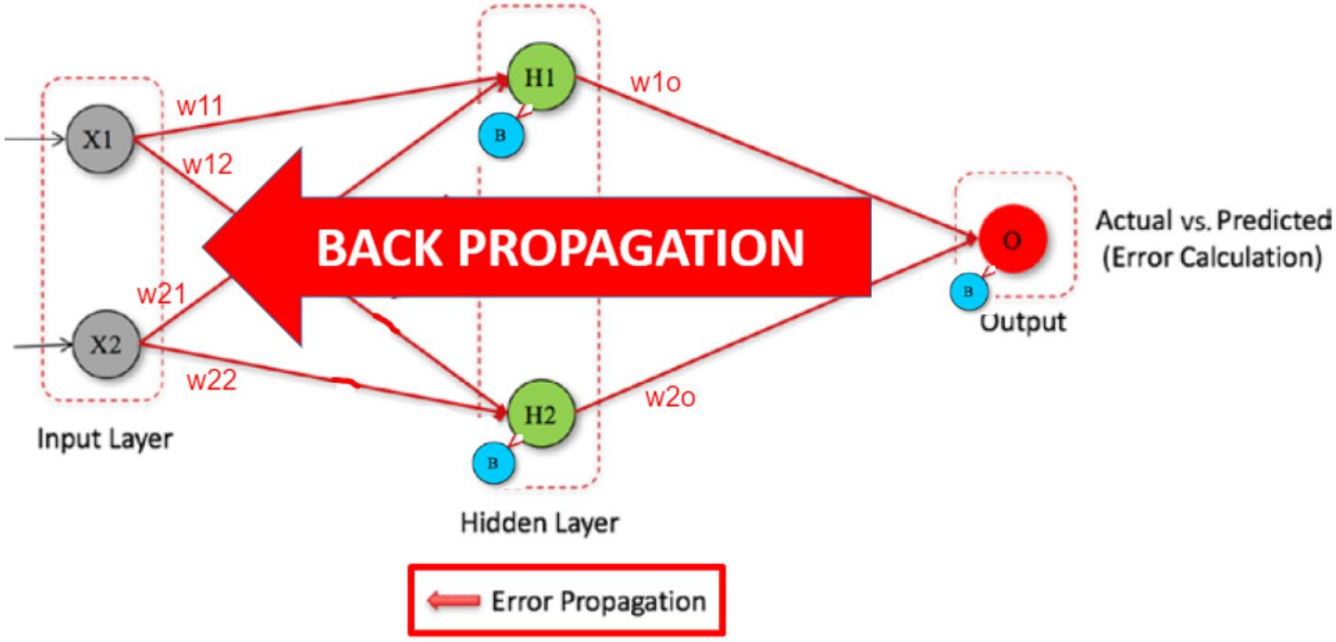
The Predicted value is compared with the Actual value.

If different, we will do Back Propagation (see next section).

(in this case, we analogously refer to the Error as the difference between a Crow vs a Cat…)

If the Error is very large, there will be major changes for the Weights and Biases.

D. BACKWARD PROPAGATION

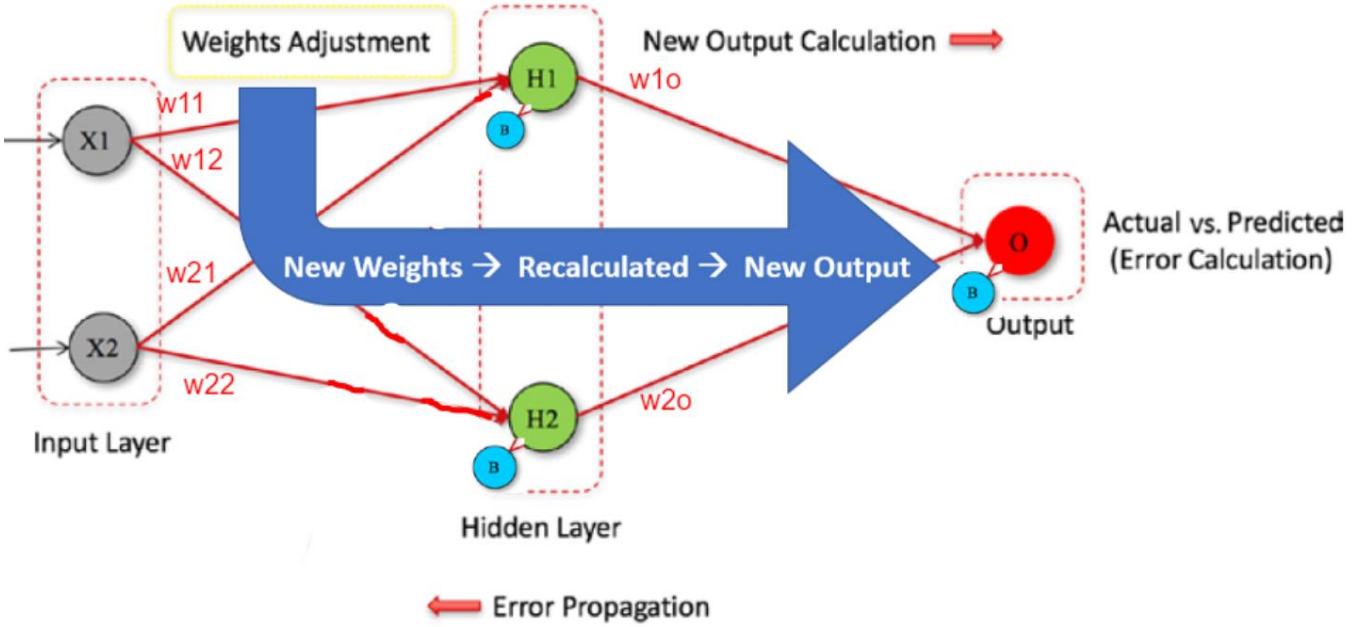


The Error (which is the difference between the Actual Value – Predicted Value), is backpropagated to the network, to adjust the Weights and Biases of the connections

Back Propagation reduces the overall error on the training data.

Detailed Back Propagation is done in the Appendix.

E. REITERATION

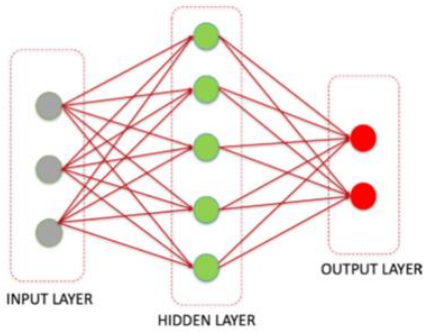


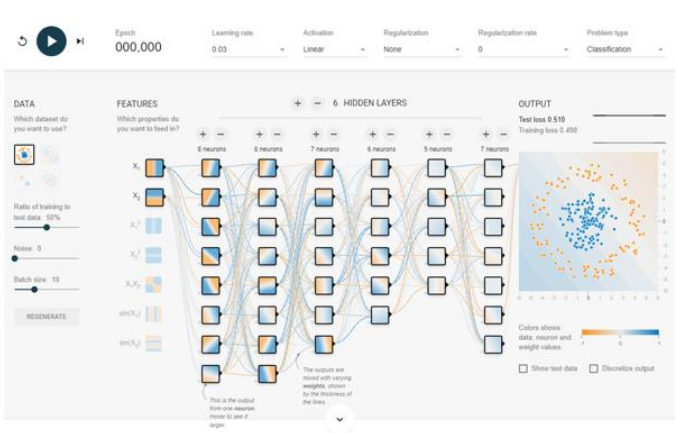
Weights are adjusted and Outputs are recalculated until overall errors are gone.

Process is repeated (Forward and Back Propagation) to the point that there is no further reduction of Loss / Error on the training data.

The final weights should now give the accurate new Output (Predicted = Actual).

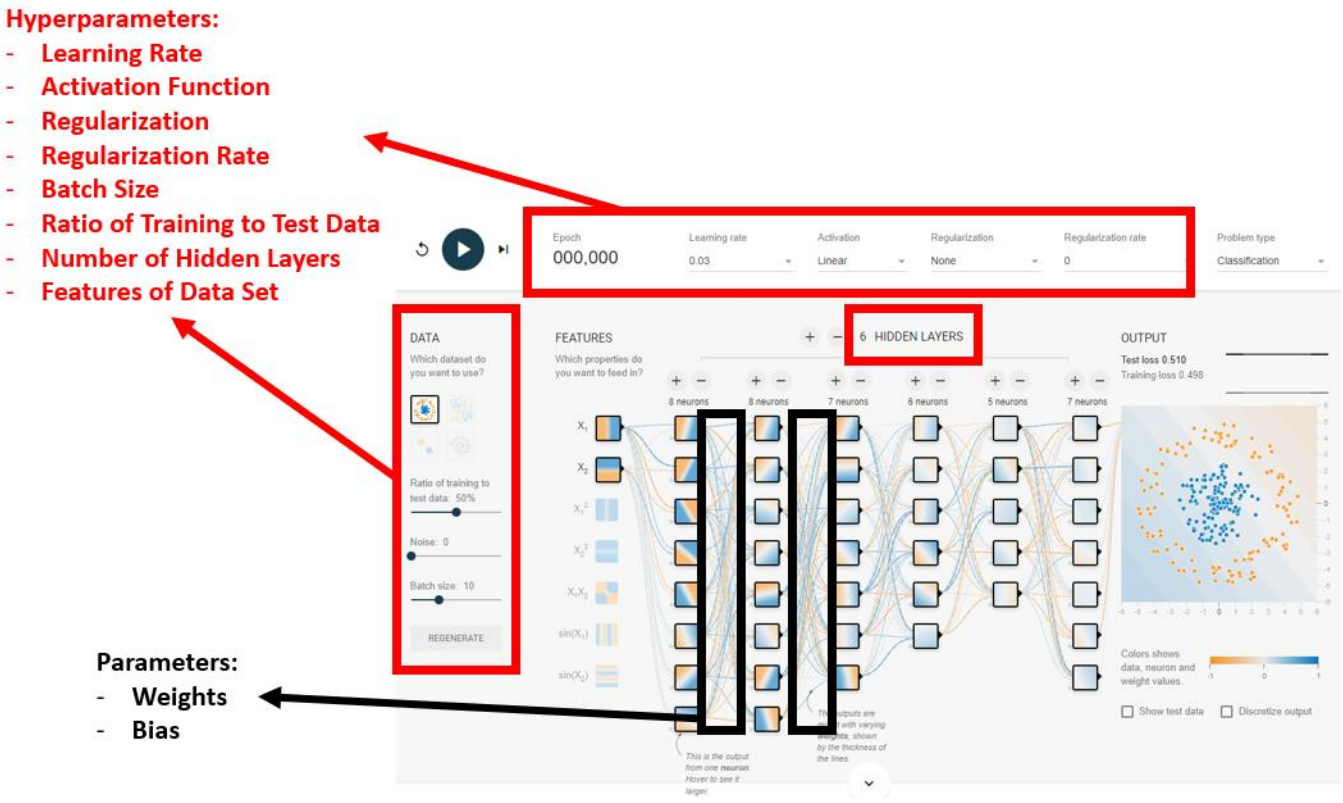
Neural Network Modeled in Tensorflow Playground





A Neural Network can be tested here at https://playground.tensorflow.org/

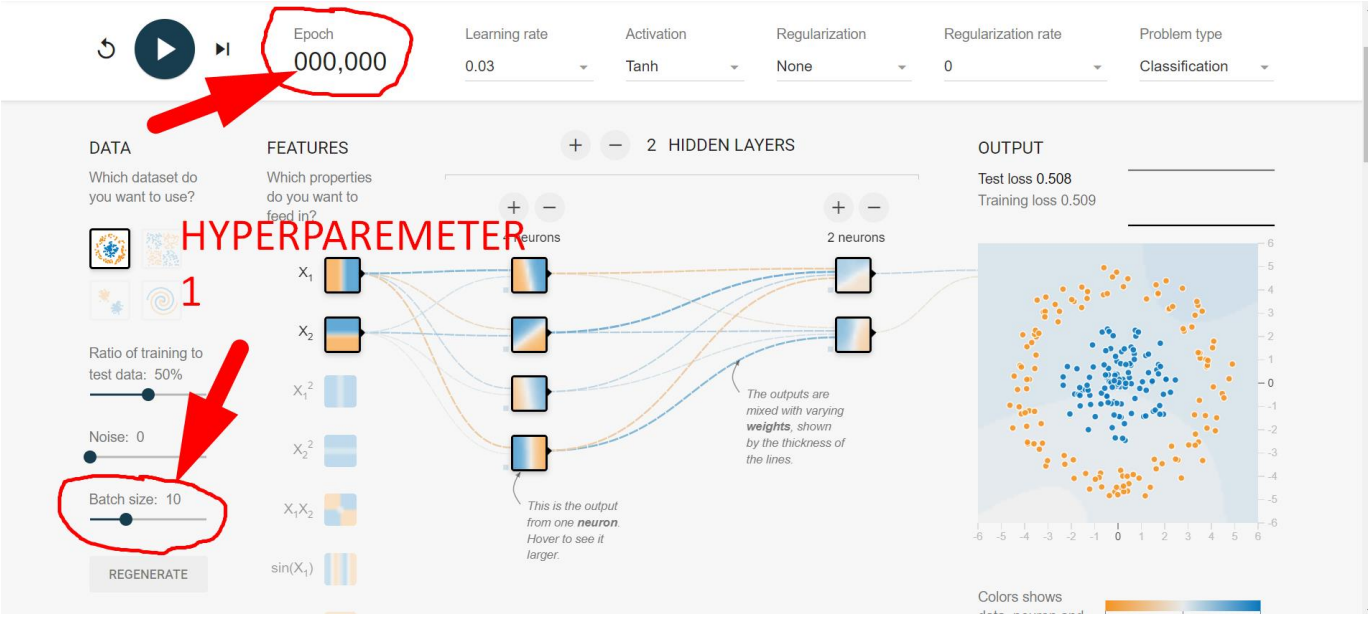
https://www.alvinang.sg/s/Learning-Tensorflow-Playground-by-Dr-Alvin-Ang.pdf



Hyperparameters are controls that can be set by the ML Engineer.

Parameters are uncontrollable items such as the Weights and Bias of every Neuron (that is automatically set by the NN after multiple epochs of Forward and Backward propagations).

V. HYPERPARAMETER 1: NUMBER OF EPOCHS AND BATCH SIZE

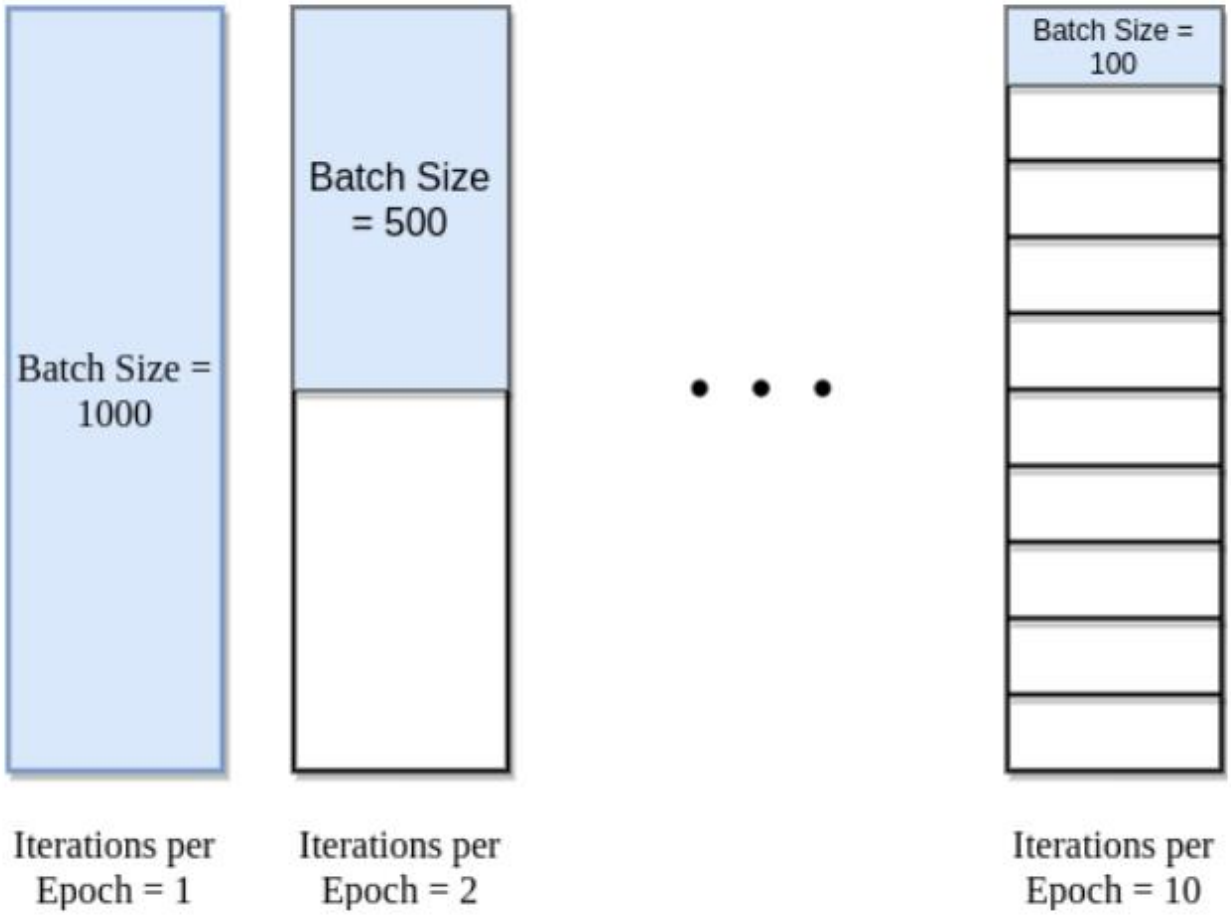


You may think of Batch Size as the number of Pictures per Batch fed into the "brain". Or, you may akin it to the number of pages of a textbook you study before going to an exam.

If you try squeezing 100 pages one shot (in an hour maybe?) into your "brain", it may result in ineffective learning because you might miss out important details.

However, if you read say 1 page very slowly per hour, you might take too long. But its very detailed studying.

A. EPOCH VS BATCH VS ITERATION

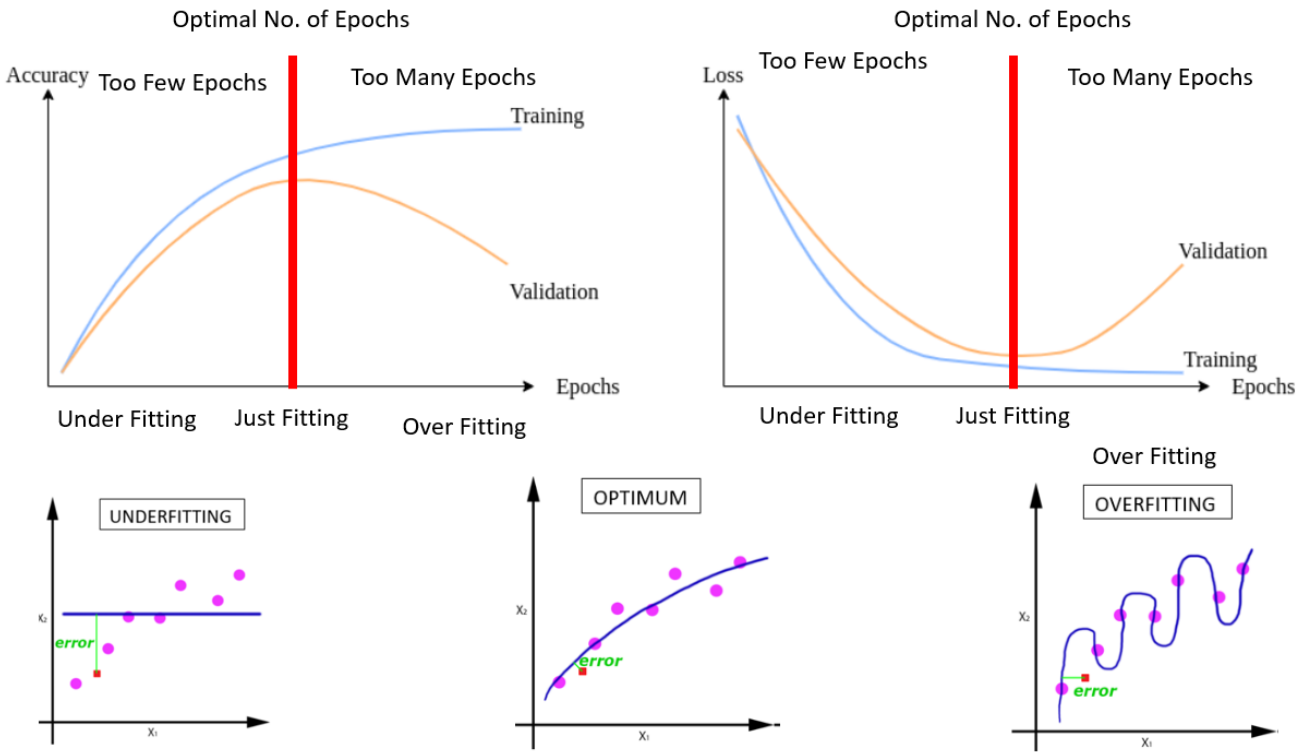


1 Iteration = 1 Forward Propagation + 1 Backward Propagation

1 Epoch may mean multiple Iterations.

If we divide the dataset of 1000 Samples into batches of 100 then it will take 10 iterations to complete 1 Epoch.

1. WHAT IS THE RIGHT NUMBER OF EPOCHS?



One Epoch is when an ENTIRE dataset is passed forward and backward through the neural network only ONCE.[1]

No right number of Epochs → We ALL have to go by trial-and-error approach.

Because every model is different, all data is different, number of Epochs required are also different.

The right number of Epochs occur when our TEST LOSS is the lowest and the Test and Training line are closest to one another.

When that happens, it also means that the NN has found the Optimum model (neither Underfit nor Overfit).

[1] https://www.baeldung.com/cs/epoch-neural-networks

2. TRAINING VS TEST (VALIDATION) DATASET

There are 2 coloured lines above: BLUE vs ORANGE.

BLUE = Training Dataset (the data that the NN used to train its model out from)

ORANGE = Test Dataset (the data that the NN is going to TEST its model on).

Sometimes, the Test Dataset is also called the Validation Set.

Sometimes, instead of splitting the Dataset into 2 parts (Train vs Test), you can split into 3 parts:

Train vs Test vs Validation

(All depends on how many rows of data you want to put into each set).

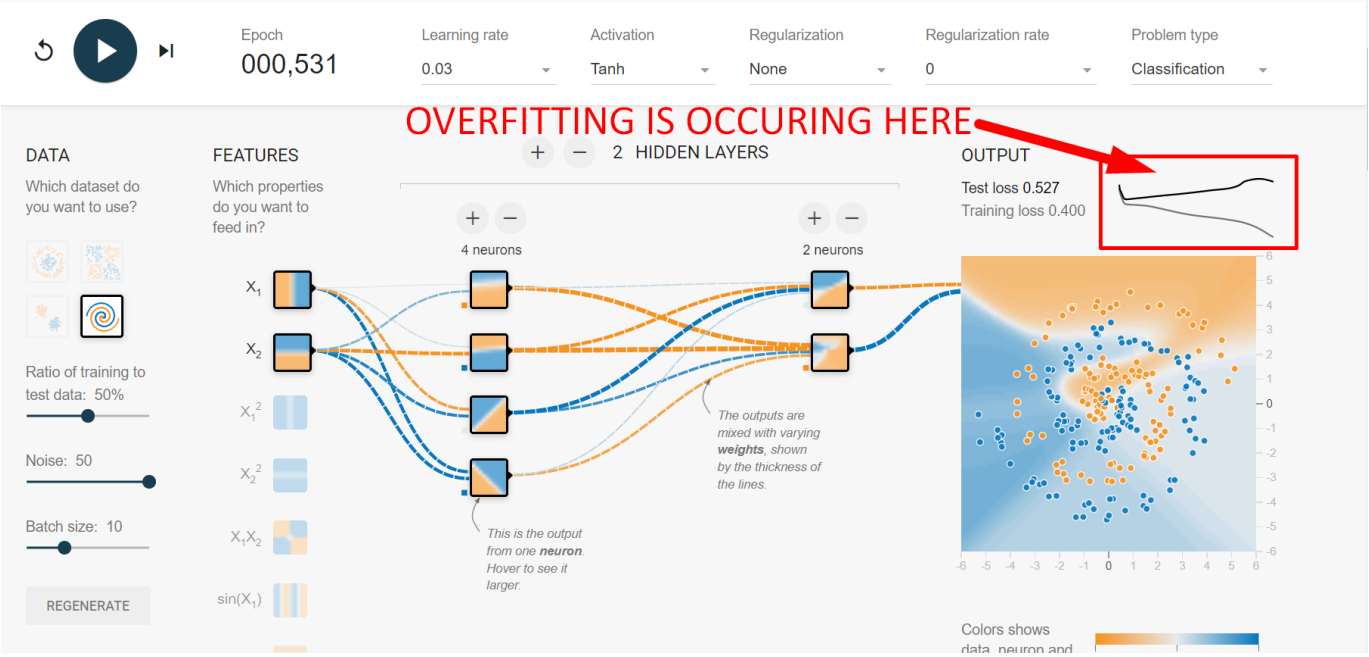
We only care about the ORANGE line (Test / Validation).

We don't care about the BLUE line (Training).

This is because we want the NN to perform on Data NOT PREVIOUSLY SEEN.

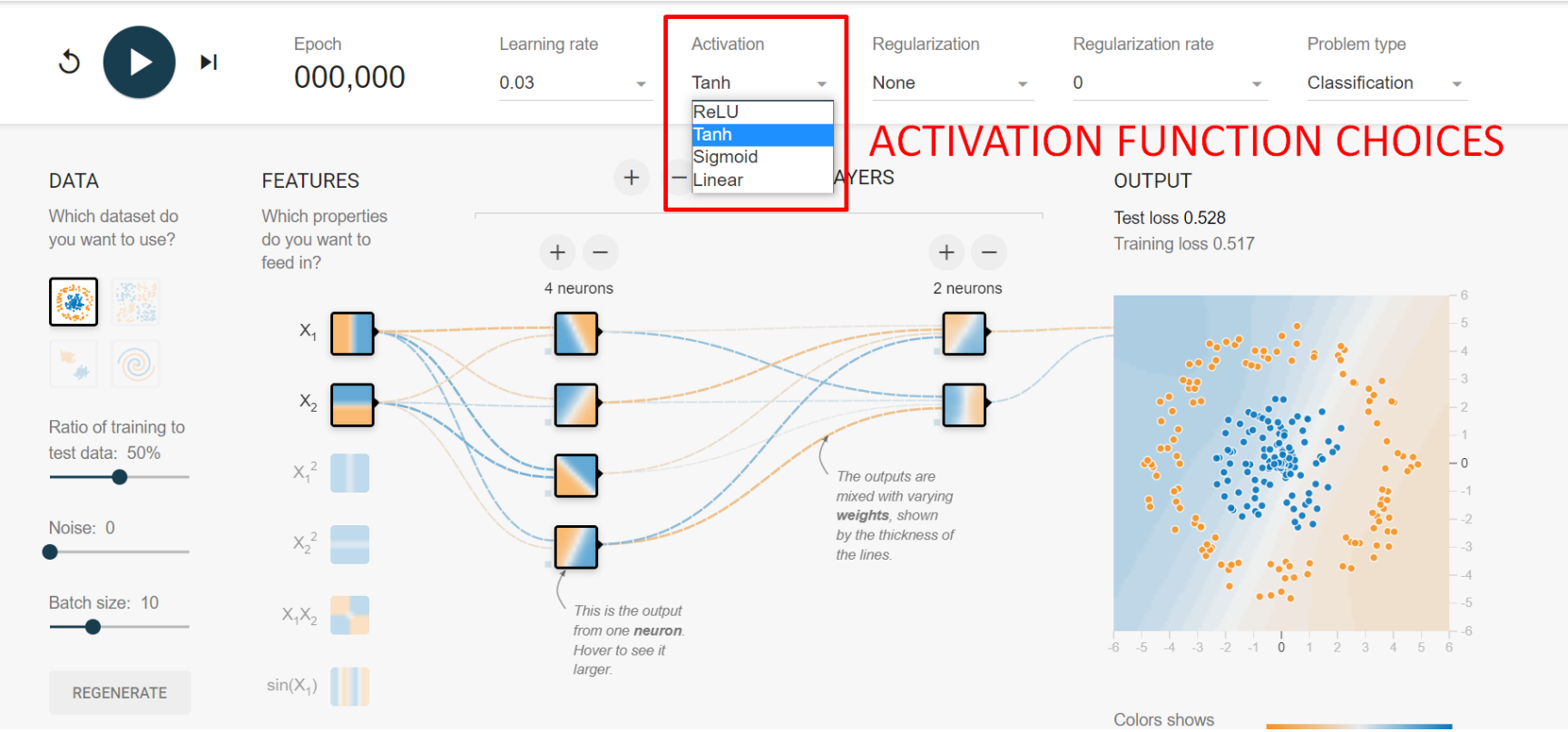
But the BLUE line is used to train the NN, thus the NN knows it perfectly well. (which explains why its Loss remains low even after many epochs).

B. OVERFITTING

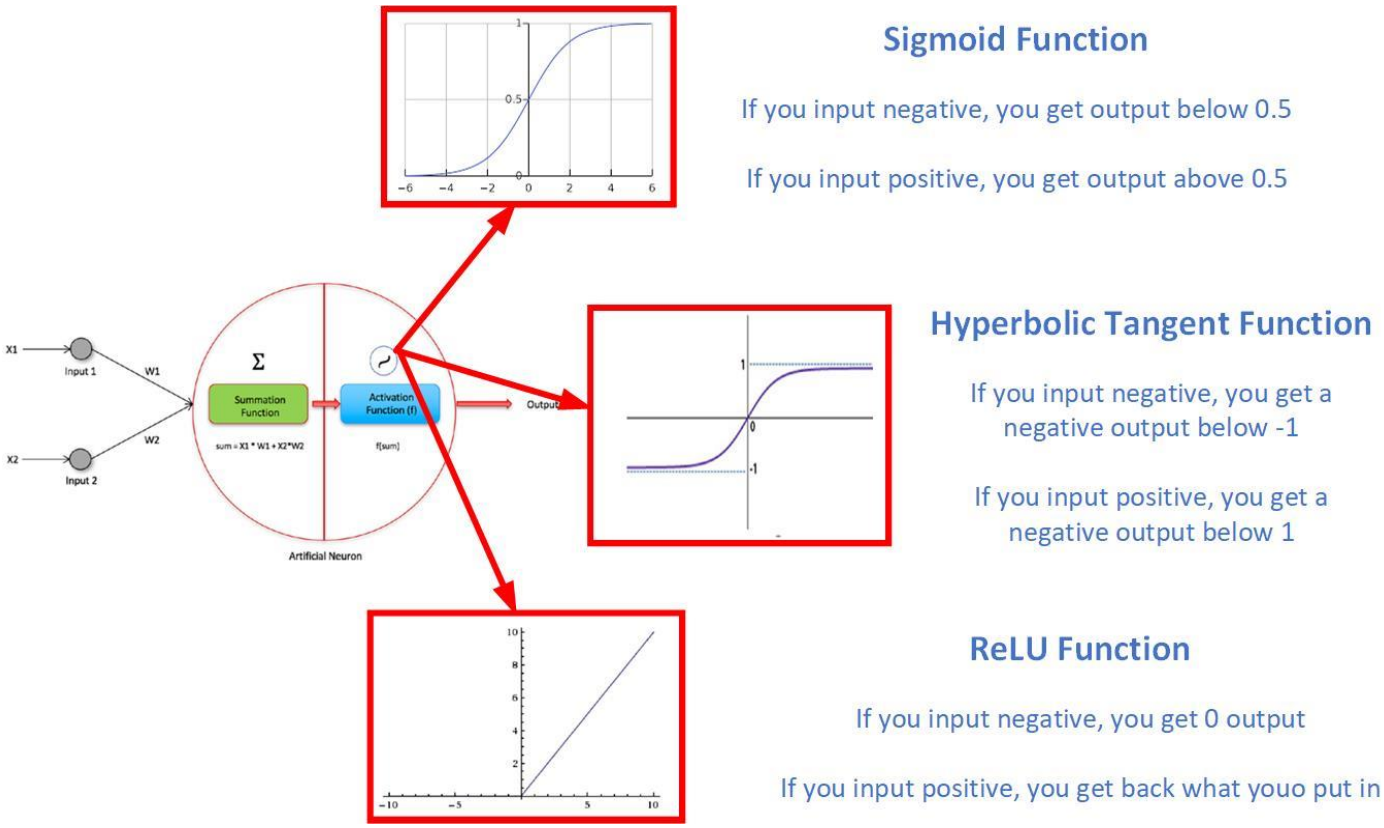


Overfitting occurs when the Test Loss line starts to deviate away from the Training Loss line.

This could be because the Data is very complex (with lots of noise) and we are using only a simple "brain" to solve it.. thus the "brain" overfits the model by fitting only simple models to complex data.



A. SOME COMMON ACTIVATION FUNCTIONS



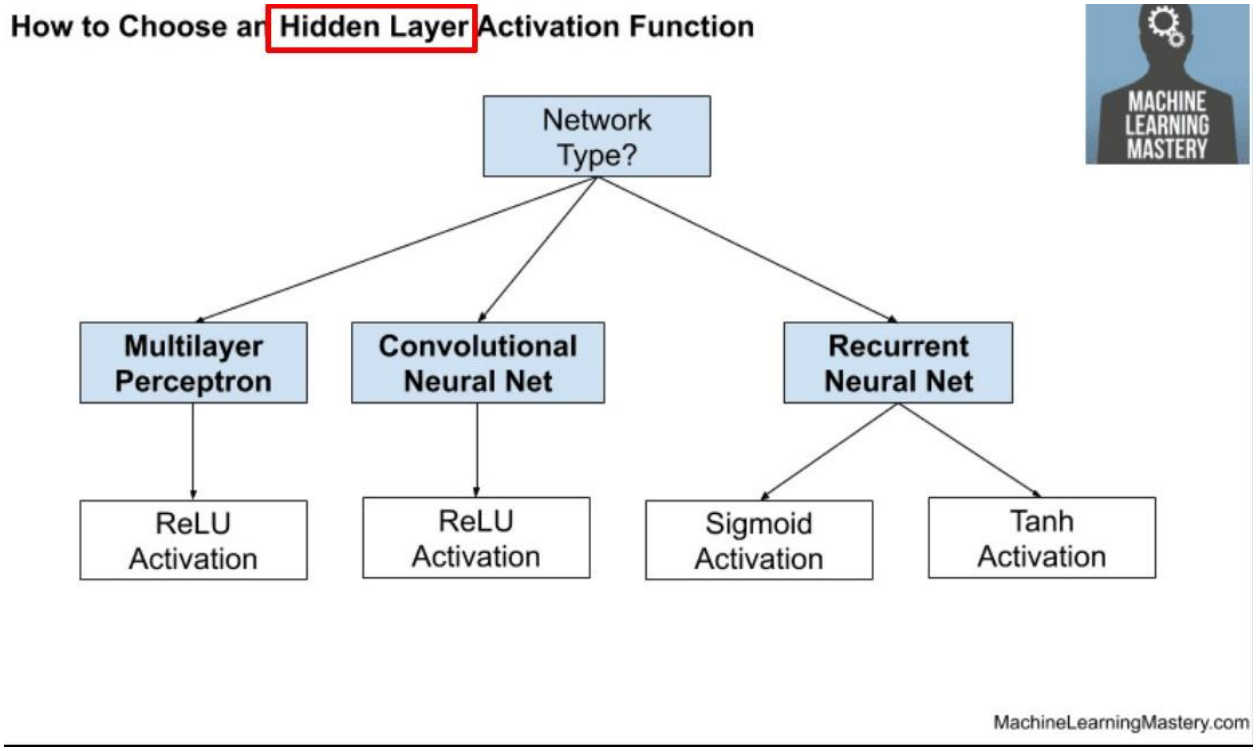
B. WHEN TO USE WHICH ACTIVATION FUNCTION?

1. FOR HIDDEN LAYERS ACTIVATION, IT DEPENDS ON YOUR NETWORK TYPE

In this manuscript, we shall only talk about ReLU because it is the most popular function.

And also, because we are only focusing on Artificial Neural Network (ANN) or also known as the Multilayer Perceptron Model (which purely uses the ReLU).

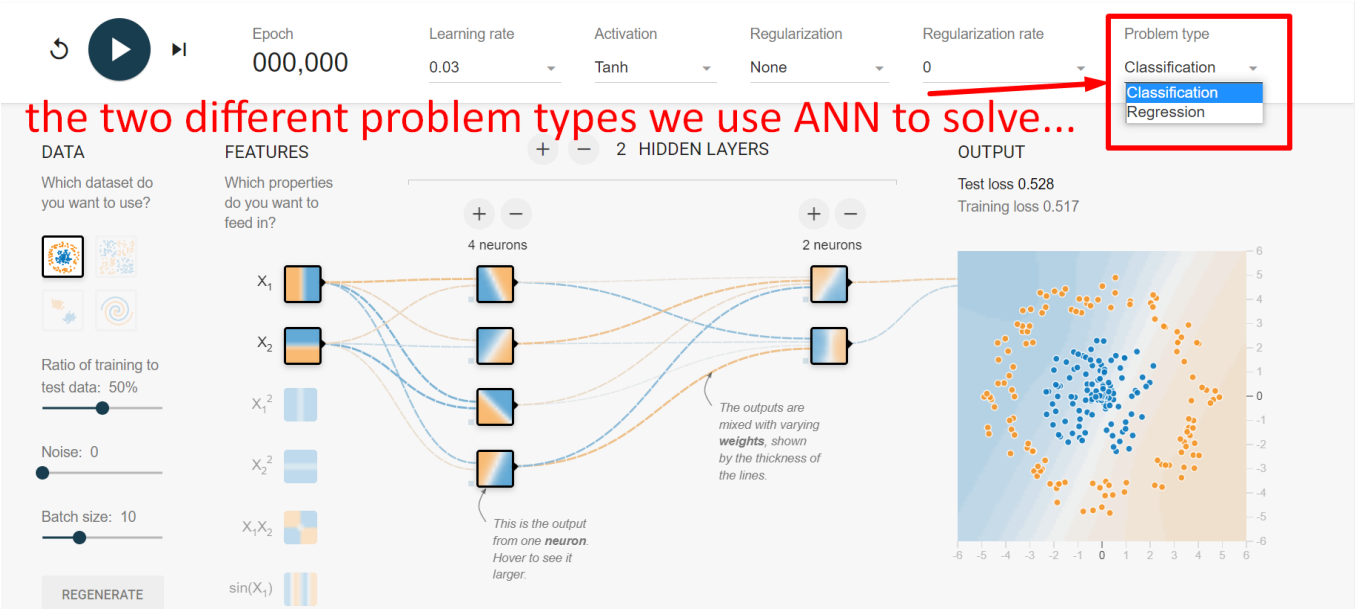
We shall not discuss about CNNs nor RNNs (which is left to other my other manuscripts).

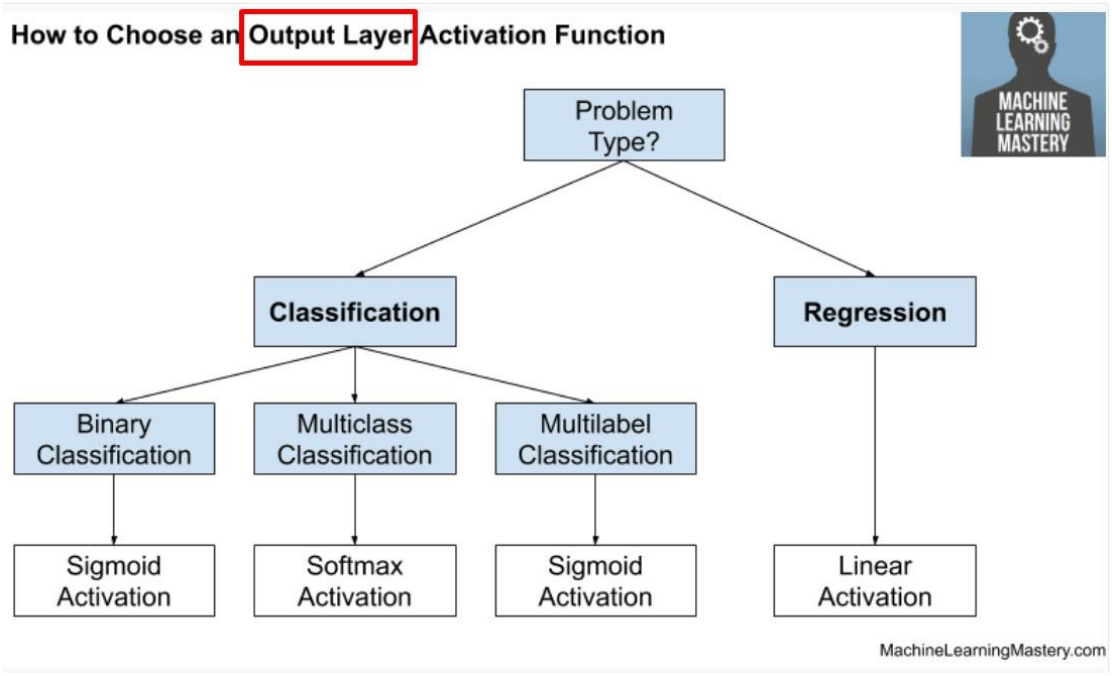


At the moment, we shall simply follow the above charts for which Activation Functions to choose.

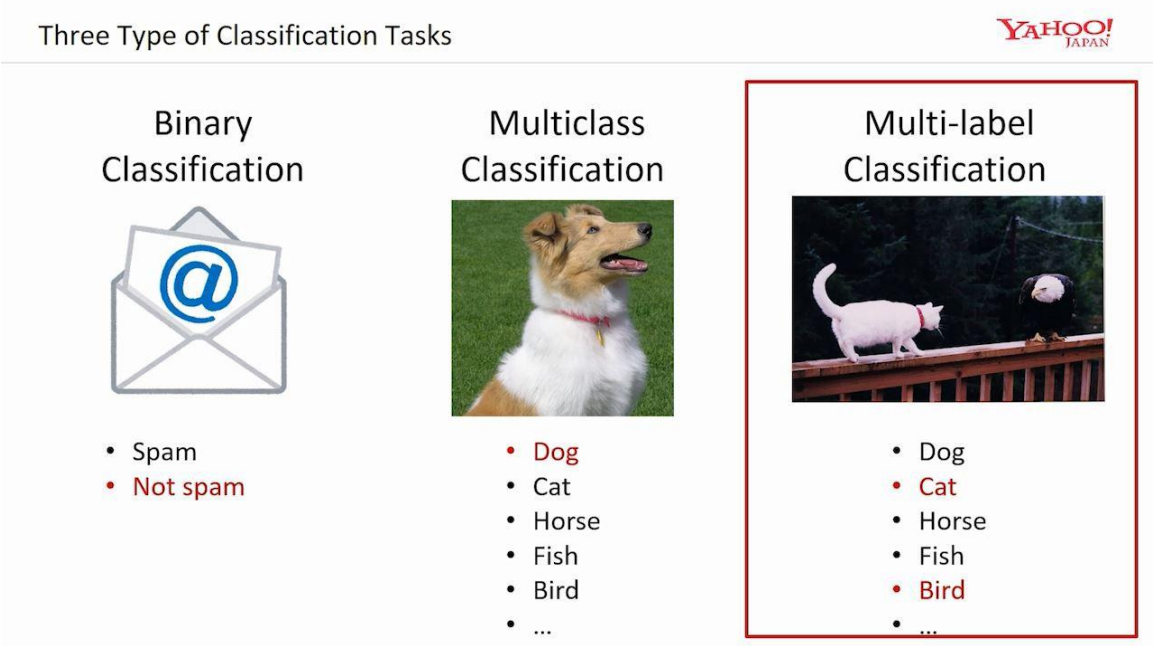
We shall not go in depth as to why we choose either of them.

2. FOR OUTPUT LAYER ACTIVATION, IT DEPENDS ON YOUR PROBLEM TYPE





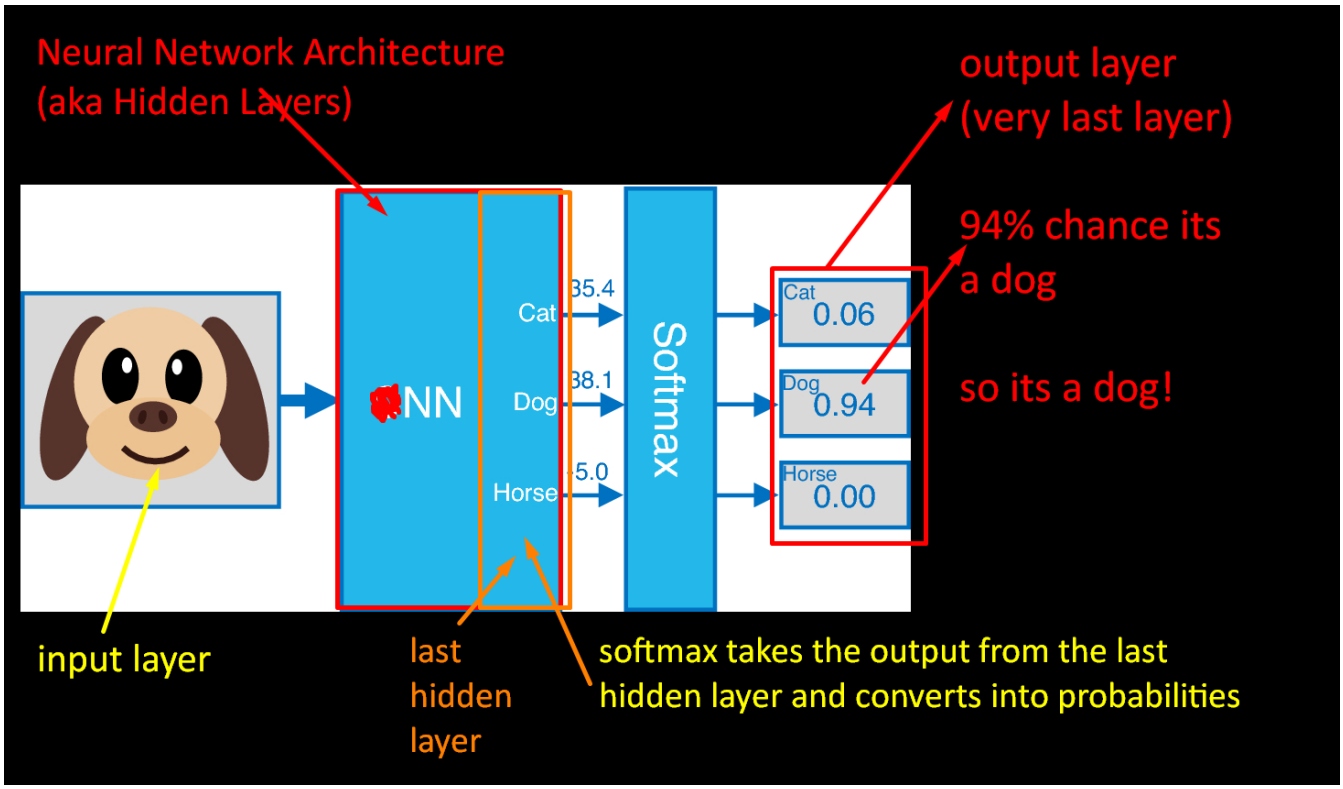
3. WHAT'S THE DIFFERENCE BETWEEN BINARY VS MULTICLASS VS MULTI-LABEL CLASSIFICATION?



Multiclass means 1 picture with 1 animal…. Predict 1 animal….DOG

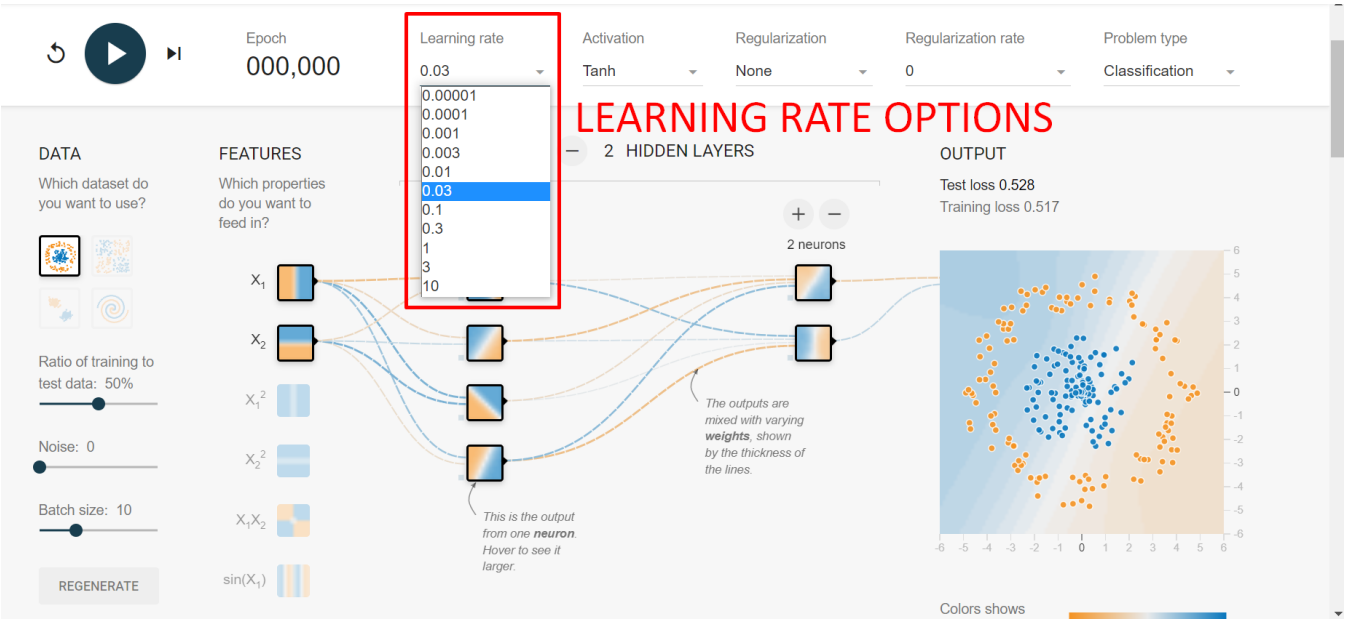
Multilabel means 1 picture with 2 animals… Predict the 2 animals….CAT and BIRD

C. ANOTHER SPECIAL TYPE OF ACTIVATION FUNCTION: SOFTMAX



- Softmax is used for CLASSIFICATION only.
- More about Softmax here: https://keras.io/api/layers/activation_layers/softmax/

VII. HYPERPARAMETER 3: LEARNING RATE



There is no way to determine the optimal Learning Rate. It is by trial and error.

The Learning Rate is akin to how fast a "brain" learns.

The larger the learning rate, the faster the "brain" is forced to learn. But this may mean overstepping the minimal point (we shall see later) or making mistakes. An example could be cram studying the night before exams and not going to sleep. The student does last minute studies to try and recover all lost time.

The smaller the learning rate, the slower the "brain" is allowed to learn. Although it means that the minimal point will surely be found, it might take too long. An example would be a conscientious student studying every day every minute right up till exams. Its too long and painful.

In order to decide the Learning Rate, other concepts need to come in.

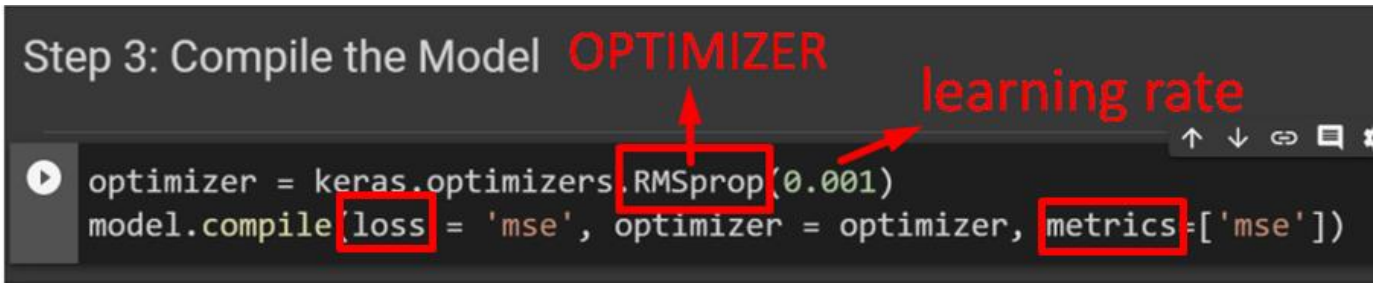
We are unable to see these Hyperparameters:

- Optimizer

- Loss (Error)
- Metrics (Accuracy)

in the Tensorflow Playground, but it is an option in Tensorflow.

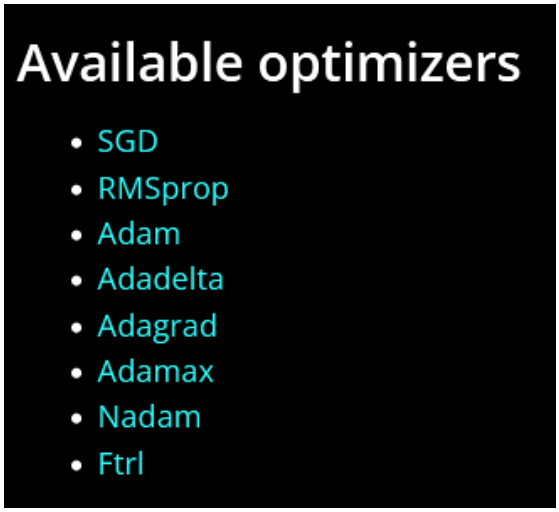
You can see below that the Learning Rate concept ties in with the Optimizer + Loss + Metric concepts.



The above shows a snapshot taken from Tensorflow.

Optimizer is one of the Hyperparameters to be chosen, and with it comes along the Learning Rate.

As Loss (Error) decreases your Metric scores (Accuracy) improve.



https://keras.io/api/optimizers/

Available metrics

Accuracy metrics

- Accuracy class
- BinaryAccuracy class
- CategoricalAccuracy class
- SparseCategoricalAccuracy class
- TopKCategoricalAccuracy class
- SparseTopKCategoricalAccuracy class

Probabilistic metrics

- BinaryCrossentropy class
- CategoricalCrossentropy class
- SparseCategoricalCrossentropy class
- KLDivergence class
- Poisson class

Regression metrics

- MeanSquaredError class
- RootMeanSquaredError class
- MeanAbsoluteError class
- MeanAbsolutePercentageError class
- MeanSquaredLogarithmicError class
- CosineSimilarity class
- LogCoshError class

https://keras.io/api/metrics/

Available losses

Note that all losses are available both via a class handle and via a function handle. The class handles enable you to pass configuration arguments to the constructor (e.g. `loss_fn = CategoricalCrossentropy(from_logits=True)`), and they perform reduction by default when used in a standalone way (see details below).

Probabilistic losses

- BinaryCrossentropy class
- CategoricalCrossentropy class
- SparseCategoricalCrossentropy class
- Poisson class
- binary_crossentropy function
- categorical_crossentropy function
- sparse_categorical_crossentropy function
- poisson function
- KLDivergence class
- kl_divergence function

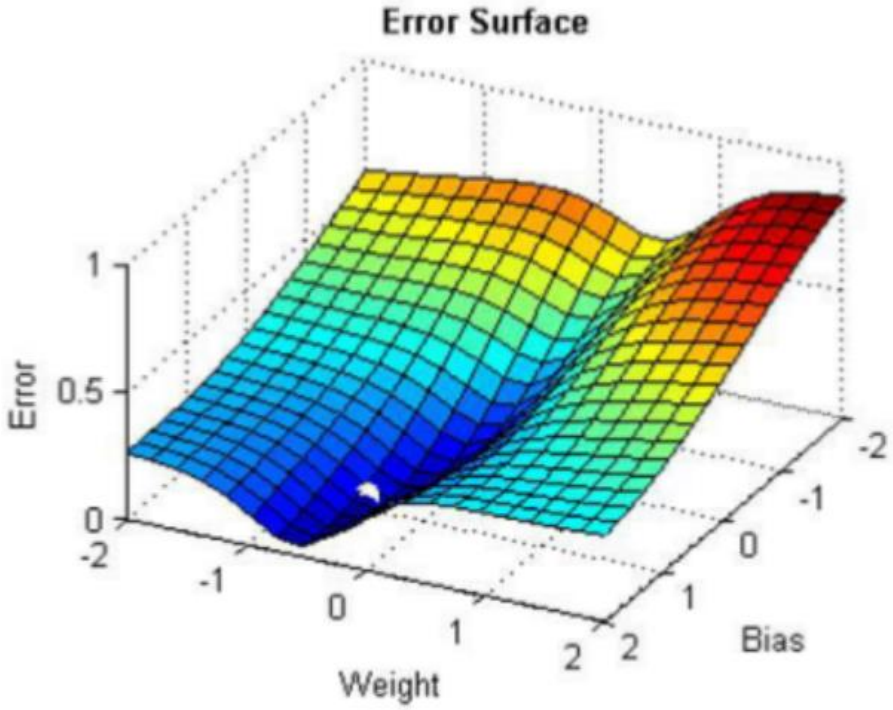
Regression losses

- MeanSquaredError class
- MeanAbsoluteError class
- MeanAbsolutePercentageError class
- MeanSquaredLogarithmicError class
- CosineSimilarity class
- mean_squared_error function
- mean_absolute_error function
- mean_absolute_percentage_error function
- mean_squared_logarithmic_error function
- cosine_similarity function
- Huber class
- huber function
- LogCosh class
- log_cosh function

https://keras.io/api/losses/

A. HYPERPARAMETER 3A: OPTIMIZER

1. HOW THE OPTIMIZER WORKS



You would like the lowest Error. You cannot decide the Weights nor Biases (Parameters).

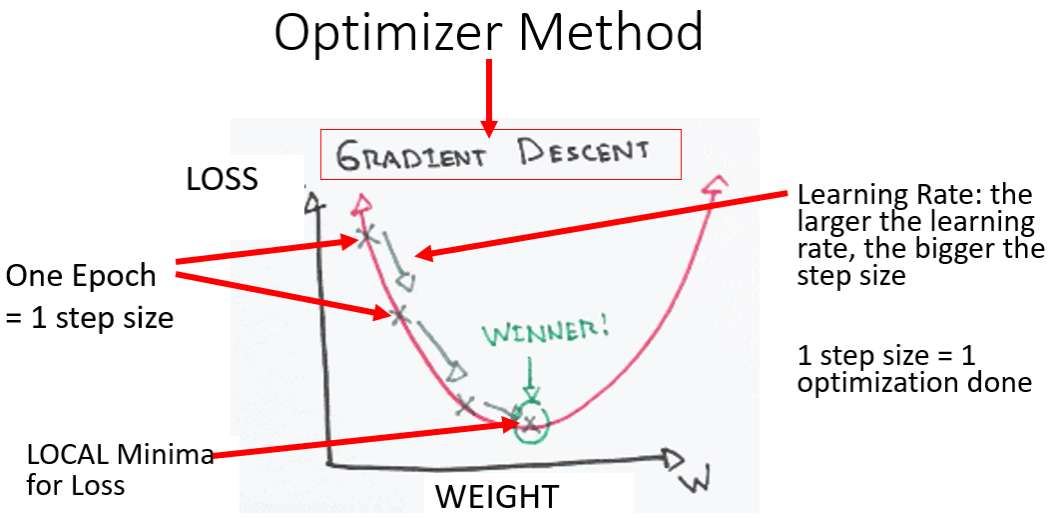
These are decided by the ANN.

But you can decide the "Optimization Model" that helps to reduce Error (or also known as Loss).

Choosing the type of Optimizer is akin to choosing the way you would like to go down the mountain.

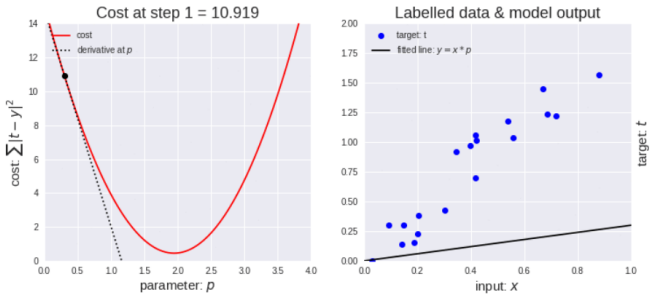
Every Optimizer has its own "style" of descending the mountain, but they all stem from the Gradient Descent method.

You may think of the various Optimization Methods as different people have different styles of studying for exams. Some like to drink coffee… some like to drink red bull… some like to smoke… some like to take power naps… all but to help them study better….



By choosing the appropriate Optimizer, you can achieve the lowest Loss (or Error) at the optimal Weight (decided by the ANN).

The Optimizer will estimate the gradient and adjust the weights, so that with each successive step the Loss is closer to the minimum.

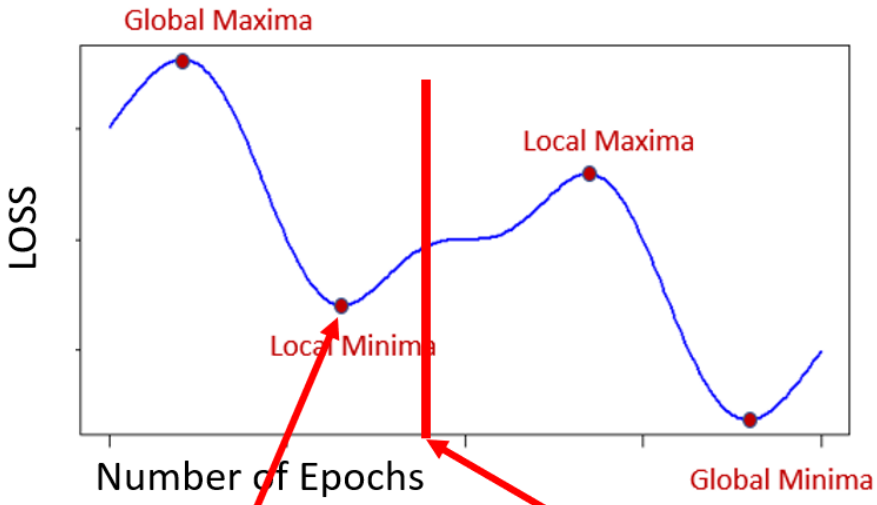


Gradient descent optimisation.

For details, please refer here on Gradient Descent Optimization:

https://medium.com/onfido-tech/machine-learning-101-be2e0a86c96a

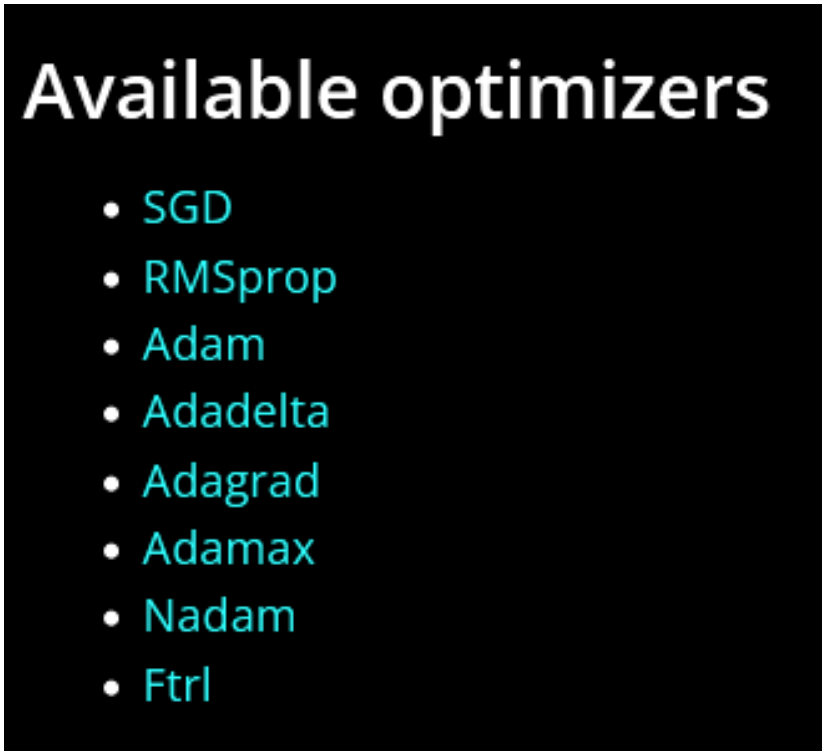
2. NOTE: THE NUMBER OF EPOCHS CHOSEN BY YOU WILL IMPACT THE LOSS



If your no. of Epochs set by you stop here, your Optimizer can only adjust the Weights to help you find the Local Minima (not Global Minima)

3. VARIOUS TYPES OF OPTIMIZERS

There are various type of Optimizers to choose from:



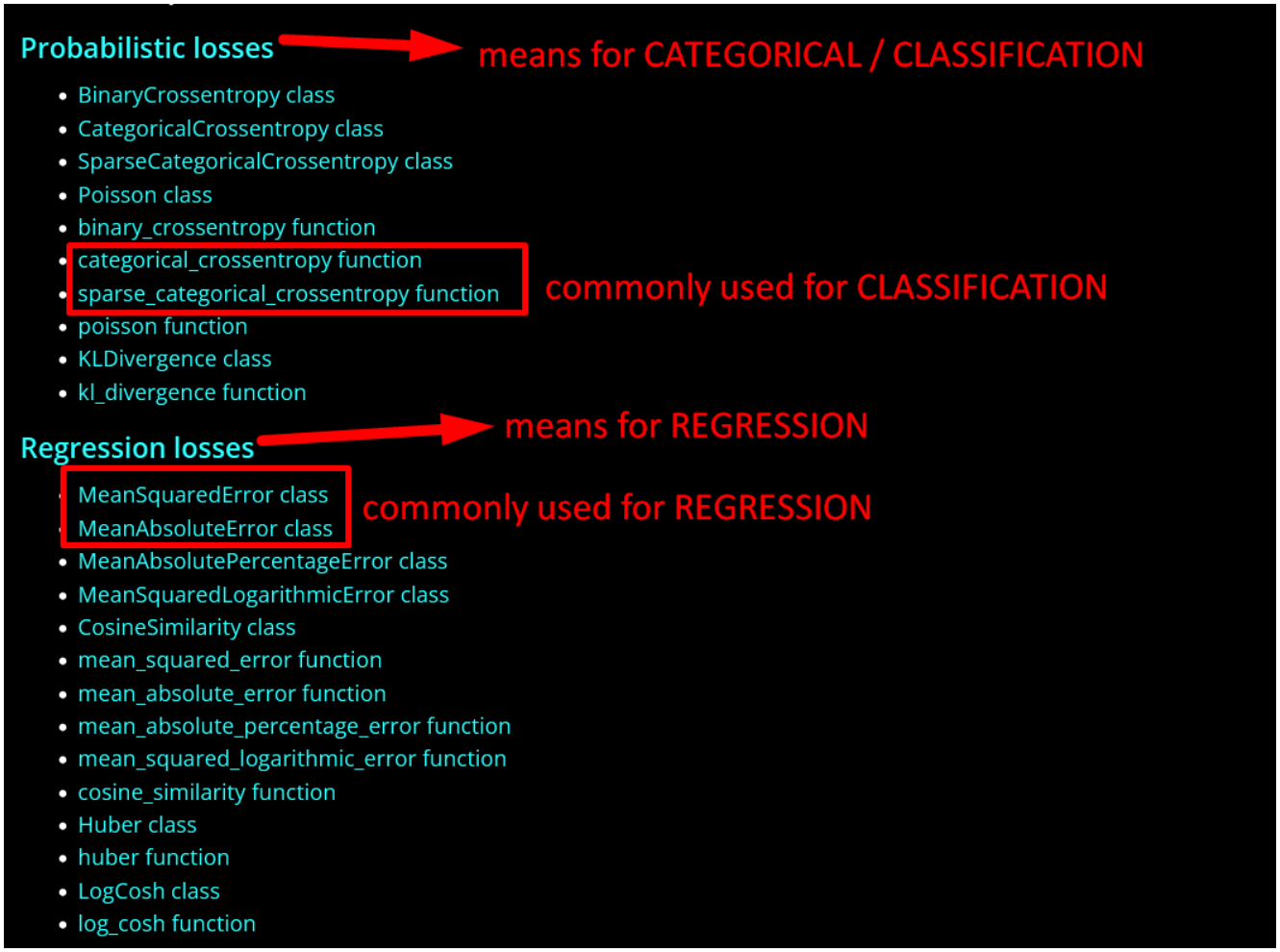
- All optimizers are found here: https://keras.io/api/optimizers/
- Most common are:
    - SGD is unstable[2]
    - RMSprop → for REGRESSION
    - Adam → for CLASSIFICATION
    - Adam is the BEST OPTIMIZER[3]

[2] https://proceedings.neurips.cc/paper/2020/file/f3f27a324736617f20abbf2ffd806f6d-Paper.pdf

[3] https://towardsdatascience.com/optimizers-for-training-neural-network-59450d71caf6

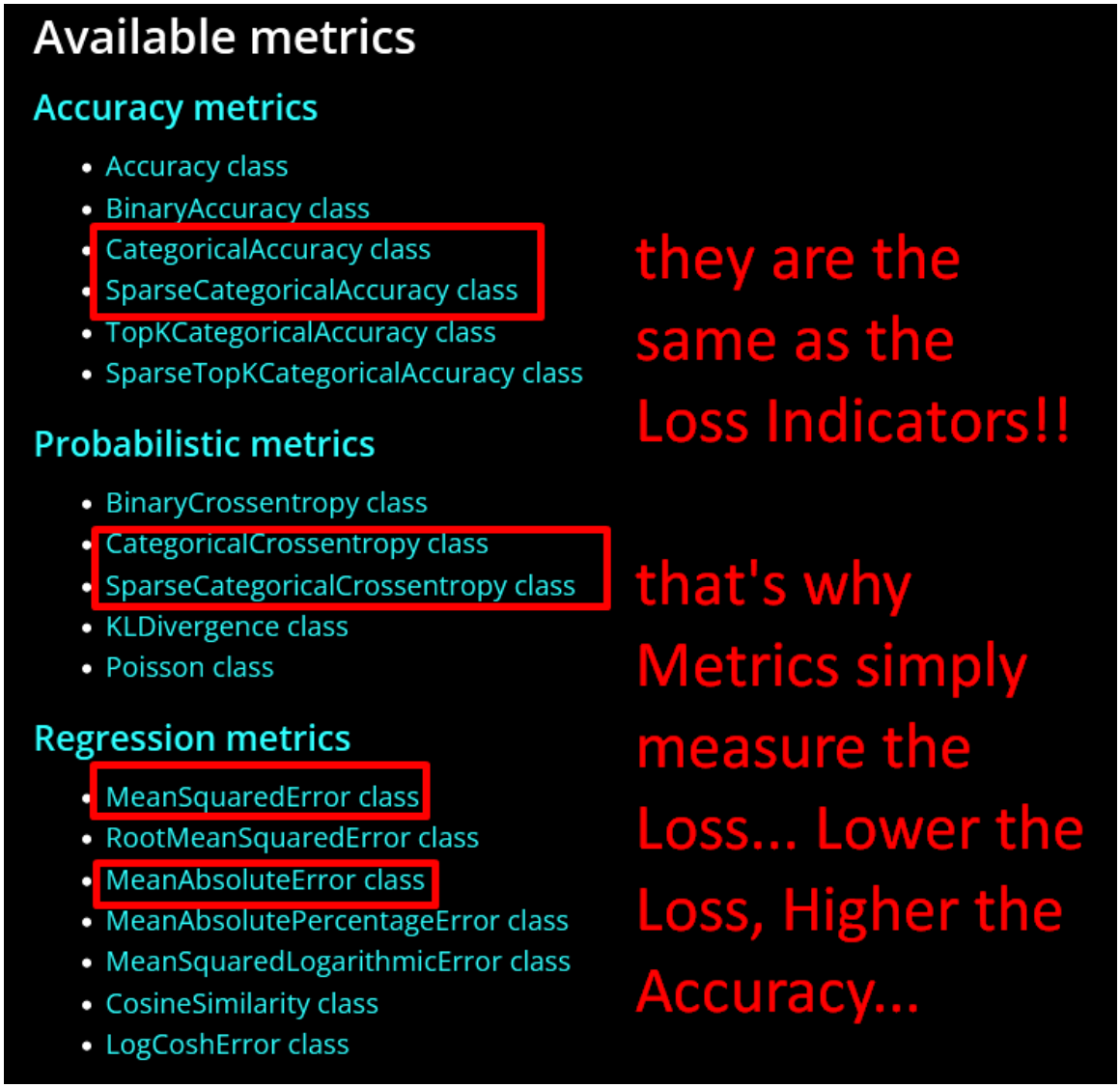
B. HYPERPARAMETER 3B: LOSS METRICS

https://keras.io/api/losses/



C. HYPERPARAMETER 3C: ACCURACY METRICS

All metrics can be found here: https://keras.io/api/metrics/



VIII. WHICH HYPERPARAMETER TO CHOOSE???

Problem Type	Last-layer Output Nodes	Hidden-layer activation	Last-layer activation	Loss function
Binary classification	1	RELU (first choice), Tanh (for RNNs)	Sigmoid	Binary Crossentropy
Multi-class, single-label classification	Number of classes		Softmax	Categorical Crossentropy
Multi-class, multi-label classification	Number of classes		Sigmoid (one for each class)	Binary Crossentropy
Regression to arbitrary values	1		None	MSE
Regression to values between 0 and 1	1		Sigmoid	MSE/Binary Crossentropy

For REGRESSION, Output Layer only constists of 1 neuron (because like the price of a house is $1 million.. or $1.5 million….. just one output).

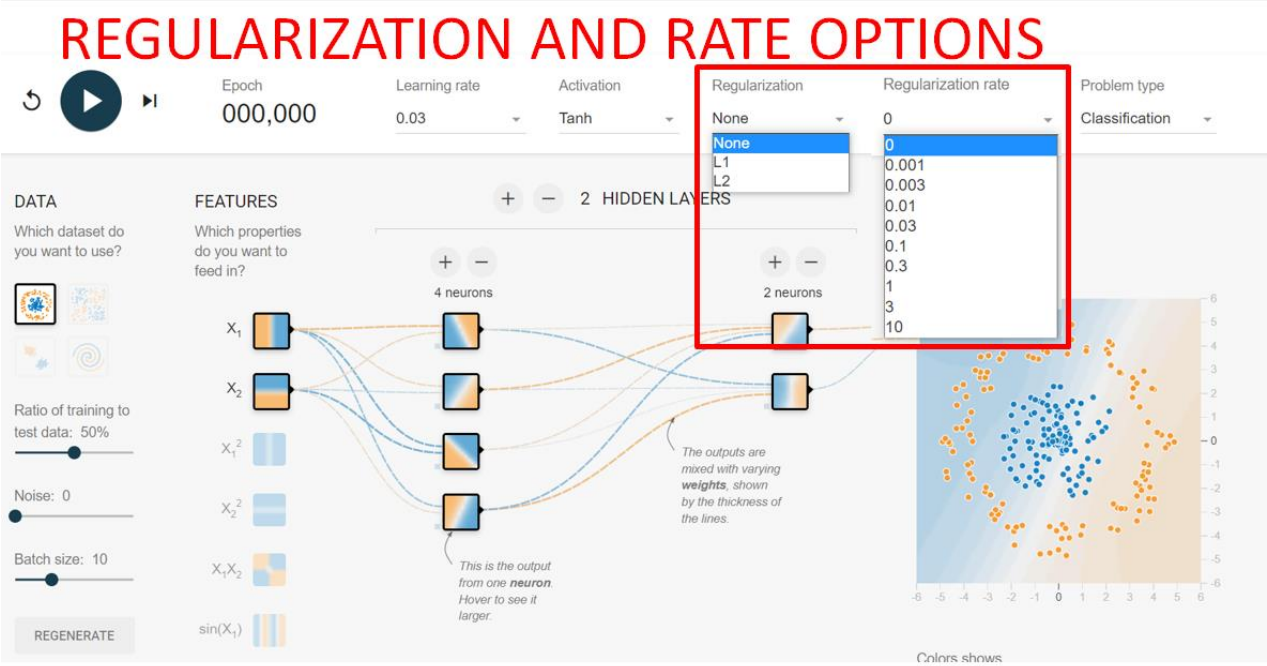
For CLASSIFICATION, Ouput Layer consists of as many neurons as there are CATEGORIES. E.g.

Dog or Cat? 2 Categories = 2 Neurons

Which type of T-Shirt? 10 Categories of T-Shirts = 10 Neurons

Above is taken from

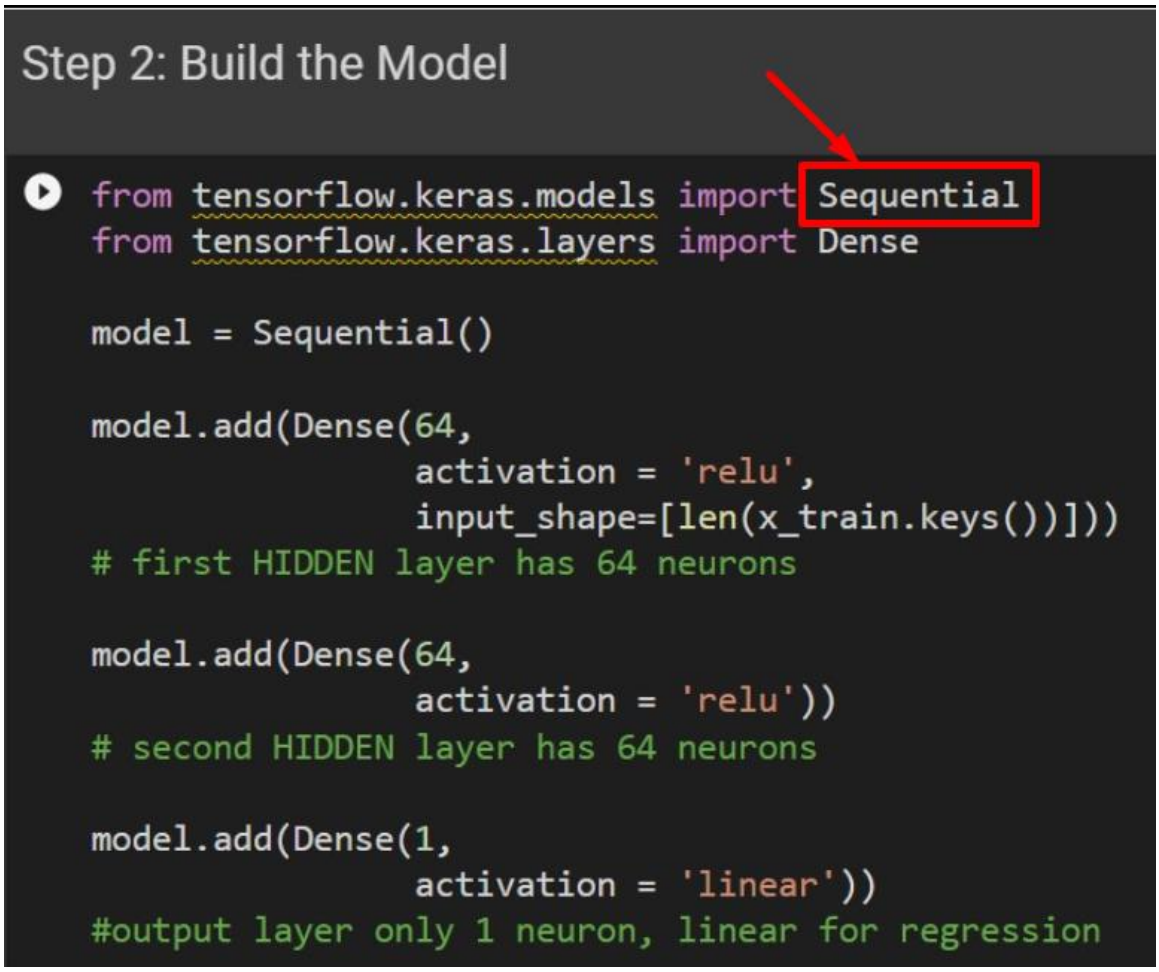
https://medium.com/analytics-vidhya/activation-functions-and-loss-functions-for-neural-networks-how-to-pick-the-right-one-542e1dd523e0



This manuscript will not dwell into Regulariztion and Regularization Rate options.

In short, they are just options to penalize or control the model frrom overfitting by reducing the power of the function.

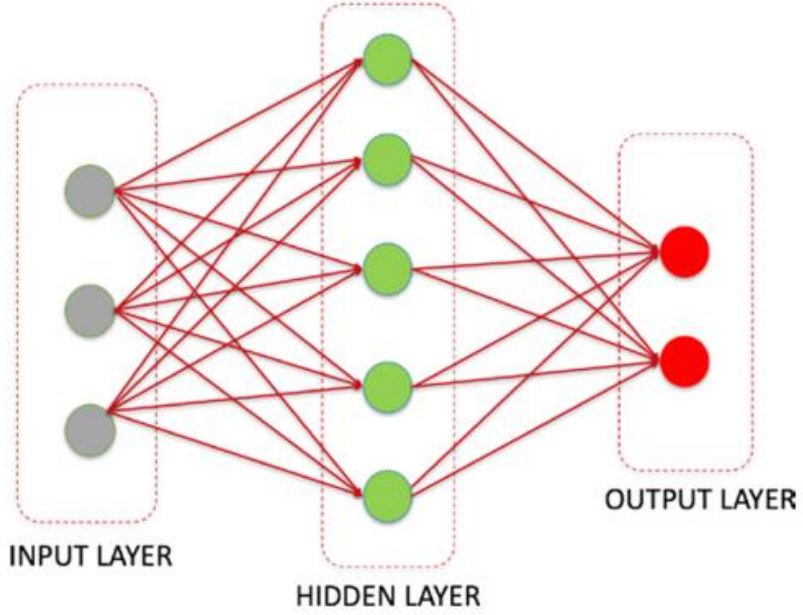
More details here: https://www.alvinang.sg/s/L1-Lasso-and-L2-Ridge-and-Elastic-Net-Regression-using-Python-by-Dr-Alvin-Ang.pdf



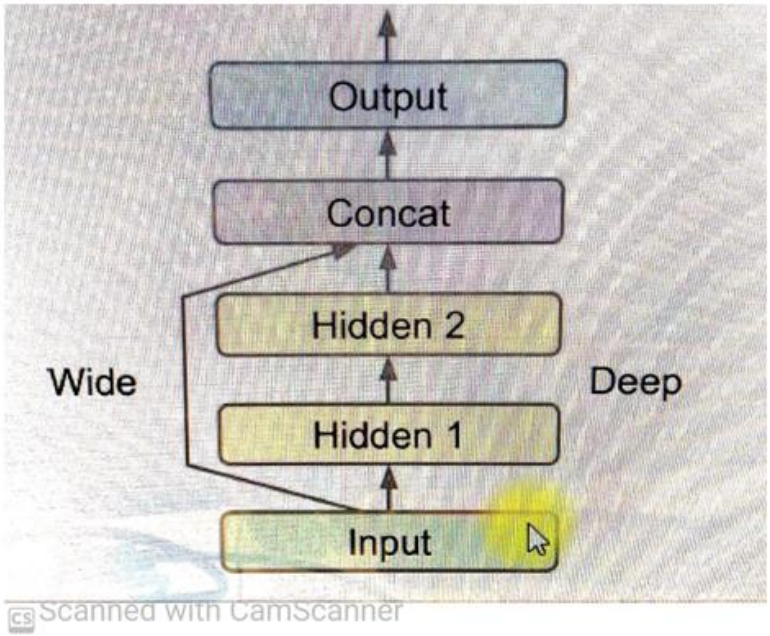
You can't see "Sequential" vs "Functional" Hyperparameter in Tensorflow Playground.

You can only see it in Tensorflow (as above).

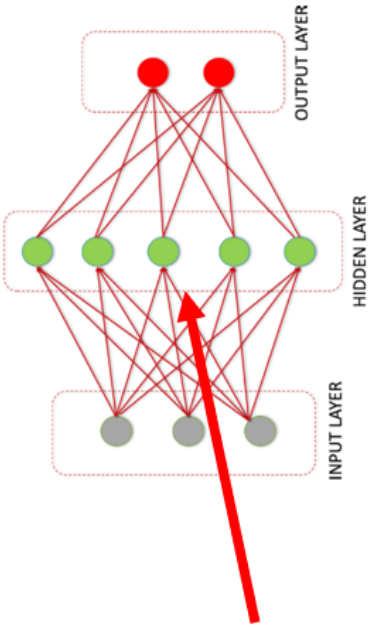
Sequential



Functional



Sequential



This functional NN is also known as the Wide and Deep NN

Is called functional because each block is a function... Where each function could be one NN block itself...

Has a complex structure topology

Deep because its going through 2 deep dense NN

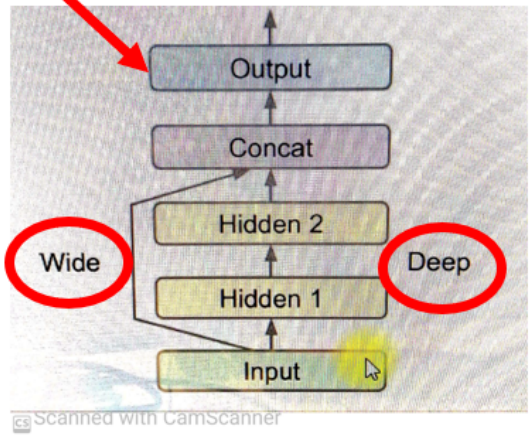
Wide because is bypassing both of them as well

In other words, the functional NN is able to learn deep patterns because the Data has to pass thru the 2 Deep Hidden NN

Yet at the same time, the Wide linkage bypasses the 2 Hidden NN Layers thus keeping the model simple.

Thus its able to learn both Simple yet Complex patterns from the Data.

Functional



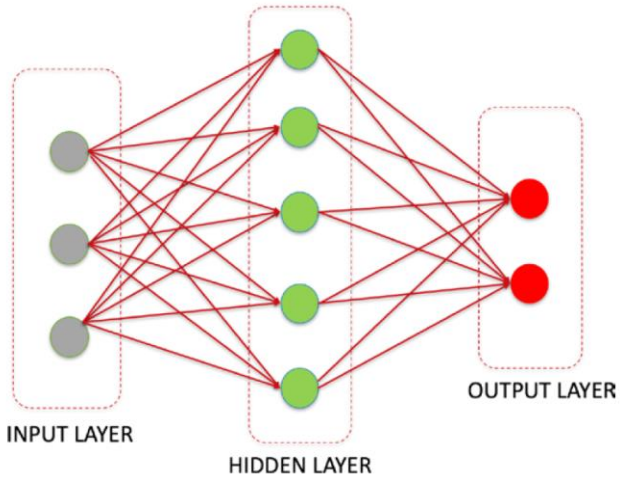
On the other hand, Data in the Sequential model has no choice but to pass through the 2 dense transformations...this will distort simple patterns in the data because there is no way round it... hence might distort accuracy....

Step 2: Build the Model

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

model = Sequential()

model.add(Dense(64,
                activation = 'relu',
                input_shape=[len(x_train.keys())]))
# first HIDDEN layer has 64 neurons

model.add(Dense(64,
                activation = 'relu'))
# second HIDDEN layer has 64 neurons
```

You can't see "DENSE" Hyperparameter in Tensorflow Playground.

You can only see it in Tensorflow (as above).



DENSE means Fully Connected, which means every single node in one layer is fully connected / outputted to every other node in the next layer.

XII. REFERENCES

1. https://www.amazon.com/Learn-PySpark-Python-based-Machine-Learning/dp/1484249607
2. https://www.udemy.com/course/cnn-for-computer-vision-with-keras-and-tensorflow-in-python
3. https://www.nature.com/articles/s41598-021-94691-7
4. https://www.baeldung.com/cs/epoch-neural-networks
5. https://www.xpertup.com/blog/deep-learning/batch-size-vs-epoch-vs-iteration/
6. https://towardsdatascience.com/epoch-vs-iterations-vs-batch-size-4dfb9c7ce9c9
7. https://medium.com/onfido-tech/machine-learning-101-be2e0a86c96a
8. https://www.geeksforgeeks.org/activation-functions-neural-networks/
9. https://stats.stackexchange.com/questions/541700/why-are-non-linear-activation-functions-required-in-multilayer-perceptron-classi
10. https://medium.com/mlearning-ai/a-little-about-perceptrons-and-activation-functions-aed19d672656
11. https://medium.com/swlh/pysparks-multi-layer-perceptron-classifier-on-iris-dataset-dcf70d553cd8
12. https://towardsdatascience.com/deep-neural-multilayer-perceptron-mlp-with-scikit-learn-2698e77155e
13. https://towardsdatascience.com/optimizers-for-training-neural-network-59450d71caf6
14. https://proceedings.neurips.cc/paper/2020/file/f3f27a324736617f20abbf2ffd806f6d-Paper.pdf
15. https://medium.com/analytics-vidhya/activation-functions-and-loss-functions-for-neural-networks-how-to-pick-the-right-one-542e1dd523e0
16. https://vileoy.uovie.com/blog/2020/05/26/loss-functions/

XIII. APPENDIX A: WHY DO WE NEED AN ACTIVATION FUNCTION?

A. 1ST REASON: TO ENABLE BACK PROPAGATION

- Without Activation Function, you can't do Back Propagation.
- The NN won't work.
- Back Propagation is a heavy mathematical technique that will not be covered here.

B. 2ND REASON: CERTAIN ACTIVATION FUNCTIONS HELP TO MODEL NONLINEARITY

- An example would be the Sigmoid Activation Function, where it outputs not just a 0 or 1 answer (a 0 or 1 would be a Step Function).
- Sigmoid function helps Classify Images in terms of Probability (between 0 and 1).
- E.g. We may predict the photo to be 33% Crow and 66% Cat (you can't be entirely sure if it's a Crow or Cat through a blurred image).
- The NN needs to model non-linearity within the data[4] but without a Non Linear Activation Function, it is essentially just a linear regression model.[5]
- You can't do this with a Linear / Step Function[6]

[4] https://stats.stackexchange.com/questions/541700/why-are-non-linear-activation-functions-required-in-multilayer-perceptron-classi

[5] https://www.geeksforgeeks.org/activation-functions-neural-networks/

[6] https://blog.roboflow.com/activation-function-computer-vision/

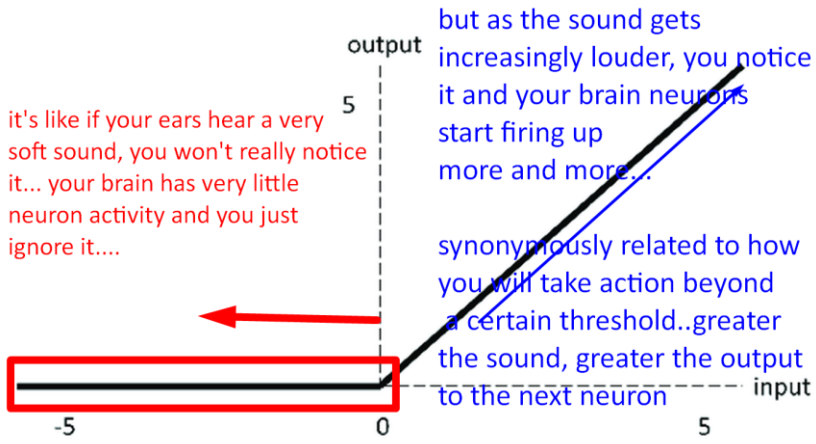
XIV. APPENDIX B: WHY IS THE RELU FUNCTION MOST POPULAR?

A. SHORT SIMPLE ANSWER

- In the past, researcher have tried Softmax and they didn't get good results.
- So they invented tanh to improve it. But likewise, it still didn't work well.
- Alex Net made use of the ReLU and it became very popular ever since which was a breakthrough for NN.

B. IT MIMICS THE HUMAN NEURONS CLOSER

- You may google "Biological Plausibility of ReLU" for more details.

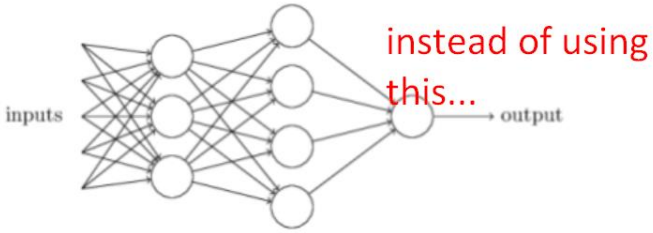


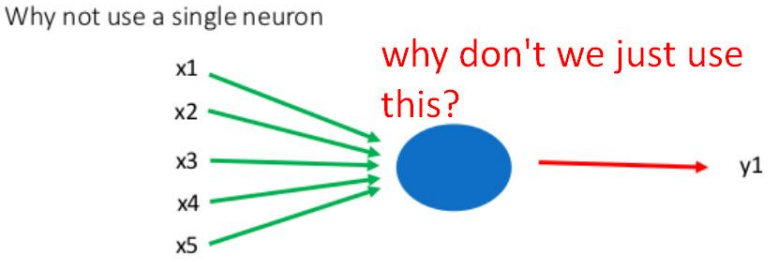
C. RELU RESOLVED THE VANISHING GRADIENT PROBLEM

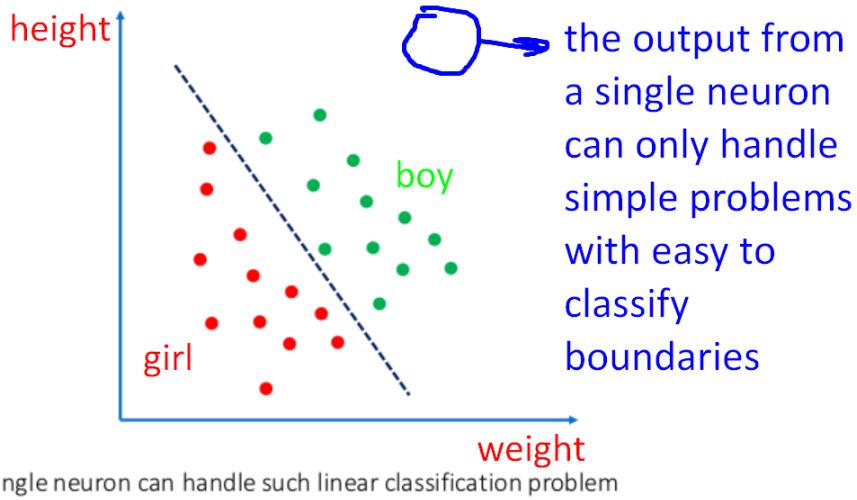
- Softmax and Tanh had a Vanishing Gradient problem, in which during back propagation, those near the inputs almost received back zero outputs… meaning those neurons near the start were as good as dead because Softmax and Tanh kept differentiating until it almost reached zero.
- But ReLU just simply killed off those neurons that received zero outputs.
- Vanishing Gradient is beyond the scope here.

XV. APPENDIX C: WHY DO WE NEED TO STACK NEURONS?

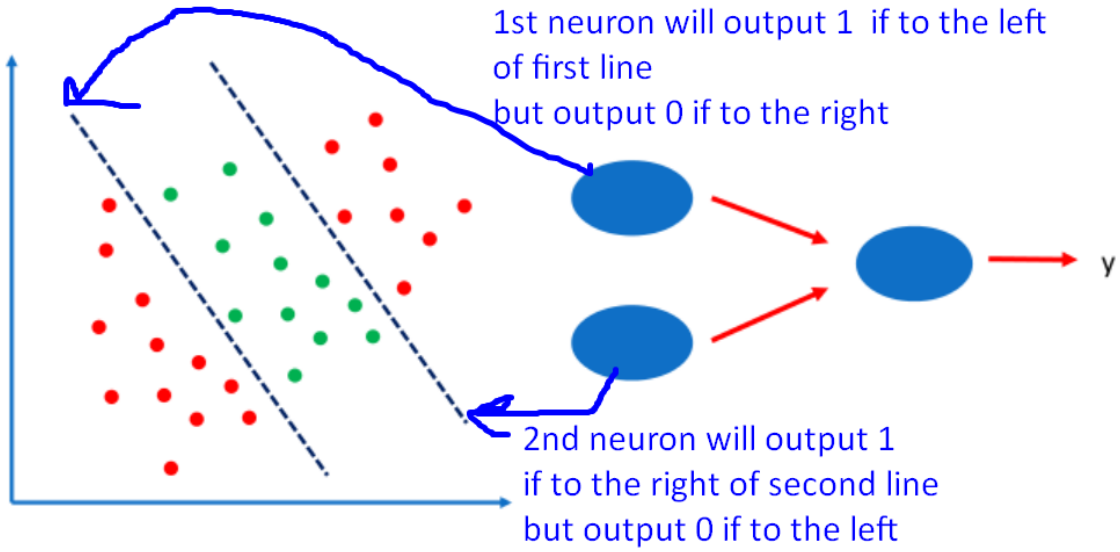
A. WHY CAN'T WE JUST USE 1 PERCEPTRON TO DO EVERYTHING?







The above picture shows the classification boundary of a simple single 1 perceptron model.



Each neuron can focus on the particular features of the object instead of the final outcome

- The picture above shows a NN with 2 stacked neurons.
- The final output neuron will fire 1 as long as any of the previous neurons fire.
- Such complex classifications cannot be handled by a single perceptron.
- Thus, you need to stack them to model very complicated data.

Loss function	Usage	Examples	
		Using probabilities *from_logits=False*	Using logits *from_logits=True*
BinaryCrossentropy	Binary classification	y_true: 1 y_pred: 0.69	y_true: 1 y_pred: 0.8
CategoricalCrossentropy	Multiclass classification	y_true: 0 0 1 y_pred: 0.30 0.15 0.55	y_true: 0 0 1 y_pred: 1.5 0.8 2.1
Sparse CategoricalCrossentropy	Multiclass classification	y_true: 2 y_pred: 0.30 0.15 0.55	y_true: 2 y_pred: 1.5 0.8 2.1

- Use Binary CE when my ***inputs (y_true) and outputs (y_pred)*** are Binary (0 or 1)
- Use Categorical CE when my ***inputs (y_true) and outputs (y_pred)*** are "One Hot Encoded"
- Use Sparse Categorical CE when my ***inputs (y_true) and outputs (y_pred)*** are Integers

If your targets are **one-hot encoded**, use `categorical_crossentropy`.

- Examples of one-hot encodings:
  - `[1,0,0]`
  - `[0,1,0]`
  - `[0,0,1]`

But if your targets are **integers**, use `sparse_categorical_crossentropy`.

- Examples of integer encodings (*for the sake of completion*):
  - `1`
  - `2`
  - `3`

A. WHAT DO I MEAN BY "INPUT AND OUTPUT"?

- For Binary CE
    - E.g. I feed my NN only with Cats and Dogs (binary 0 for Dogs and 1 for Cats) → input (y_true as shown in the above picture).
    - After training, my NN will be able to output a prediction y_pred (only 0 or 1).
- For Categorical CE
    - E.g. I feed my NN with "One Hot Encoded" jpgs of 0 to 9
    - After training, my NN will be able to output a prediction of y_pred (One Hot Encoded Format as below)

B. WHAT IS "ONE HOT ENCODING"?

```
0    [1 0 0 0 0 0 0 0 0 0]
1    [0 1 0 0 0 0 0 0 0 0]
2    [0 0 1 0 0 0 0 0 0 0]
3    [0 0 0 1 0 0 0 0 0 0]
4    [0 0 0 0 1 0 0 0 0 0]
5    [0 0 0 0 0 1 0 0 0 0]
6    [0 0 0 0 0 0 1 0 0 0]
7    [0 0 0 0 0 0 0 1 0 0]
8    [0 0 0 0 0 0 0 0 1 0]
9    [0 0 0 0 0 0 0 0 0 1]
```

- For Sparse Categorical CE
    - E.g. I feed my NN with jpgs of pictures with an Integer to represent a label
    - (e.g. 1 for shirt, 2 for shorts, 3 for socks)
    - After training, my NN will be able to output a prediction of y_pred of 1 / 2 / 3
    - (my NN will be able to tell me the new picture is a shirt / shorts / socks.

Dr. Alvin Ang earned his Ph.D., Masters and Bachelor degrees from NTU, Singapore. He is a scientist, entrepreneur, as well as a personal/business advisor. More about him at www.AlvinAng.sg.