# CSE 598 - Bio-Inspired AI and optimisation
# Mini-Project 7: Artificial Neural Networks

**Ajay Kannan - 1219387832**

## Introduction

Linear regression, fundamental of a neural network, is an attempt to model the relationship between two variables by fitting a linear equation to observed data, where one variable is considered to be an explanatory variable and the other as a dependent variable. In statistics, linear regression is a linear approach for modelling the relationship between a scalar response and one or more explanatory variables. The case of one explanatory variable is called simple linear regression. For more than one, the process is called multiple linear regression. This term is distinct from multivariate linear regression, where multiple correlated dependent variables are predicted, rather than a single scalar variable.

## Methodology

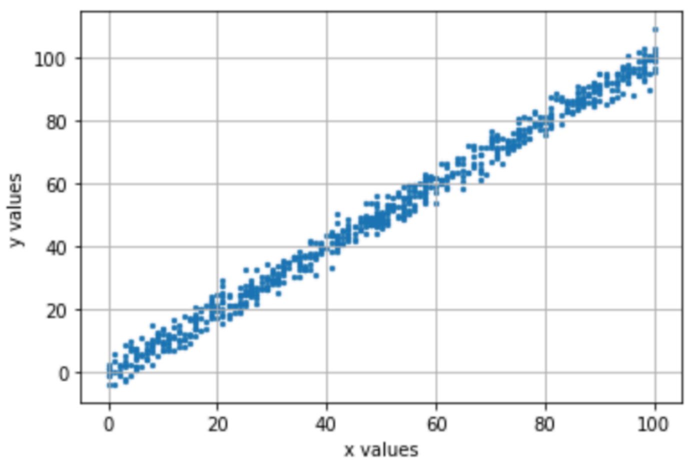
1. Linear model -
In disciplines where it is common to focus on datasets with just a few features, explicitly expressing models long-form like this is common. In machine learning, we usually work with high-dimensional datasets, so it is more convenient to employ linear algebra notation. When our inputs consist of d features, we express our prediction => y = w1*x1 + w2*x2 + … + wk*xk + b

In this solution we take a simple linear model => y = w*x + b.

2. Dataset -
- The training dataset is a CSV file with 700 data pairs (x,y).
- The x-values are numbers between 0 and 100.
- The corresponding y-values have been generated using the Excel function NORMINV(RAND(), x, 3). Consequently, the best estimate for y should be x.
- The test dataset is a CSV file with 300 data pairs.

```
    x          y
0  24  21.549452
1  50  47.464463
2  15  17.218656
3  38  36.586398
4  87  87.288984
```

Training set is shown beside and a pique of the dataset shown above.



3. Gradient Descent -
Gradient descent is an optimisation algorithm which is commonly-used to train machine learning models and neural networks. Training data helps these models learn over time, and the cost function (C) within gradient descent specifically acts as a barometer, gauging its accuracy with each iteration of parameter updates.

$$C = \frac{1}{2N}\sum_{i=1}^{N}(h(x_i) - y_i)^2$$

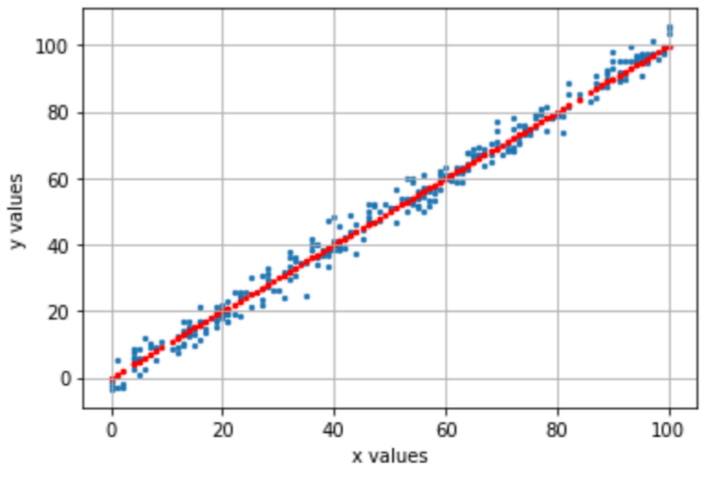
The primary set-up for learning neural networks is to define a cost function (also known as a loss function) that measures how well the network predicts outputs on the test set. The goal is to then find a set of weights and biases that minimises the cost. One common function that is often used is the mean squared error, which measures the difference between the actual value of y and the estimated value of y (the prediction). The equation has only weight and bias.

4. Metrics -
This is simply the average of the absolute difference between the target value and the value predicted by the model. Not preferred in cases where outliers are prominent.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|\hat{y}_i - y_i|$$

Also, Explained variance regression score function is used in the code to calculate variance. The difference is that the explained variance use the biased variance to determine what fraction of the variance is explained. R-Squared uses the raw sums of squares. If the error of the predictor is unbiased, the two scores are the same.

## Results



-The training is done and it is witnessed that after 1000 iterations of training, it converges.

```
iteration: 0 cost:  743.4332672935938
iteration: 400 cost:  3.9352561522452425
iteration: 800 cost:  3.935237092273996
iteration: 1200 cost:  3.9352228867692007
```

-The testing set is run through the gradient descent after training.
-Blue dots are true values and red is predicated.
- Mean absolute error is 2.418092. That is 2.41 is the variance.
- Explained Variance score is 0.988.

## Discussion

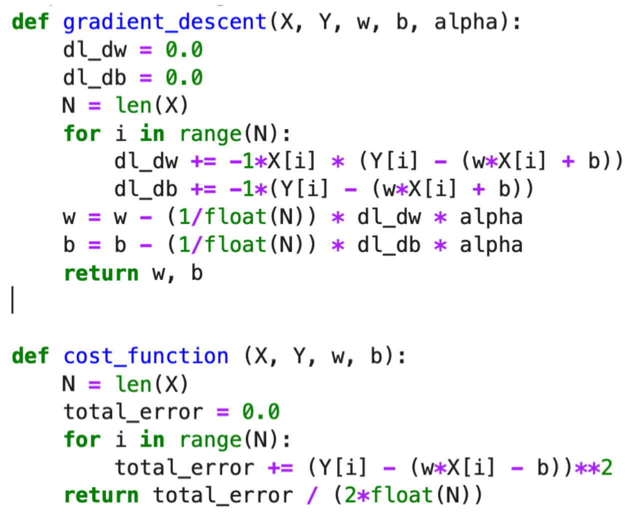
Machine learning, more specifically the field of predictive modelling is primarily concerned with minimising the error of a model or making the most accurate predictions possible, at the expense of explainability. In applied machine learning we will borrow, reuse and steal algorithms from many different fields, including statistics and use them towards these ends. As such, linear regression, **basic form of neural network**, was developed in the field of statistics and is studied as a model for understanding the relationship between input and output numerical variables, but has been borrowed by machine learning. It is both a statistical algorithm and a machine learning algorithm.

In this experiment, the two graphs explained above show the training graph and the testing graph. The mean square error or variance is calculated with regards to the testing set. Suppose if points are not in a straight line (multi-variate), this linear regression model would fail. Then we would have to go to higher order regression.

To sum up, Supervised Machine Learning has a broad range of algorithms. Linear Regression is among mainly used ones.

## Appendix

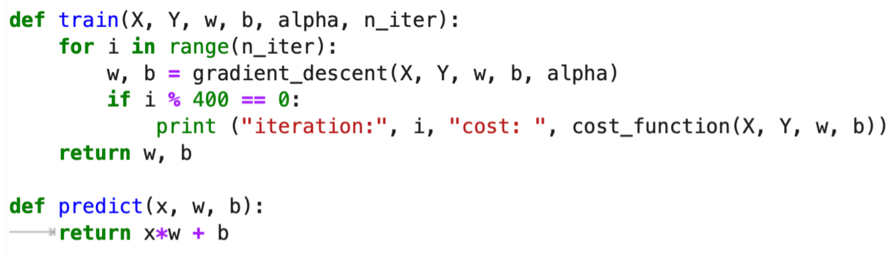
1. Gradient Descent & Cost Function



```
def gradient_descent(X, Y, w, b, alpha):
    dl_dw = 0.0
    dl_db = 0.0
    N = len(X)
    for i in range(N):
        dl_dw += -1*X[i] * (Y[i] - (w*X[i] + b))
        dl_db += -1*(Y[i] - (w*X[i] + b))
    w = w - (1/float(N)) * dl_dw * alpha
    b = b - (1/float(N)) * dl_db * alpha
    return w, b

def cost_function (X, Y, w, b):
    N = len(X)
    total_error = 0.0
    for i in range(N):
        total_error += (Y[i] - (w*X[i] - b))**2
    return total_error / (2*float(N))
```

2. Train & predict function



```
def train(X, Y, w, b, alpha, n_iter):
    for i in range(n_iter):
        w, b = gradient_descent(X, Y, w, b, alpha)
        if i % 400 == 0:
            print ("iteration:", i, "cost: ", cost_function(X, Y, w, b))
    return w, b

def predict(x, w, b):
    return x*w + b
```