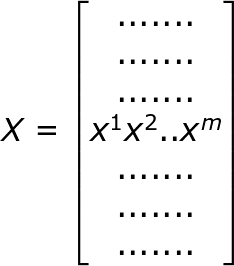
Logistic regression is a generalized linear model that we can use to model or predict categorical outcome variables. For example, we might use logistic regression to predict whether someone will be denied or approved for a loan, but probably not to predict the value of someone’s house.

Let’s say we have *n* dimensional of input feature. A single training example will be represented as *(x,y)* where **x is n dimensional feature vector** and **y is label (0/1, True/False etc.).***m* denotes the total number of training examples. To put it concisely, our feature matrix looks as:

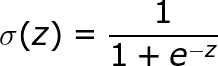


https://cdn-images-1.medium.com/max/750/1*XFs9AW6nQcens5Lg8uZUAA.png

The problem statement formulations turn out to be given X, we need to calculate ŷ = P( y=1 | X). What this means is that we need to calculate the probability of target variable to be 1 (or 0) given the training set

We apply sigmoid function so that we contain the result of ŷ between 0 and 1 (probability value). The sigmoid function definition is as follows:

https://cdn-images-1.medium.com/max/750/1*xDjD0feFXCHkhgqMHYFvrg.png



**Task 1.1: NORMALIZATION**-

We need to normalize our feature matrix X.

Normalization is useful for speeding up convergence and for interpreting coefficients.

The normalization technique we use here is:

**1-((max-X)/rng)**

Max- Maximum value of column

Rng=maximum value of column-minimum value of column

**Task 1.2: Logisitc Function (Sigmoid)**

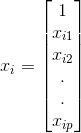
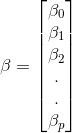
We apply sigmoid function so that we contain the result of ŷ between 0 and 1 (probability value). The sigmoid function definition is as follows:

Your task is to define the following function

https://latex.codecogs.com/gif.latex?h%28x_i%29%20%3D%20g%28%5Cbeta%5ET%20x_i%29%20%3D%20%5Cfrac%7B1%7D%7B1%20+%20e%5E%7B-%5Cbeta%5ET%20x_i%7D%7D

where,  
https://latex.codecogs.com/gif.latex?g%28z%29%20%3D%20%5Cfrac%7B1%7D%7B1%20+%20e%5E%7B-z%7D%7D

g(z) is known as the sigmoid function.

Hint:

T=Transpose of 

For getting transpose of ‘beta’ use beta.T.

For multiplying two matices A and B, use np.dot(A,B)

**Task 2.1: cost\_func**

When implementing logistic regression, our job is to learn parameters X and ‘Beta’ so that ŷ is approximately equal to the test target . To learn the parameters X and ‘Beta’, we need to define a cost function which we would use to train the logistic regression model. A cost function is an estimator of how good or bad our model is in predicting the known output in general.

**TASK- Compute COST FUNCTION(Omit the summation)**

https://latex.codecogs.com/gif.latex?J%28%5Cbeta%29%20%3D%5Csum_%7Bi%3D1%7D%5E%7Bn%7D%20-%20y_ilog%28h%28x_i%29%29%20-%20%281-y_i%29log%281-h%28x_i%29%29

HINT: Compute the two terms under the summation individually and store them in variables, and then compute the difference. For h(x) call logistic\_func(sigmoid function)

**Task 2.2- Gradient Descent, grad\_desc()**

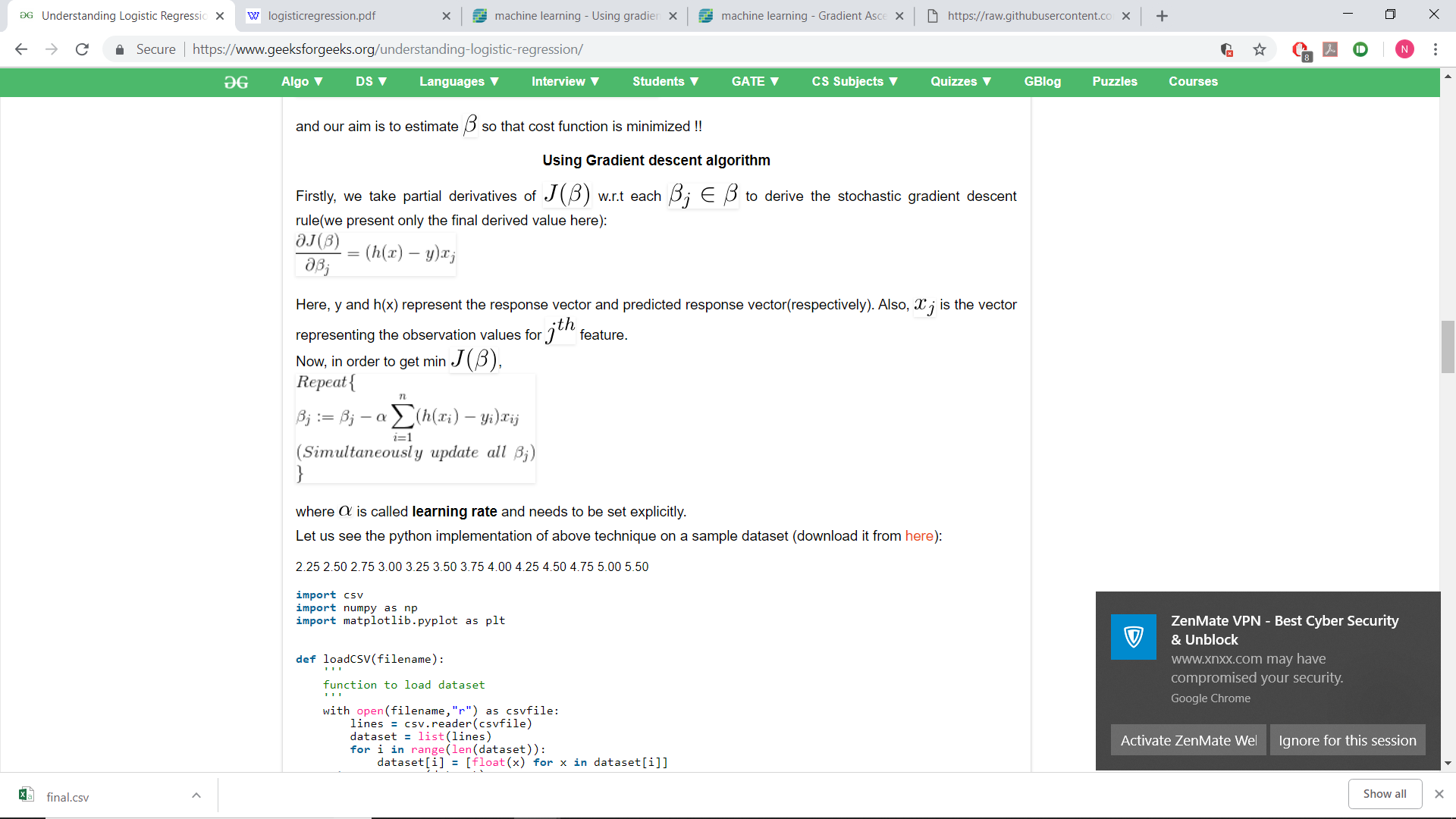
Gradient Descent

Basically, the cost function measures how well our parameters X and Beta are doing on the training data set.

So, it seems natural to minimize the cost function for minimal error across the training data set to find X and ‘Beta’. We would achieve the value of the parameters using gradient descent technique.

We aim to find a value of ‘Beta’ so that the cost function is minimised

The gradient descent formula to find optimum ‘Beta’ is given by:



where Alpha is called **learning rate** and is set explicitly.

**h(x)** is our logistic function

**TASK- Update values of Beta, and compute cost for each Beta, and check if cost function is MINIMUM.**

**Hint: The term under the summation *“(h(x)-y)x”* term has been computed using** *log\_grad()* **function which has been defined for you, you can call log\_grad() for simplification. Omit the summation sign in your formula.**

By doing this we are trying to find optimal values of Beta so that the cost function is minimum