Deep Dive into Neural Networks

Agenda

Discussion about ANN and it's components

Feedforward neural network

Loss function

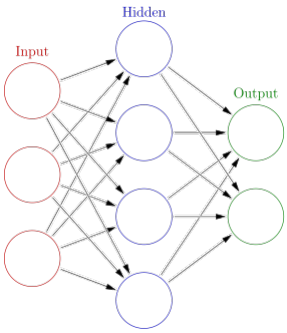
Gradient Descent and Backpropagation

Hyperparameter discussion

Write our first ANN

Neural Network

https://www.youtube.com/watch?v=aircAruvnKk&list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi



This is the basic structure of a neural network

Each circle represents a node

Each arrow represents a connection called weights which gives an idea how strong the connection is

Input nodes pass the input data

So if a dataset has n input features it will have n input nodes

In [138]:

```
df = pd.read_csv('../../data/training_dataset_500.csv')

df[df['House']==1]
```

Out[138]:

	ID	Label	House	Year	Month	Temperature	Daylight	EnergyProduction
0	0	0	1	2011	7	26.2	178.9	740
1	1	1	1	2011	8	25.8	169.7	731
2	2	2	1	2011	9	22.8	170.2	694
3	3	3	1	2011	10	16.4	169.1	688
4	4	4	1	2011	11	11.4	169.1	650
5	5	5	1	2011	12	4.2	199.5	763
6	6	6	1	2012	1	1.8	203.1	765
7	7	7	1	2012	2	2.8	178.2	706
8	8	8	1	2012	3	6.7	172.7	788
9	9	9	1	2012	4	12.6	182.2	831
10	10	10	1	2012	5	17.6	214.2	955
11	11	11	1	2012	6	20.8	143.0	837

F1, F2, F3, F4, F5 -> F6

So in case of a dataframe like this we will have 3 input nodes

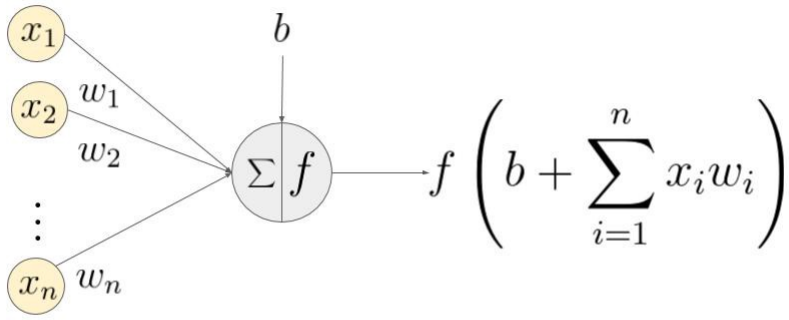
Output nodes represent the number of values a target node can take

So it it's a iris classification problem we can have 4 outputs since there are 4 categories of iris

In between the input and output nodes there can be any number of layers with nodes called hidden layers.

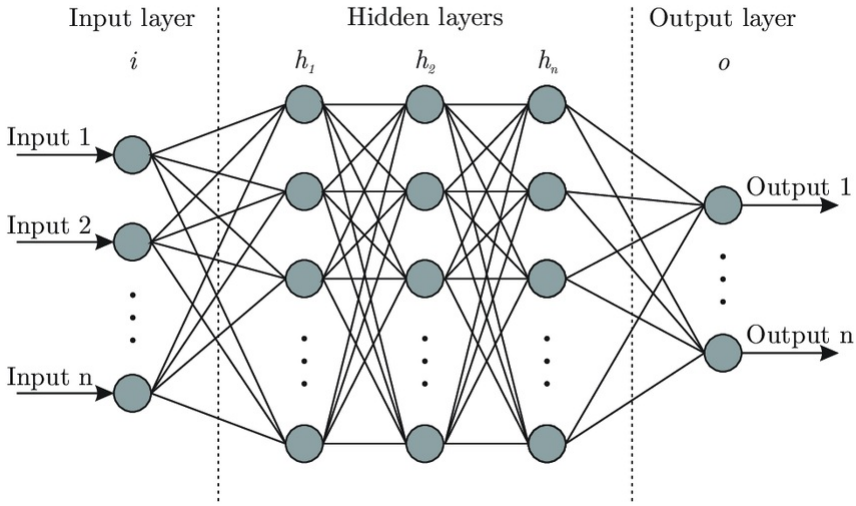
Input nodes and output nodes are fixed but the number of hidden layers and hidden nodes is a hyperparameter.

Weights and bias



An example of a neuron showing the input (x_1 - x_n), their corresponding weights (w_1 - w_n), a bias (b) and the activation function f applied to the weighted sum of the inputs.

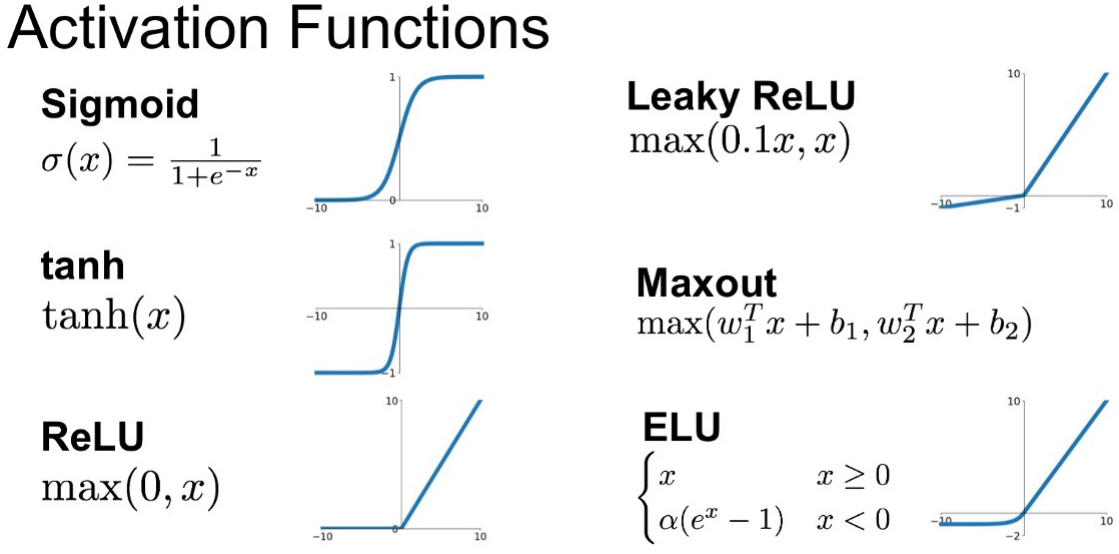
Multi-layered neural network



Activation Function

Used to introduce non-linearity

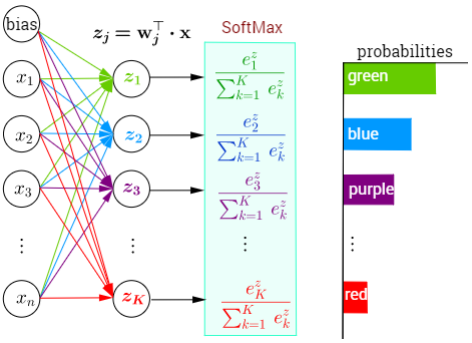
y = x1^2 + x2^3

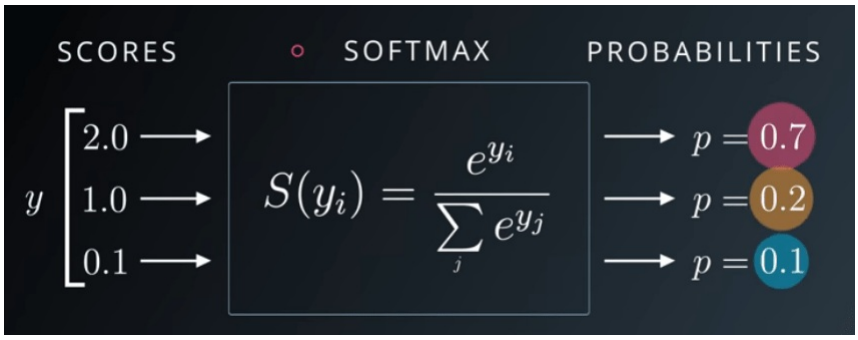


Activation function does two things :-

1. Add non-linearity
2. Lights up or kills a neuron

Softmax function





S(2) = e^2/(e^2+e^1+e^0.1)

S(1) = e^1/(e^2+e^1+e^0.1)

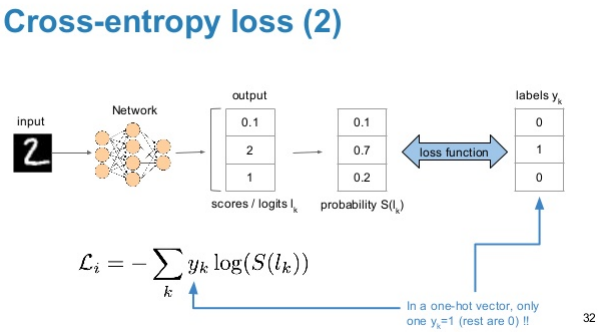
S(0.1) = e^0.1/(e^2+e^1+e^0.1)

S(2) + S(1) + S(0.1) = 1

Linear -> MSE

Logistic -> Cross-Entropy

Cross-entropy loss



L = - ((0 * log(0.1)) + (1 * log(0.7) + (0 * log(0.2)))

Why cross-entropy and not mean square loss

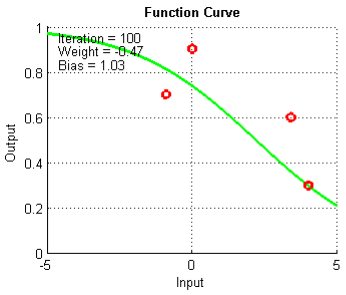
Gradient Descent

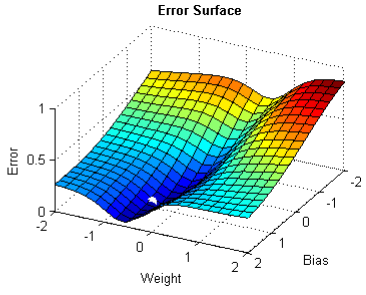
Neural network does forward propagation to give a certain output

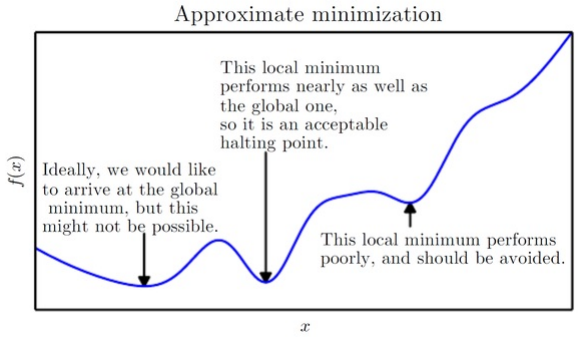
We then see how much the output differs from expect output using the cost/loss function

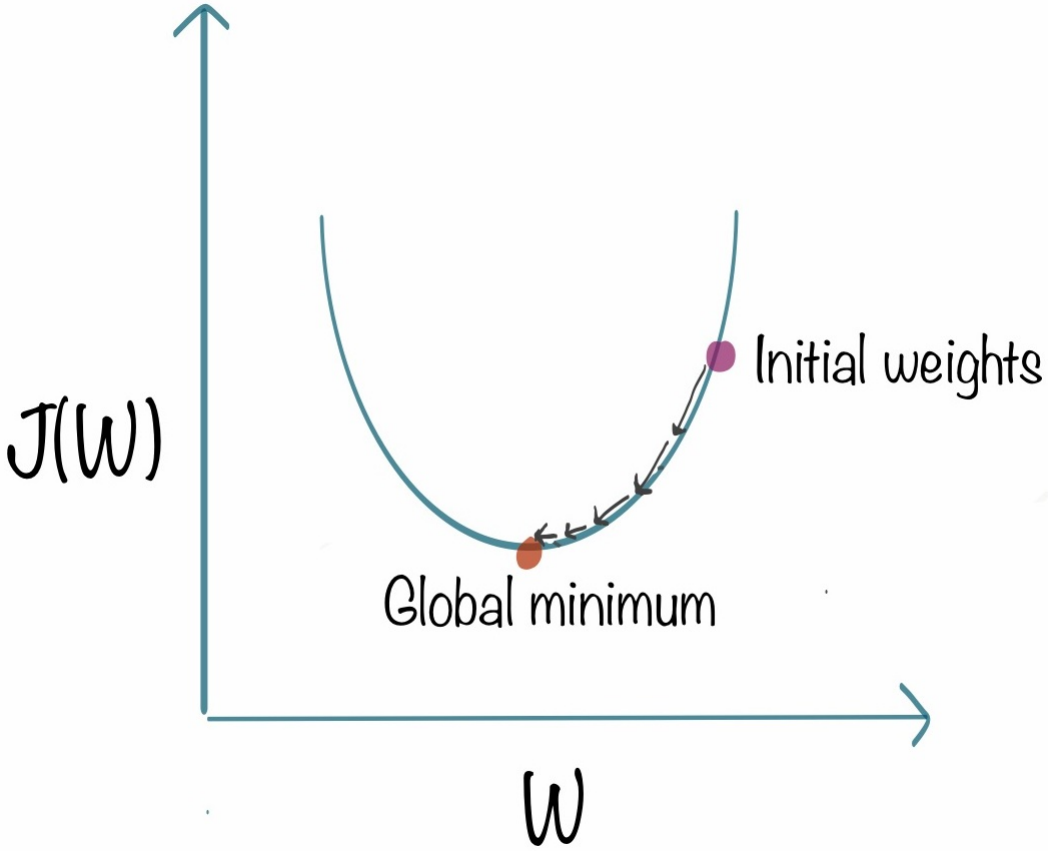
Our goal is to reduce the loss function and get the global minima using the weights and biases

To do this start with random set of weights and gradually nudge them to reach the local minima









https://www.youtube.com/watch?v=IHZwWFHWa-w&list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi&index=2

Gradient Descent

Remember that the general form of gradient descent is:

$$
\begin{aligned}
&Repeat\ \{ \\
&\quad \theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta) \\
&\}
\end{aligned}
$$

We can work out the derivative part using calculus to get:

$$
\begin{aligned}
&Repeat\ \{ \\
&\quad \theta_j := \theta_j - \frac{\alpha}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)} \\
&\}
\end{aligned}
$$

Backpropagation

The output here vastly differs from expected

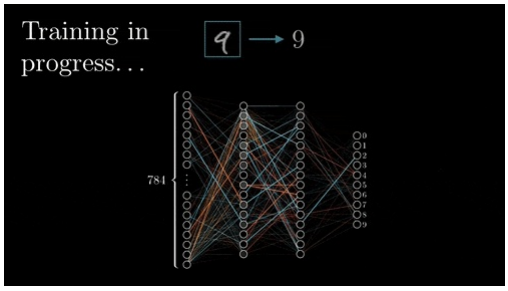
This is the output we want

To achieve this we need to push up the activation at 2 to 1 and push the rest down to 0

3 ways to change to output

1. Change bias
2. Change weights
3. Change activations

Cumulative sum of changes. Recursively apply the process and move backwards and hence backpropagation

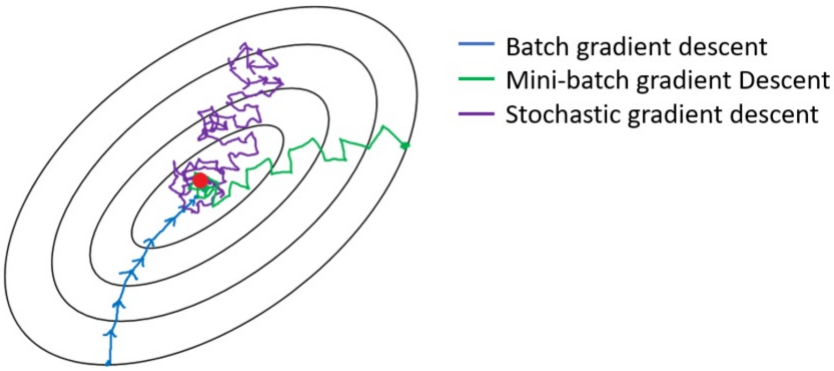


https://www.youtube.com/watch?v=Ilg3gGewQ5U&list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi&index=3

Training concepts

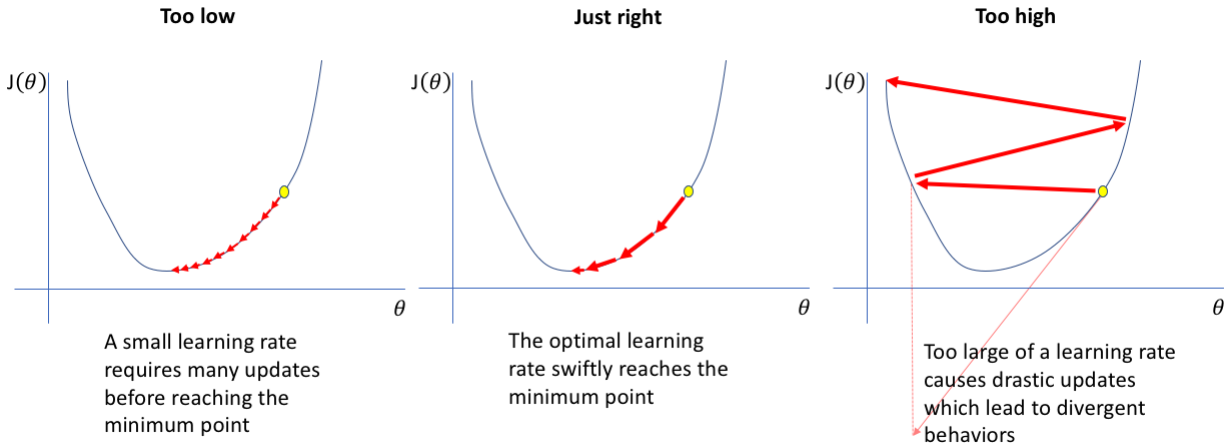
Batch

Batch is a set of data points



Learning rate

Pace at which you would want to train



Batch size and epoch

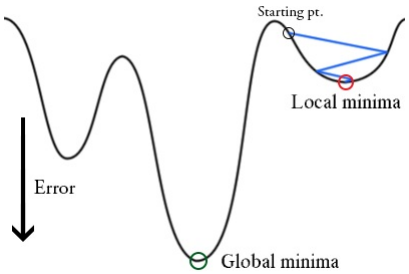
Batch size is number of data points present in a single training step i.e iteration

An epoch is when all the examples in the training dataset is visited once

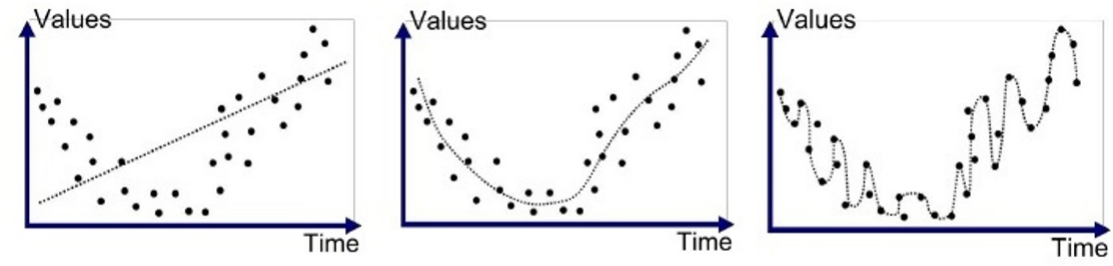
Difference between neural networks and other things you saw until now

Neural networks global minima might look like below, so you need to train more to reach global minima

If batch gd is used can get stuck in local minima hence mini batch gd is used



Underfitting and overfitting



Underfitted Good Fit/Robust Overfitted

Underfitting is battled by increasing the capacity of network

Overfitting is battled by adding regularisation techniques

28x28

Input normalization

Keep pixel values between 0-1

Divide every pixel value by 255

Training and Test dataset

We divide the dataset into train and test to validate if the model is not overfitting