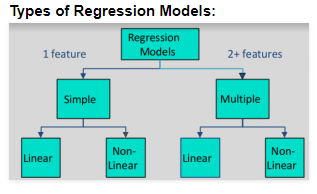
**Linear Regression**

**Linear regression:**

Linear regression is a basic and commonly used type of predictive analysis.

The relationship between dependent variable *(Y)* and one or more independent variables (*X).*



**What is Simple linear regression?**

Simple linear regression assumes that a linear relationship exists between the response variable and explanatory variable; it models this relationship with a linear surface called a hyper plane. A hyper plane is a subspace that has one dimension less than the ambient space that contains it. In simple linear regression, there is one dimension for the response variable and another dimension for the explanatory variable, making a total of two dimensions. The regression hyper plane therefore, has one dimension; a hyper plane with one dimension is a line.

First, the regression might be used to identify the strength of the effect that the independent variable(s) have on a dependent variable.  Typical questions are what is the strength of relationship between dose and effect, sales and marketing spending, or age and income.

Second, it can be used to forecast effects or impact of changes.  That is, the regression analysis helps us to understand how much the dependent variable changes with a change in one or more independent variables.  A typical question is, “how much additional sales income do I get for each additional $1000 spent on marketing?”

Third, regression analysis predicts trends and future values.  The regression analysis can be used to get point estimates.  A typical question is, “what will the price of gold be in 6 months?”

**Line equation**

**y=mx+c+e**

Y

# error(e) slope(m)

# Predicted value (minimize the error)

# (c) Intercept X

**What is cost function?**

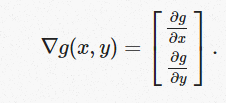
Error is also called as cost function.  Cost function is the sum of squared errors over your training set.

Cost function = (predicted value - actual value)^2

Cost function=(y[i]-(m\*x[i]+c)) ^2

**What is Gradient Descent?**

There is an incredibly simple way to minimize a multivariable function iteratively: gradient descent. As you may remember from your calculus class, the gradient of a function g(x,y)is



More importantly, however, the gradient of a function is a vector which points towards the direction of maximum increase. Consequently, in order to minimize a function, we just need to take the gradient, look where it’s pointing, and head the other direction. Thus, gradient descent can be succinctly described in just a few steps:

* Choose a random starting point for your variables. For performance reasons, this starting point should really be random - use a pseudorandom number generator to choose it.
* Take the gradient of your cost function at your location.
* Move your location in the opposite direction from where your gradient points, by just a bit. Specifically, take your gradient ∇g, and subtract α∇g from your variables, where α is just some small number (0.1? 0.5? 0.01?) that you can tune to adjust how fast your algorithm runs.
* Repeat steps 2 and 3 until you’re satisfied and repeating them more doesn’t help you too much.

Using gradient descent with a few hundred iterations, we can easily find parameters β for our linear regressions which give us a nice fit. (Note that there are faster algorithms than gradient descent, but they operate on the same basic principles!)

Here is a series of graphs of data points and lines of best fit that are generated by running gradient descent for a different number of iterations. Note how with each iteration, the line fits the data better.

# y=mx+c

# 

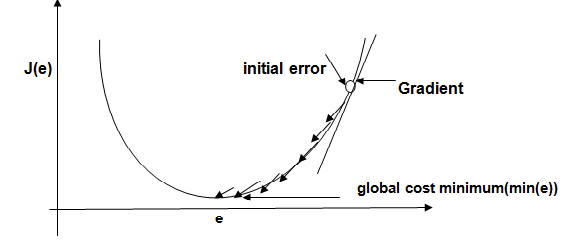
# Cost function (e) = (y[i]-(m\*x[i]))^2

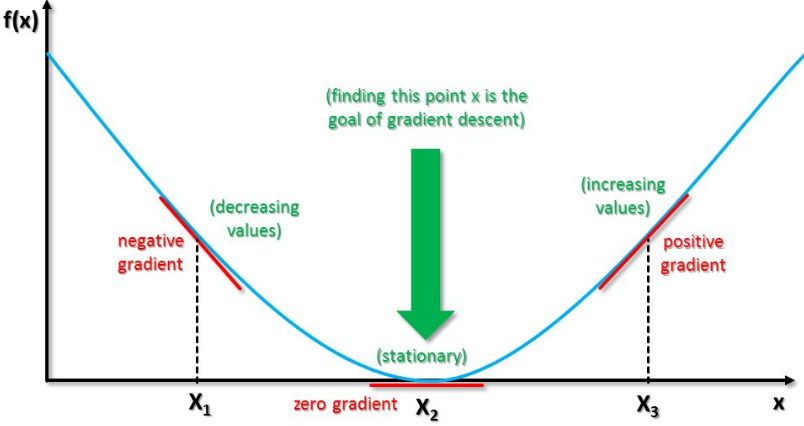
Gradient descent = d(e)/dm = d/dm ( y[i]-m\*x[i])^2

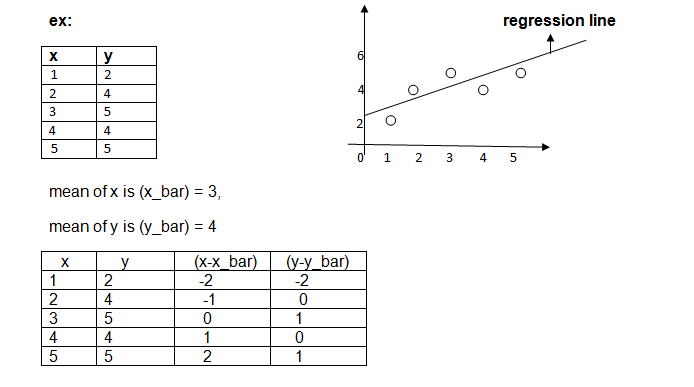
= 2 (y[i] - m \* x[i] ) \* d/dm (-m \* x[i])

= 2(y[i] - m \* x[i] )\* -x[i]

Gradient descent = -2(y[i] - m \* x[i])\* x[i]







b1 (slope) = (x-x\_bar)(y-y\_bar)

(x-x\_bar)^2

b0 (intercept) = ( y\_bar) - b1\*(x\_bar)

|  |  |
| --- | --- |
| (x-x\_bar)^2 | (y-y\_bar)(x-x\_bar) |
| 4 | 4 |
| 1 | 0 |
| 0 | 0 |
| 1 | 0 |
| 4 | 2 |

sum=10 sum=6

b1 (slope) = (x-x\_bar)(y-y\_bar) = 0.6

(x-x\_bar)^2

b0 (intercept) = ( y\_bar) - b1\*(x\_bar) = 2.2

**RMSE (Root mean square error):** compare between (Estimated distance - mean),

(Actual distance -mean)

Estimated values

6 regression line

4 mean=4

2

0 1 2 3 4 5 6

6 (actual values)

4 mean=4

2

0 1 2 3 4 5

**RMSE (Root mean square error):**

sqrt (1/m ( y^-y[i])^2 , y^=b0+b1\*x

m= len(x), i=1 to m

**R^2** score = 1 (y - y^)^2

(y-y\_bar)^2

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| x | y | y-y\_bar | (y-y\_bar)^2 | y^ | y^-y\_bar | (y^-y\_bar)^2 |
| 1 | 2 | -2 | 4 | 2.8 | -1.2 | 1.44 |
| 2 | 4 | 0 | 0 | 3.4 | -.6 | .36 |
| 3 | 5 | 1 | 1 | 4 | 0 | 0 |
| 4 | 4 | 0 | 4 | 4.6 | 0.6 | 0.36 |
| 5 | 5 | 1 | 1 | 5.2 | 1.2 | 1.44 |

**R^2 score** = 1 - (y - y^)^2

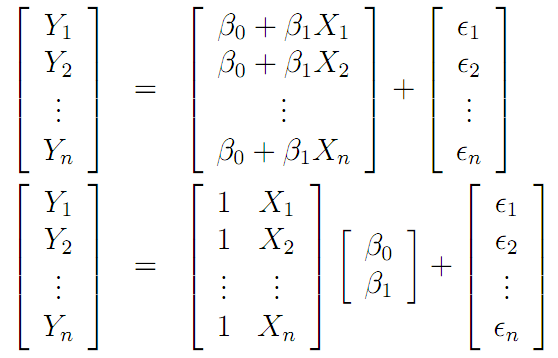
(y-y\_bar)^2

If R^2 is near to 1 better the line fits the data & when R^2 is far from 1, line not represents data at all.

**Multiple linear Regression**

**Multiple linear Regressions:**

[Multiple linear Regressions](http://www.statisticssolutions.com/academic-solutions/membership-resources/member-profile/data-analysis-plan-templates/data-analysis-plan-multiple-linear-regression/) is the most common form of linear regression analysis. As a predictive analysis, therelationship between two or more independent variables(x) and one dependent variable(y).



Similar to Simple Linear Regression, we have input variable(**X**) and output variable(**Y**). But the input variable has n features. Therefore, we can represent this linear model as follows;

Y=β0+β1x1+β1x2+…+βnxn

xi is the ith feature in input variable. By introducing x0=1, we can rewrite this equation.

Y=β0x0+β1x1+β1x2+…+βnxn

x0=1

Now we can convert this equation to matrix form.

Y=βTX

Where,

β=[β0β1β2..βn]T

And ,

X=[x0x1x2..xn]T

We have to define the cost of the model. Cost basically gives the error in our model. **Y** in above equation is the hypothesis (approximation). We are going to define it as our hypothesis function.

hβ(x)=βTx

And the cost is,

J(β)=12m∑i=1m(hβ(x(i))−y(i))2J

By minimizing this cost function, we can get find β. We use **Gradient Descent** for this.

### Gradient Descent:

### Gradient Descent is an optimization algorithm. We will optimize our cost function using Gradient Descent Algorithm.

##### Step 1

Initialize values β0, β1… βn with some value. In this case we will initialize with 0.

#### Step 2

#### Iteratively update,

βj:=βj−α∂βjJ(β)

This is the procedure. Here α is the learning rate. This operation ∂βjJ(β) means we are finding partial derivate of cost with respect to each βj. This is called Gradient.

In step 2 we are changing the values of βj in a direction in which it reduces our cost function. And Gradient gives the direction in which we want to move. Finally we will reach the minima of our cost function. But we don’t want to change values of βj drastically, because we might miss the minima. That’s why we need learning rate.

But we still didn’t find the value of ∂βjJ (β). After we are applying the mathematics.

The step 2 becomes.

βj:=βj−α1m∑i=1m(hβ(x(i))−y(i))x(i)j

We iteratively change values of βj according to above equation. This particular method is called **Batch Gradient Descent**.