REGRESSION ANALYSIS : LINEAR

BY – MAUAJAMA FIRDAUS & TULIKA SAHA

MACHINE LEARNING

- It is the science of getting computer to learn without being explicitly programmed.
- Machine learning is an area of artificial intelligence concerned with the development of techniques which allow computers to "learn".
- More specifically, machine learning is a method for creating computer programs by the analysis of data sets.

APPLICATIONS

- Search Engines like Google, Bing etc.
- Facebook photo tagging application.
- Self Customizing Programs and many more.

SUPERVISED LEARNING

- Given the "right answer" for each of our examples in the training set.
- The task of the algorithm is to find many more such right answers for new examples.
- Consists of two problems-
 - Regression Problem
 - Classification Problem

REGRESSION PROBLEM

• The term "regression" refers to the fact that we are trying to predict a continuous-valued output.

Housing price prediction.



CLASSIFICATION PROBLEM

 It refers to the fact that we are trying to predict discrete valued outputs.



UNSUPERVISED LEARENING

- Given a set of data or examples in the training set without prior information of what the data is all about.
- The task of the algorithm is to find the structure among the data.
- Clustering Problem is an example of Unsupervised Learning.



LINEAR REGRESSION

Why are we looking at this algorithm ?

- Machine Learning a field of predictive modelling
- Linear Regression
 - Developed in the field of statistic
 - A model that attempts to show the relationship between two variables with a linear equation
 - Borrowed by Machine Learning
- Finds it's application in
 - Evaluating Trends and Sales Estimates
 - Analyzing the Impact of Price Change and many more

LINEAR REGRESSION

- Basic Framework
 - Dependent variable y, also called the output/explained variable which is to be predicted
 - Independent variables x_i, also called the input/explanatory variables that is to be used for making predictions.
- Regression is the general task of attempting to predict the value of output variable y from the input variables X_i.



Training set of	Size in feet ² (x)	Price (\$) in 1000's (y)
housing prices	2104	460
(Portland, OR)	1416	232
	1534	315
	852	178

Notation:

m = Number of training examples
x's = "input" variable / features
y's = "output" variable / "target" variable



Given a training set,

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

where $\theta'_{i}s$: Parameter

How to choose $\theta'_i s$





Idea : Choose θ_0 , θ_1 so that $h_{\theta}(x)$ is as close to y for our training examples (x, y).

So going with the idea, If hypothesis : $h_{\theta}(x) = \theta_0 + \theta_1 x$ We want to minimise the squared error function i.e. $(h_{\theta}(\mathbf{x}^{(i)}) - \mathbf{y}^{(i)})^2$ i.e the difference between the predicted value $(h_{\theta}(x))$ and the actual label (y) should be as minimum as possible.

Therefore, the cost function becomes

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

where summation (\sum) represents sum over my training set from i to m .

and where $\binom{1}{2m}$ represents that minimising one half of something shall give us same values of θ_0 and θ_1 as minimising the entire thing.

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: minimize $J(\theta_0, \theta_1)$

So, overall we want to find the value of $\theta_0 \& \theta_1$ that minimises the entire expression.

GRADIENT DESCENT ALGORITHM

- Gradient Descent algorithm is basically used to minimise some function J(say a cost function).
- Problem Setup
 - Have some function J(,)
 - Want to $\theta_0 \ \theta_1$
- Outline $\min_{\theta_0, \theta_1} J(\theta_0, \theta_1)$
 - Start with some & (say = 0, = 0)
 - Keep changing θ_0 the redurc θ_0 is the second seco

GRADIENT DESCENT ALGORITHM

repeat until convergence {

$$\theta_j \coloneqq \theta_j - \alpha \frac{\delta}{\delta \theta_j} J(\theta_0, \theta_1)$$
, (for $j = 0 \& j = 1$) }
where := is the assignment operator

- α is the learning rate that controls how big a step we can take downhill while creating descent
- Simultaneous update temp $0 := \theta_0 - \alpha \frac{\delta}{\delta \theta_0} J(\theta_0, \theta_1)$ temp $1 := \theta_1 - \alpha \frac{\delta}{\delta \theta_1} J(\theta_0, \theta_1)$ $\theta_0 := temp 0$ $\theta_1 := temp 1$

For eg :) say we have only one parameter, so, $\min_{\theta_1} J(\theta_1) \& \theta_1 \in \mathbb{R} \text{ , hence the GD equation becomes}$ $\theta_1 := \theta_1 - \alpha \frac{d}{d\theta_1} J(\theta_1)$

 $\frac{d}{d\theta_1} J(\theta_1)$ is the derivative term which basically denotes the slope of the line that is just tangent to the function $J(\theta_1)$ at the point θ_1







If 🕶 is too small , then gradient descent will converge slowly because very small and hence lot of steps are being taken before it gets anywhere close to the global minimum.



If \checkmark is too large, then gradient descent will overshoot the minimum because the steps taken are huge . It may fail to converge .

GRADIENT DESCENT FOR LINEAR REGRESSION

So, applying GD algorithm to minimise the squared error cost function i.e. $\min_{\theta_0, \theta_1} J(\theta_0, \theta_1)$ where $J(\theta_0, \theta_1) = \frac{1}{2m} (\lim_{i=1}^{m} \Sigma(h_{\theta}(x^{(i)}) - y^{(i)})^2)$ $h_{\theta}(x) = \theta_0 + \theta_1 x$ Calculating the derivative part,

$$\frac{\delta}{\delta\theta_{j}} J(\theta_{0}, \theta_{1}) = \frac{\delta}{\delta\theta_{j}} \frac{1}{2m} \left(\sum_{i=1}^{m} \Sigma(h_{\theta}(x^{(i)}) - y^{(i)})^{2} \right)$$

$$\frac{\delta}{\delta\theta_{j}} J(\theta_{0}, \theta_{1}) = \frac{\delta}{\delta\theta_{j}} \frac{1}{2m} \left(\sum_{i=1}^{m} \Sigma(\theta_{0} + \theta_{1}x - y^{(i)})^{2} \right)$$

GRADIENT DESCENT FOR LINEAR REGRESSION

So for
$$\theta_0$$
, or $\mathbf{j} = 0$:
 $\frac{\delta}{\delta \theta_0} J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=1}^{m} \Sigma(h_\theta(\mathbf{x}^{(i)}) - \mathbf{y}^{(i)})$

for
$$\theta_1$$
, or $\mathbf{j} = 1$:
 $\frac{\delta}{\delta \theta_1} J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=1}^{m} \Sigma(\mathbf{h}_{\theta}(\mathbf{x}^{(i)}) - \mathbf{y}^{(i)}) \cdot \mathbf{x}^{(i)}$

GRADIENT DESCENT FOR LINEAR REGRESSION

Now, by applying GD algorithm to the previous equation(derivative part),

repeat until convergence {

$$\begin{split} \theta_0 &\coloneqq \theta_0 - \alpha \frac{1}{m} \mathop{}_{i=1}^{m} \Sigma(h_\theta \left(x^{(i)} \right) - y^{(i)}) \\ \theta_1 &\coloneqq \theta_1 - \alpha \frac{1}{m} \mathop{}_{i=1}^{m} \Sigma(h_\theta \left(x^{(i)} \right) - y^{(i)}) \,. \, x^{(i)} \end{split}$$

 $update \theta_0 \& \theta_1$ simultaneously

FEATURE SCALING

- If one feature has a range say 0-2000 & another say 0-5 i.e. a 2000 to 5 ratio, then the contours of the cost function takes up a very skewed elliptical shape & the GD may end up taking a long time & can oscillate back & forth before it can finally find its way to the global minimum.
- A useful thing to do is scale the features i.e. the different features take on similar range of values.
- So, by feature scaling we get the features in a range of -1 to +1.

LINEAR REGRESSION WITH MULTIPLE VARIABLES

for multiple features,

$$\begin{split} \mathbf{h}_{\theta}(x) &= \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \dots \dots + \theta_n x_n \\ \text{For convenience of notation , define } \mathbf{x}_0 = 1 \end{split}$$

$$\mathbf{X} = \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ \vdots \\ \vdots \\ x_n \end{bmatrix} \in \mathbb{R}^{n+1}, \boldsymbol{\Theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \vdots \\ \vdots \\ \theta_n \end{bmatrix} \in \mathbb{R}^{n+1}$$

So, $\mathbf{h}_{\theta}(\mathbf{x}) = \theta_0 \mathbf{x}_0 + \theta_1 \mathbf{x}_1 + \theta_2 \mathbf{x}_2 + \dots + \theta_n \mathbf{x}_n$ $\therefore \quad \mathbf{h}_{\theta}(\mathbf{x}) = \mathbf{\theta}^{\mathsf{T}} \mathbf{x}$

POLYNOMIAL REGRESSION

• It allows to use the machinery of linear regression to fit very complicated even non-linear function to the data.



POLYNOMIAL REGRESSION

The form of hypothesis is :

$$\begin{split} h_{\theta}(x) &= \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 \\ h_{\theta}(x) &= \theta_0 + \theta_1(size) + \theta_2(size)^2 + \theta_3(size)^3 \\ \end{split}$$
where $x_1 = size$, $x_2 = (size)^2$, $x_3 = (size)^3$

• Feature Scaling is required here

NORMAL EQUATION

- GD is basically an algorithm for linear regression that takes multiple iterations to reach the global minimum or to minimise a cost function $I(\theta)$.
- Normal Equation is a method to solve for analytically i.e. in one step we get the optimum value.
- Intuition : If 1D , $(\theta \in \mathbb{R})$ $J(\theta) = a\theta^2 + b\theta + c$

To minimise the above quadratic function, we calculate the derivative of the function and set it to 0 i.e. $\frac{d}{d\theta} J(\theta) = 0$ & solve for Θ .

NORMAL EQUATION

Examples: m = 4.

	Size (feet ²)	Number of bedrooms	Number of floors	Age of home (years)	Price (\$1000)
x_0	x_1	x_2	x_3	x_4	y
1	2104	5	1	45	460
1	1416	3	2	40	232
1	1534	3	2	30	315
1	852	2	1	36	178
$X = \begin{bmatrix} 1\\1\\1\\1 \end{bmatrix}$	$ \begin{array}{rrrr} 2104 & 5 \\ 1416 & 3 \\ 1534 & 3 \\ 852 & 2 \\ \end{array} $	$ \begin{array}{cccc} 1 & 45 \\ 2 & 40 \\ 2 & 30 \\ 1 & 36 \end{array} $	y	$=\begin{bmatrix} 460\\232\\315\\178\end{bmatrix}$	

 $\boldsymbol{\theta} = (\boldsymbol{X}^T \boldsymbol{X})^{-1} \boldsymbol{X}^T \boldsymbol{Y}$

GRADIENT DESCENT Vs NORMAL EQUATION

GRADIENT DESCENT

- Need to choose *a*.
- Needs many iterations
- Works well even if n is large where n is the no. of features.

NORMAL EQUATION

- No need to choose *c*.
- Needs no iteration
- Slow if n is large because it needs to compute (X^TX)⁻¹ which is n*n matrix.