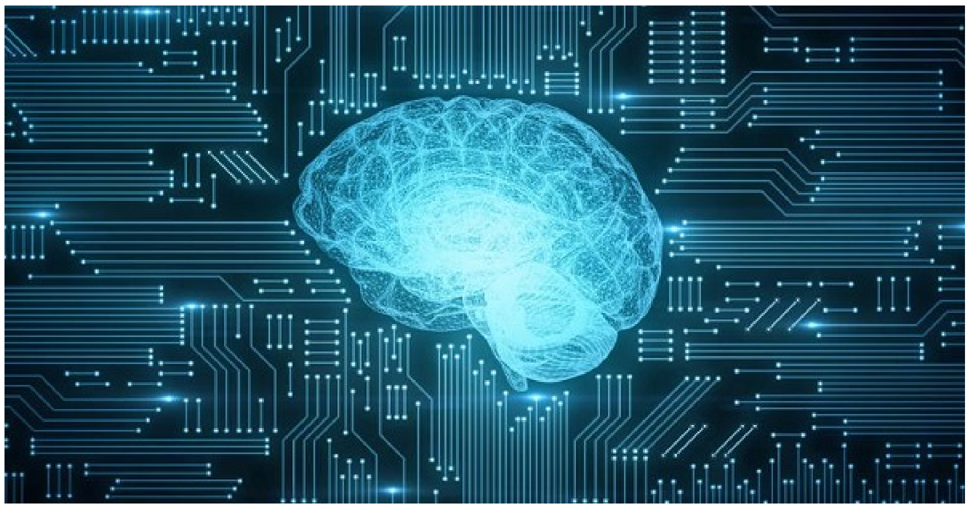
Final Report SOS
Neural Networks and Deep Learning

Satwik Murarka
Roll No. 190020101

Mentor-T Sanjev Vishnu



1 Introduction

Neural networks is a programming paradigm which enables the machine to learn from observational data and execute problems on it own instead of the conventional process of specifying the set of instructions.It is inspired by the human brain which has a similar process and learns through network of neurons.
Deep learning is a subset of the broader class of machine learning in artificial intelligence and uses the architecture of neural networks for learning and can be also thought of as the techniques and algorithms used to train neural networks.
Deep learning has been present and was extensively researched in the 1980's but due to some constraints was not able to develop very much.Firstly the computing power of machines was not high at that time and secondly due to lack of storage space of data.But with modern day systems these problems has been overcome and we have been able to apply deep learning techniques for solving problems and building models.
In this report I would be first giving a basic introduction to machine learning and its algorithms and then explore neural networks and deep learning for the major part.

2 Machine Learning

Machine Learning is an application of artificial intelligence which allows machines to learn and perform tasks without being explicitly programmed. A modern definition of machine learning by Tom Mitchell is-
"A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E."
Machine learning algorithms can be broadly divided into three types-

- Supervised Learning
- Unsupervised Learning
- Reinforcement Learning

2.1 Supervised Learning

Supervised learning is the machine learning task of learning from a given set of training data with known value of output to be able to perform the given task or solve the problem at hand.The problems solved using supervised learning can be mainly classified into-

- Regression
- Classification

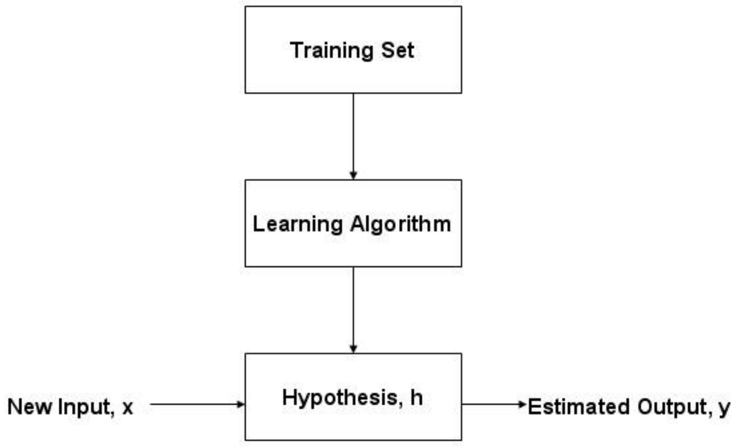


Figure 1: Flow Chart for Supervised Learning

2.1.1 Regression

Regression is a predictive statistical process where a model tries to find a relation between the input and output which is continuous valued.With the help of training data we train the model to develop a hypothesis to help map the inputs to outputs.This is usually done with an algorithm known as linear regression.

The hypothesis function of a linear regression with multi variables is given as-

$$h_\theta(X) = \theta^T X \tag{1}$$

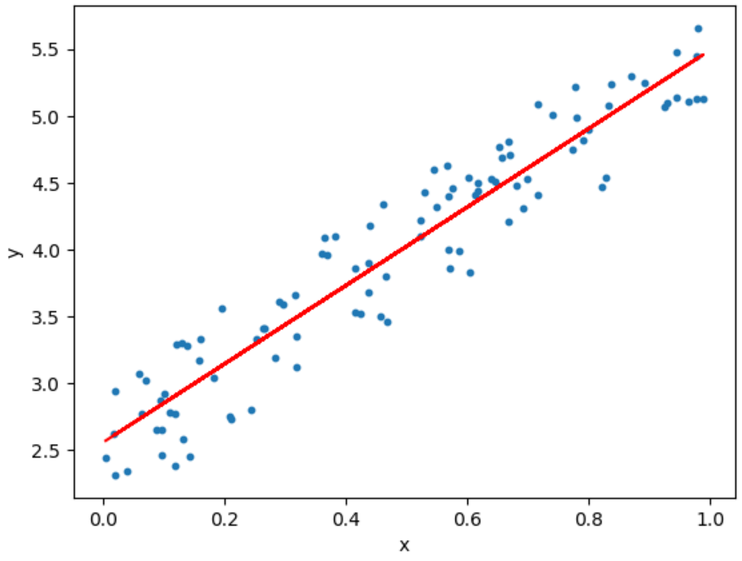


Figure 2: Linear Regression

$$h_\theta(X) = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x2 + \theta_3 x_3 + ... \tag{2}$$

where θ is the vector with the feature values and X represents the input matrix containing all the training examples.With the help of this hypothesis function we compute the cost function and then minimize it using the gradient descent algorithm to obtain the optimum feature matrix.This is discussed in detail in the neural networks section.

Apart from gradient descent for small datasets we can use normal equation for directly computing the feature vector.

$$\theta = (X^T X)^{-1} X^T y \tag{3}$$

where θ is the feature vector, X is the input matrix and y is the output vector.Regression can be also non-linear in which we use a non linear hypothesis function to fit our training data.

Hypothesis: $h_\theta(x) = \theta_0 + \theta_1 x$

Parameters: θ_0, θ_1

Cost Function: $J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$

Goal: $\underset{\theta_0, \theta_1}{\text{minimize}}\ J(\theta_0, \theta_1)$

Figure 3: Overview of Linear Regression

2.1.2 Classification

Classification refers to separating the data points into binary or multi-class categories according to some features.It can be also referred to as choosing a boundary hypothesis to separate different classes of data points.Popular classification algorithms are-

- Linear Classifiers
- Support Vector Machines
- Decision Trees

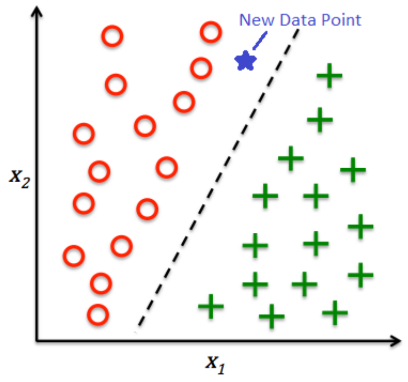


Figure 4: Classification

Another algorithm used for binary classification problems and also to predict probability of inputs is known as logistic regression.The general form is-

$$h_\theta(X) = g(\theta^T X) \tag{4}$$

where

$$g(x) = \frac{1}{1+e^{-x}} \tag{5}$$

Here the function g(x) is known as the sigmoid function.

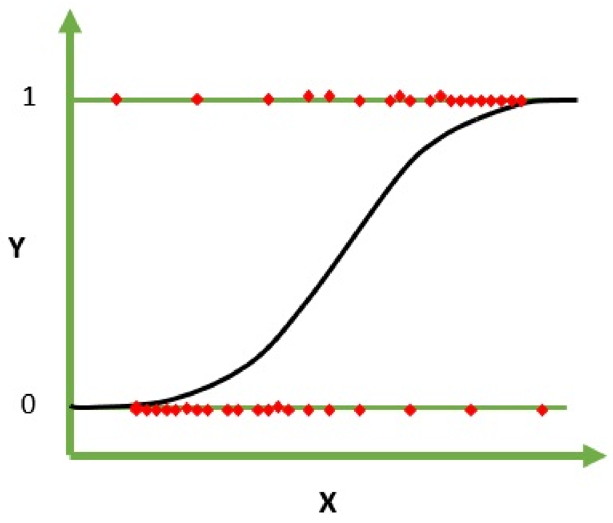


Figure 5: Logistic Regression

The cost function for logistic regression is different from that used in linear regression.It is defined as-

$$J(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases} \tag{6}$$

It satisfies the two basic requirements of a cost function which are that it is always positive and as the hypothesis function tends to output y the cost function goes to zero.It also helps us to overcome the problem of non-convexity which we would have encountered had we taken the MSE(Mean Squared Error) cost function.

2.2 Unsupervised Learning

Unsupervised Learning is a machine learning algorithm which takes in information which unlabelled and unstructured thus trying to find hidden structures and patterns in the data.It is called unsupervised due to the lack of guidance and the computer on its own tries to learn from data.The algorithms under this can be classified into-

- Clustering
- Association

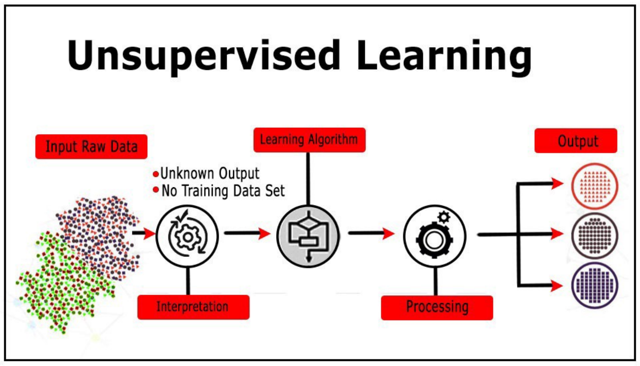


Figure 6: Unsupervised Learning

2.2.1 Clustering

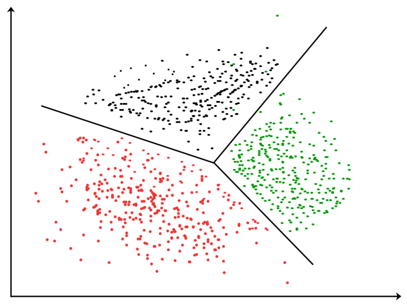


Figure 7: Clustering

Clustering is a process of separating the input values into different clusters such that data points in the same cluster are similar to each other and that in different are dissimilar to each other.One of the simplest clustering algorithms is the K-means clustering algorithm.

2.3 Reinforcement Learning

Reinforcement learning is a field of machine learning which involves the machine learning by interacting with an environment.The RL agent learns from the consequences of its actions and the model is checked by a system of rewards and punishments.The agent learns to maximize its rewards and minimize its penalties.

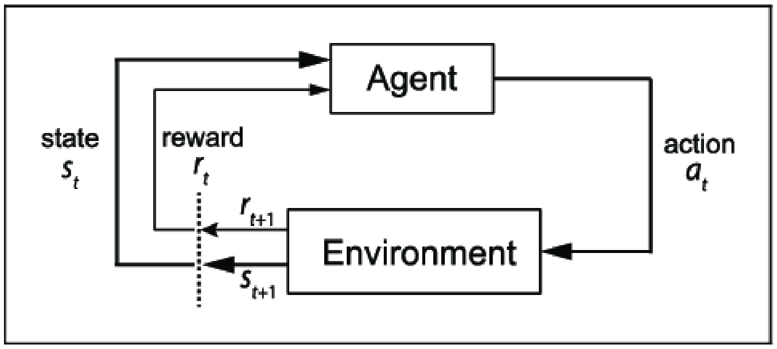


Figure 8: Reinforcement Learning

3 Neural Networks

Neural networks is an old but powerful algorithm to solve for classification problems for both binary and multi class.They are used for processing and learning from more complex data.Neural Networks can be of many types some of which are-

- Artificial Neural Network
- Convolutional Neural Network
- Recurrent Neural Network

The neural network aims to mimic the functioning of the brain and is thus comprised of many small interconnected building blocks known as artificial neurons.

3.1 Artificial Neurons

Artificial neurons are building blocks of artificial neural networks(ANNs) and the first of its kind was developed in the 1950s and 1960s by the scientist Frank Rosenblatt, inspired by earlier work by Warren McCulloch and Walter Pitts.This was called the perceptron but nowadays it is more common and efficient to use what is called the sigmoid neuron.

3.1.1 Perceptron

A perceptron works by taking in some binary inputs x_1, x_2,.., and produces a single binary output.As shown in figure 9 the perceptron takes three inputs and produces a single output.The output can be computed using a set of rules which was proposed by Rosenblatt.He introduced the concept of weights $w_1, w_2, ..$, expressing the importance of the respective inputs to the outputs.The output is whether 0 or 1 is decided by if the weighted sum $\sum_j w_j x_j$ is greater or lesser than a threshold value.
We may express this in a more simplified and mathematical way by first writing $\sum_j w_j x_j$ as a dot product $w \cdot x \equiv \sum_j w_j x_j$,where w and x are vectors whose components are weights and inputs respectively.The other change would be to move the threshold to the left hand side and replace it by what is called a perceptron's bias, $b \equiv -$threshold.Using the bias and vectors the perceptron rule can be written as:

$$\text{output} = \begin{cases} 0 & \text{if } w \cdot x + b \leq 0 \\ 1 & \text{if } w \cdot x + b > 0 \end{cases} \tag{7}$$

Bias can be thought of a measure of how easy is it for a perceptron to output a 1 or in biological terms how easy is for it to fire.Larger the bias, easier it would be for the output to be one.
A network of perceptrons can be used to simulate a circuit containing NAND gates and as NAND gates are universal for computation it infers that perceptrons are also universal for computation.The computational universality of perceptrons is simultaneously reassuring and disappointing. It's reassuring because it tells us that networks of perceptrons can be as powerful as any

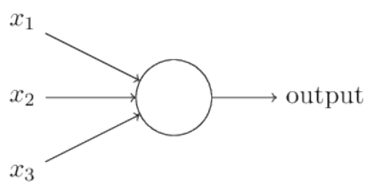


Figure 9: Perceptron

other computing device. But it's also disappointing, because it makes it seem as though perceptrons are merely a new type of NAND gate.
It turns out that we can devise learning algorithms which can automatically tune the weights and biases of a network of artificial neurons and this tuning happens in response to external stimuli, without direct intervention by a programmer.

3.1.2 Sigmoid Neuron

Learning algorithms make use of the process of bringing about small changes in the weights and bias in order to bring about small changes in output.We cannot achieve this with a system of perceptrons as small change in weights or bias may bring about a complete change in output.We need a suitable curve to solve this problem and the sigmoid neuron helps us tackle this problem.
Sigmoid neurons are depicted the same way as perceptrons in figure 1.It also takes several inputs and produces a single output.The difference is that the inputs $x_1, x_2, ..,$ can be any number between 0 and 1.The output is $\sigma(w \cdot x + b)$ where σ is the sigmoid function given by:

$$\sigma(z) \equiv \frac{1}{1+e^{-z}}. \tag{8}$$

Putting values we get:

$$\sigma(w \cdot x + b) \equiv \frac{1}{1+\exp(-\sum_j w_j x_j - b)}. \tag{9}$$

The algebraic form of sigmoid neuron seems very different from perceptrons but in reality they are very similar to each other.When $z = w \cdot x + b$ is very large the limit of the function tends to function and when z is very negative the limit tends to zero.So we can say the sigmoid neuron follows

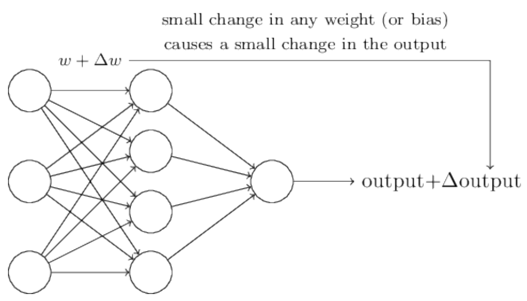

small change in any weight (or bias)
causes a small change in the output
$w + \Delta w$
output+Δoutput

Figure 10: Small changes in bias(or weight) cause changes in Output

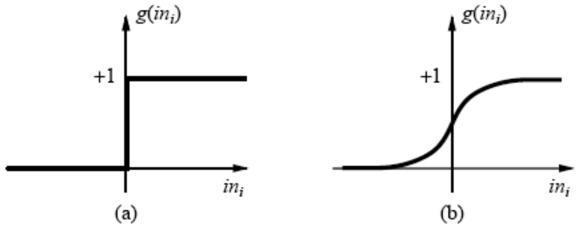


Figure 11: (a) Step function (b) Sigmoid function

the perceptron model for extreme cases and provides a continuous output for intermediate values.

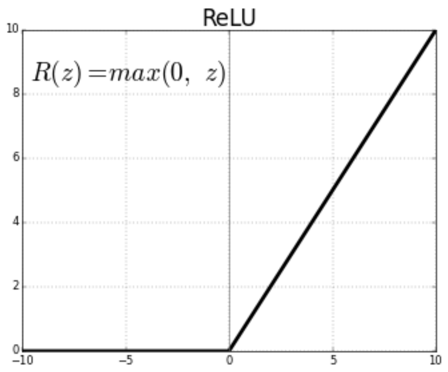
If σ had in fact been a step function, then the sigmoid neuron would be a perceptron, since the output would be 1 or 0 depending on whether $w \cdot x + b$ was positive or negative. But using the actual function helps us get a smoothed curve which is important as small changes in bias and weights will result only in small changes in output. The small change in output approximated with the help of calculus as:

$$\Delta\text{output} \approx \sum_j \frac{\partial\,\text{output}}{\partial w_j}\Delta w_j + \frac{\partial\,\text{output}}{\partial b}\Delta b \tag{10}$$

We may also use some other models of artificial neuron in which we use other functions other than sigmoid and operate accordingly.These types of functions are known as activation functions and they just like the above two systems are mathematical equations that determine the output of a neural network by relating it with the inputs,weights and bias.

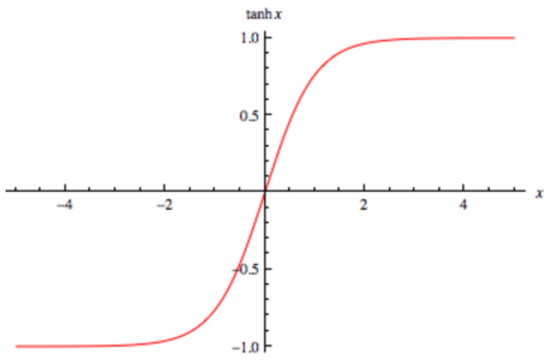
3.1.3 Some other Activation Functions

1. Rectifier



$$f(z) = max(0, z)$$

2. Hyperbolic Tangent



$$f(z) = \frac{1 - e^{-2z}}{1 + e^{-2z}}$$

4 Architecture of Artificial Neural Networks

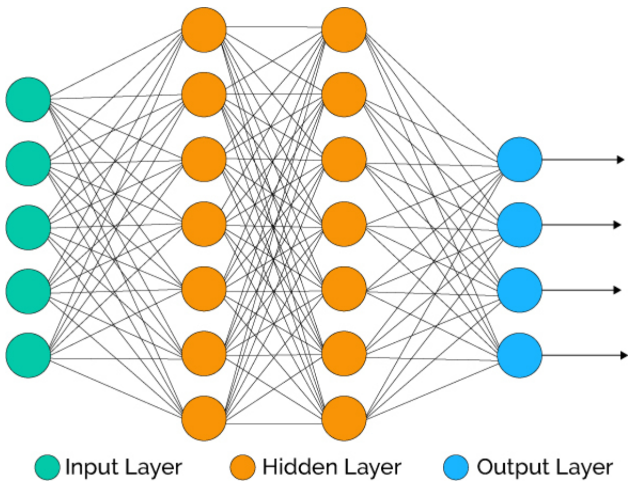


Figure 12: Design of a Neural Network

The leftmost layer acts as the input layer and rightmost as the output layer.The layers in between are known as the hidden layers.Such multiple layer networks are also called as multilayer perceptrons or MLPs.The design of input and output layers are very simple but the hidden layers take some time to design.There are no particular set of rules which help us design hidden layers.Instead, neural networks researchers have developed many design heuristics for the hidden layers, which help people get the behaviour they want out of their nets.

The networks in which output from one layer is used as input for second layer are known as feedforward neural networks.This means there are no loops in the network and information is feed only forward.The networks in which feedback loops are possible are known as recurrent neural networks.

Neural networks can be also classified depending on the number of hidden layers present.Networks with one or less hidden layers are known as shallow neural networks while those with more than one are known as deep neural networks.

5 Learning in Neural Networks

Till now we saw the different types of artificial neurons and the architecture of neural networks.Now we must understand how learning takes place in a neural network.A neural network learns through different algorithms using data which is called training data.By learning we mean difference between the the predicted output of an event and actual output be minimized.For this we define what is known as a cost function.

5.1 Cost Function

Cost function is a parameter to measure the performance of our deep learning model which would help us change the weights and bias to improve the performance of our network.Cost function is defined as:

$$C(w,b) \equiv \frac{1}{2n} \sum_x \|y(x) - a\|^2 \tag{11}$$

Here, w denotes the collection of all weights in the network, b all the biases, n is the total number of training inputs, a is the vector of outputs from the network when x is input, and the sum is over all training inputs, x and $y(x)$ denotes the actual outcome.
The cost function is always non-negative and when it becomes very small, i.e, $C(w,b) \approx 0$ then the predicted and actual output are almost equal and our algorithm has finding the appropriate weights and biases.So the aim of our training algorithm would be to minimize the cost function ans we would do it by an algorithm known as gradient descent.

5.2 Gradient Descent

Gradient descent is an optimization algorithm used to minimize some function by moving in small steps towards or in direction of steepest descent which is determined by the negative of the gradient.
As an example we try to minimize a function, $C(v_1, v_2)$, with two independent variables with the help of gradient descent.With the help of calculus change in C is given as

$$\Delta C \approx \frac{\partial C}{\partial v_1}\Delta v_1 + \frac{\partial C}{\partial v_2}\Delta v_2. \tag{12}$$

We can write the above equation in form of vector multiplication by defining vector of changes in v, $\Delta v \equiv (\Delta v_1, \Delta v_2)^T$ and the gradient vector $\nabla C \equiv \left(\frac{\partial C}{\partial v_1}, \frac{\partial C}{\partial v_2}\right)^T$. With these ΔC can be rewritten as:

$$\Delta C \approx \nabla C \cdot \Delta v. \tag{13}$$

This equation helps us decide how to make Δv negative.Suppose

$$\Delta v = -\eta \nabla C, \tag{14}$$

where η is a small positive parameter known as the learning rate.
The learning rate needs to be of the optimum value so that the process does not takes much time and also the minimum is not skipped due to large steps taken.This guarantees that $\Delta C \leq 0$ and so C will always decrease if we change v according Equation (14).This can be extended to functions with more than two variables as well.The minimum value of C is found out by updating the variables as:

$$v \rightarrow v' = v - \eta \nabla C. \tag{15}$$

This is known as the update rule.
Now as we understood the concept of gradient descent we can apply it to our neural network. We restate the gradient descent rule with the help of weights and biases as:

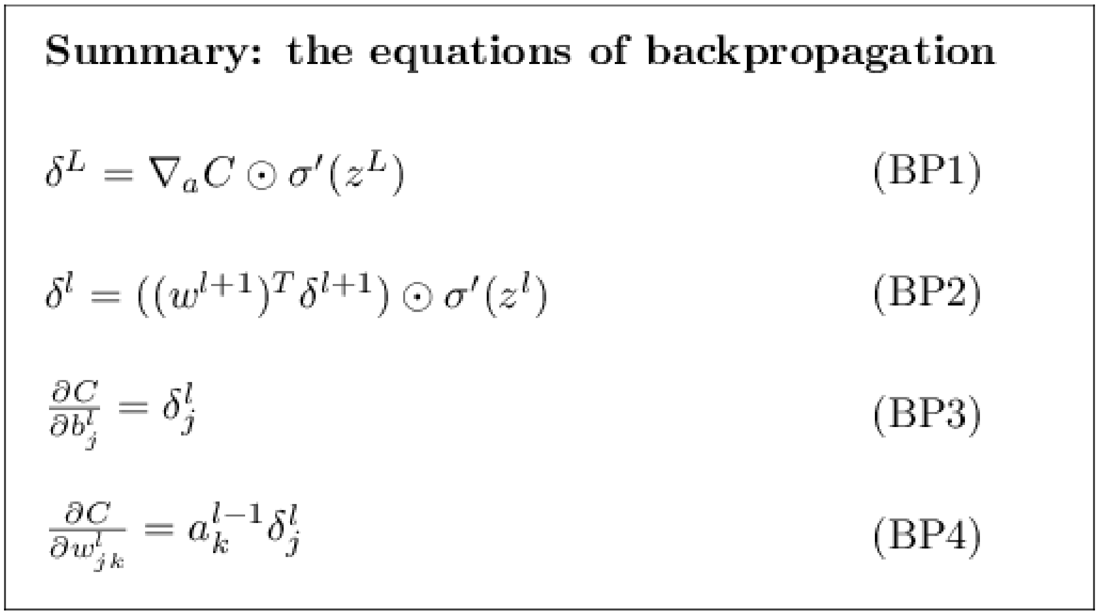
$$w_k \;\rightarrow\; w_k' = w_k - \eta \frac{\partial C}{\partial w_k} \tag{16}$$

$$b_l \;\rightarrow\; b_l' = b_l - \eta \frac{\partial C}{\partial b_l}. \tag{17}$$

By apply this update repeatedly we hope to find the minimum of the cost function.In practice we do not apply gradient descent for single training inputs,which is known as stochastic gradient descent, but group them randomly and averaging over this small sample proves to speed up learning.This process is known as mini batch gradient descent.If we apply gradient descent to the whole batch of training data then it is called batch gradient descent.
We can compute the gradient of the cost function by an algorithm known as backpropagation.

5.3 Backpropagation

Backpropagation is an algorithm used to find the gradient of the cost function which is very expensive computationally if computed directly.
Backpropagation is about understanding how the change to weights and biases affect the cost function.This means we have to compute the partial derivatives $\partial C/\partial w^l_{jk}$ and $\partial C/\partial b^l_j$ where l represents the current layer,k represents the k^{th} neuron in the l^{th} layer and j represents the j^{th} neuron in the l^{th} layer for bias and $(l+1)^{th}$ layer for weight.
In this process we compute something called error δ^l_j in the j^{th} neuron in the l^{th} layer.Using backpropagation we would relate this error to the partial derivatives. Backpropagation is called so because we compute the error



Summary: the equations of backpropagation

$$\delta^L = \nabla_a C \odot \sigma'(z^L) \qquad \text{(BP1)}$$

$$\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l) \qquad \text{(BP2)}$$

$$\frac{\partial C}{\partial b^l_j} = \delta^l_j \qquad \text{(BP3)}$$

$$\frac{\partial C}{\partial w^l_{jk}} = a^{l-1}_k \delta^l_j \qquad \text{(BP4)}$$

Figure 13: Equations used in Backpropagation Algorithm

function backwards as the cost function is related to the outputs of the network.We need to work out the equations and find the error and update the weights and biases.

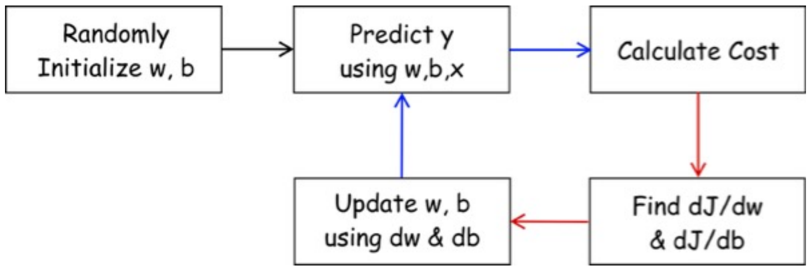


Figure 14: Flow chart representing Backpropagation

6 Improving Learning in Neural Networks

Learning in neural networks can be improved in many ways by changing the different basic algorithms involved in the process. Hyperparameters can also be changed to select the best model from a number of options.

6.1 Cross-Entropy cost function

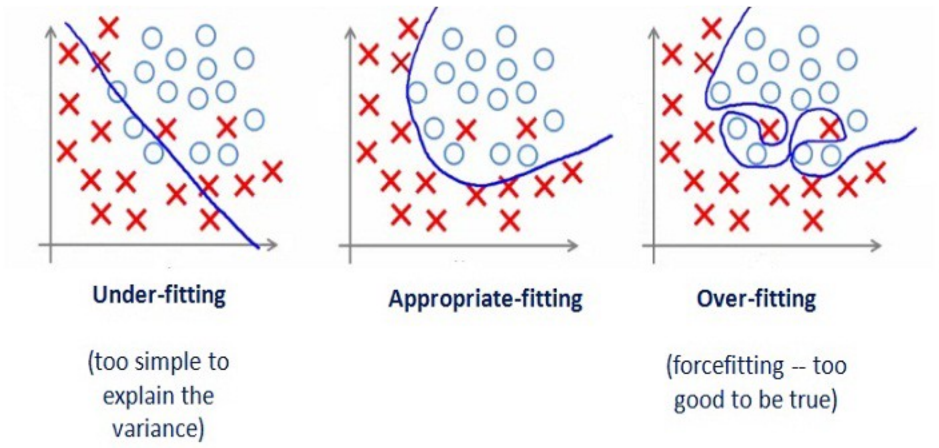
In contrast to human behaviour which learns more quickly upon making bigger mistakes, our model performs quite the opposite to it.This is due to the magnitude of derivatives being small which leads to a problem known as the learning slowdown.
To overcome this problem of learning slowdown we introduce a new cost function known as the cross-entropy cost function.This is represented as-

$$C = -\frac{1}{n}\sum_{x}\left[y \ln a + (1-y)\ln(1-a)\right] \tag{18}$$

where a is the predicted output,y is the actual output and x represents the training inputs.

6.2 Overfitting and Regularization

Overfitting is the problem encountered when the hypothesis fits the training data very well but fails to generalize to new examples.This can be due to reasons like presence of too many features or lack of training data.A model which overfits data is said to have high variance as compared to a model which under fits data.Underfitting means that the hypothesis is not able to fit the training data and is said to have high bias.We need to develop a model


Under-fitting
(too simple to explain the variance)
Appropriate-fitting
Over-fitting
(forcefitting -- too good to be true)

so that there is a trade off between bias and variance. Another method to solve that problem of overfitting is known as regularization.
Regularization is used for tuning the hypothesis function by adding an additional term in the cost function which restricts the features or parameters from taking extreme values.The cost function after regularization can be represented as-

$$C = C_0 + \frac{\lambda}{2n}\sum_w w^2, \tag{19}$$

where C_0 is the unregularized cost function,n is number of training examples,λ represents the regularization parameter and is multiplied with sum of the squares of weights.This would lead smaller size for hypothesis features which would lead to a simpler model less prone to overfitting.

6.3 Feature Scaling

We can speed up our gradient descent algorithm if we have the features in almost the same range.This can be visualized from contour plots from which we can deduce that we require more of steps and hence more time with models having varied numerical features.
Feature scaling can be done by some methods such as diving the feature by the range of values(max-min) or apply mean normalization which is-

$$x_{new} = \frac{x_{old} - \mu}{(\text{max-min})} \tag{20}$$

where μ represents the mean over the training set and others have the same meaning as above in the document.

6.4 Hyperparameter Tuning

The parameters which describe a model and its architecture is known as a hyperparamter and searching for these hyperparameters so that we can get a best performing model is known as hyperparameter tuning.This is usually done manually and it helps us increase our understanding of the neural network.Taking an example we have a task of choosing the learning rate parameter.We select an optimum value for this such that it is-

- Not too large which could have resulted in the minimum getting skipped
- Not too small which would result in the process taking long time

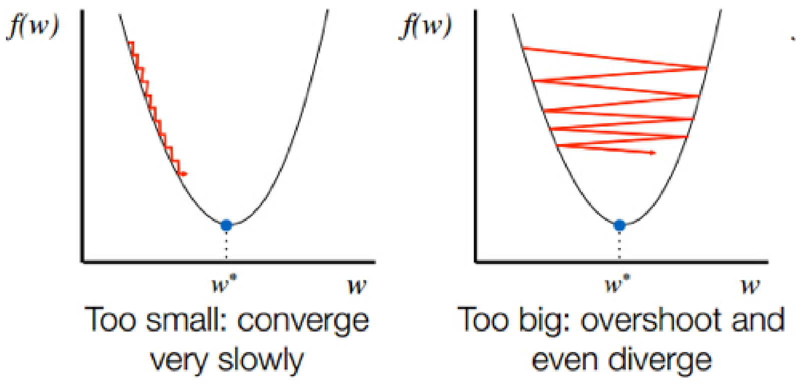

f(w)
w*
w
Too small: converge very slowly
f(w)
w*
w
Too big: overshoot and even diverge

Figure 15: Learning Rate

We divide our data set into a third set known as the validation set which is to evaluate the fit of the model on the training data and allow hyperparameter tuning simultaneously. There are also some automated ways in which we can choose the appropriate hyperparameters for our model.Common methods among these are Grid search and Random search.

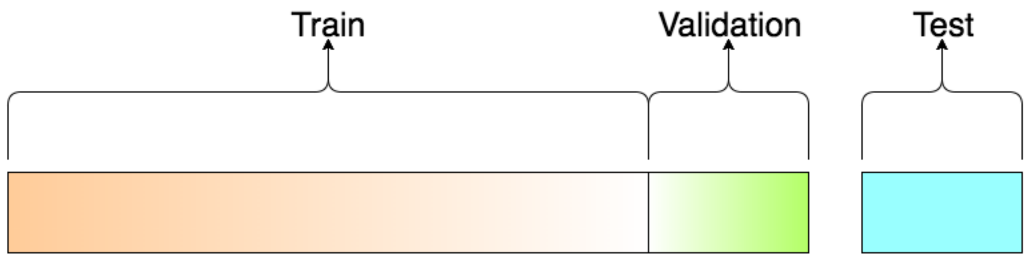


Figure 16: Division of Data

7 Practical Project and Other Learnings

Through my project I have been able to explore and learn many different applications and languages.I learnt Python and its libraries like pandas,numpy,matplotlib and sklearn for working and practical applications of the subject.I also learnt to work with libraries like Tensorflow and Keras which are very useful in deep learning.I also learnt LaTeX for writing this report.
I have attempted to solve a very basic classification problem which wants to find whether a customer would stay or leave a telecom service.The dataset for this problem has been taken form Kaggle.The following link is of the github repo which contains the datset and code for the problem.

GithubRepo :`https://github.com/SatwikMurarka/Summer-of-Science.git`

References

[1] Michael A. Nielsen, ”Neural networks and deep learning”, Determination press, 2015

[2] Andrew Ng,”Machine Learning”,Coursera Course

[3] Ian Goodfellow, Yoshua Bengio, and Aaron Courville,”Deep learning”, MIT press, 2016

[4] Towards Data Science,"`https://towardsdatascience.com/`"

[5] Deep Learning A-Z,"`https://www.udemy.com/course/deeplearning/`"