Linear Regression

Data (1) existing data
(2) future data

Linear Regression Big picture: use existing data (input and output are known) to predict future data (only knows input). Machine use the existing data to extract some model (knowledge): $\{\theta_0, \theta_1, ..., \theta_n\}$, where n is the dimension of input data. Gradient descent is a technique to help find optimal $\{\theta_0, \theta_1, ..., \theta_n\}$ through iteratively refinement.

Input: X (x_1, x_2, … x_n)
Output: y (also called "label")
Goals: We hope we can find a "**linear**" relationship between the input X and output data y.
Assumptions:

(1) The true relationship might not be "linear", but we just use linear to simplify the maths.

(2) Output may not be 1D, however, again to make the problem equation simpler, we choose 1D. Even the output is multi-dimensional, we can line them up in row-wise to convert it into 1D.

Course 1	Course 2	Course 3
[A, B, C, F]	[A, B, C, F]	[A, B, C, F]

Now y?
Original data: AAA ABA ABB AAB
y: [1, 2, 3, 4, …]

(3) Is the output data y known or not known?

Linear regression has two stages. - "**Prediction**"
Price of a house depends on many factors: x_1: area (size), x_2: year, x_3: location, x_4: floors, x_5: bedroom #...
X = [x_1, x_2, x_3, x_4, x_5, …]
y = price

stage I: Training Stage (both X and y are known, consider both as input)

You need to do some homework on existing houses to give you some idea about how much the house be cost depending on these factors. During this stage, you need collect those existing houses as training data. So for training data, the output (label) y is known! So in fact, it's input for training.
What is the output? It's the learned parameters $\{\theta_0, \theta_1, ..., \theta_n\}$, each parameter corresponds to the coefficient of [x_1, x_2, x_3, x_4, x_5, ...].

Stage II: Testing/Prediction Stage (X is known, y is unknown)
Is y known? Not known. For a new input data X:

$$f_\theta(X) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + ... + \theta_n x_n \longrightarrow y'$$

We hope y' (= $f_\theta(X)$) is as close to y as possible!

y = f(x) if for the function f(), the exponential part of x is at most 1, then we can call this function as linear function.

$y = 5x$
$y = 3x + 7$
$y = 2$ $(y = 2 + 0*x)$

We can extend dimension of X into multi-dimensional data, e.g. x is 3D: (x1, x2, x3)

X = (x_1, x_2, x_3), we use the lower case x with a number index as the value for the corresponding dimension.
y as the output data, we usually prefer it to be 1D for maths convenience.

Question: write an example of linear function between y and X

$y = 5x_1 + 2x_2 – 7x_3 + 10$
$y = 9\ x_2$

y – label (e.g. "face" = 1, "food" = 2, "building" = 3)
X = (x_1, x_2, x_3) - is the input image

The goal of training is to learn the values for the coefficients ($\theta_0, \theta_1, \theta_2, \theta_3$):

$y = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3$

For the whole classification problem domain, most of all the methods, including neural network, deep learning, …, is just to find an optimal values for the coefficients ($\theta_0, \theta_1, \theta_2, \theta_3, \ldots$) through training, an optimal estimation of these coefficients would produce an accurate prediction on y; otherwise, the result maybe not satisfactory.

Definition of Linear Regression

Below shows all about training stage!

Input Data: X (x₁, x₂, …) - it can be multidimensional

Output Data: y - by default, the output is 1D for mathematic computation reason. Sometimes the output data is also called "label"

Equation: between y and X is linear combination.

Goal: to learn an optimal model, more specifically, to learn those optimal parameters, $\theta_0, \theta_1, \theta_2, \ldots$ How many such parameters $\{\theta_j\}$ equals to the number of X dimension + 1. "j" indicates the dimension index.

Optimal:

Given a set of data pairs $\{<X^{(i)}, y^{(i)}>\} = <X^{(1)}, y^{(1)}>, <X^{(2)}, y^{(2)}>, \ldots, <X^{(m)}, y^{(m)}>$, coming in pairs. We will learn the parameters $\{\theta_0, \theta_1, \ldots, \theta_n\}$ that minimize the following equation:

One sample <X, y>: $\theta_0 X_0 + \theta_1 X_1 + \ldots + \theta_n X_n \xrightarrow{\text{predict}}$ y', we hope that y' is as to y as possible.

$$\boxed{\underset{\{\theta_0, \theta_1, \ldots, \theta_n\}}{\text{minimize}}\left(\frac{1}{m} \sum_{i=1,\ldots,m} \left(\underbrace{\boxed{(\theta_0 X_0^{(i)} + \theta_1 X_1^{(i)} + \ldots + \theta_n X_n^{(i)})}}_{\text{Prediction}} - \underbrace{\boxed{y^{(i)}}}_{\text{Truth}}\right)^2\right)} \text{ — Cost function } J(\theta)$$

The reason of use square instead of "absolute operator" is to allows conveniently compute the derivative, i.e. df()/dx $(5x^2 + 7x + 5)' = 10x + 7$

$(12x + 9)' = 12$

m – the number of training data

n – the number of input data X's dimension

i – indicates the input training data (sample) index, placing as superscript

j – indicates the dimension index, placing as subscript

Gradient Descent

This is a popularly used method to find the optimal parameters $\{\theta_0, \theta_1, \ldots, \theta_n\}$ to minimize the equation above. During the training stage, the above equation has the changing variables are $\{\theta_0, \theta_1, \ldots, \theta_n\}$ instead of X.

$$J(\theta) = \frac{1}{2m} \sum_{i=1,\ldots,m} ((\theta_0 X_0^{(i)} + \theta_1 X_1^{(i)} + \ldots + \theta_n X_n^{(i)}) - y^{(i)})^2$$

The reason we let $J(\theta)$ defined as 1/2m for maths reason that make the result of first derivative easier.

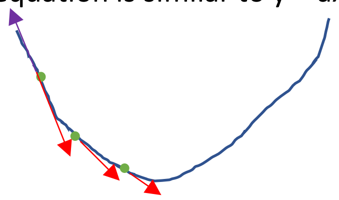
Idea: give a random guess of the possible values for $\{\theta_0, \theta_1, \ldots, \theta_n\}$, then plug in these values into $J(\theta)$ and get the error from the equation. Then try to modify these guessed values $\{\theta_0, \theta_1, \ldots, \theta_n\}$ by adjusting each individuals, denoted as $\{\Delta\theta_0, \Delta\theta_1, \ldots, \Delta\theta_n\}$, s. t. they are updated as:

$$\theta_0 = \theta_0 + \Delta\theta_0$$
$$\theta_1 = \theta_1 + \Delta\theta_1 \ldots$$
$$\theta_n = \theta_n + \Delta\theta_n$$

Hopefully, this update will allow the overall error decrease. Then we can repeat this process until the error gets small enough.

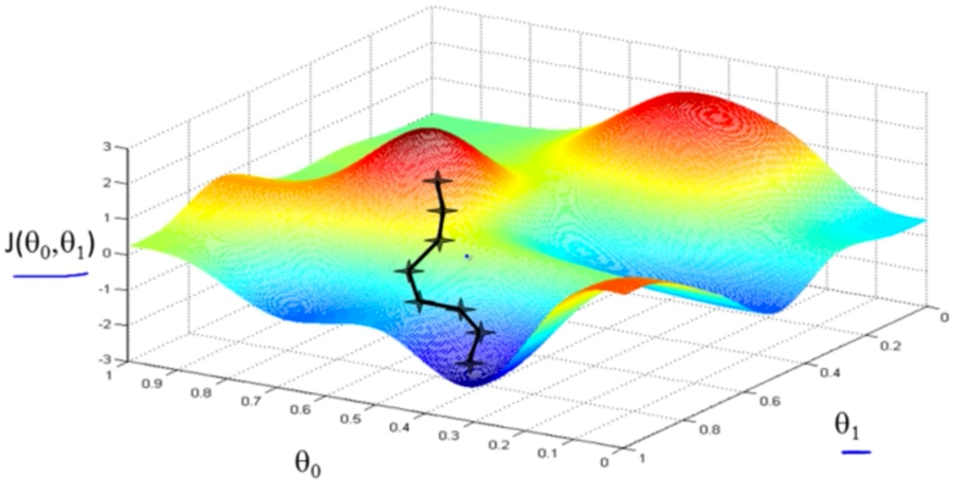
The Key is to compute $\{\Delta\theta_i\}$

Simple case: If there is only one theta to predict $\{\theta_0\}$, the error equation is just a quadratic function. The equation is similar to $y = ax^2 + bx + c$, where y is just the $J(\theta)$, and x is θ_0.



```
Repeat {
        $\theta_0 = \theta_0 - \frac{dJ(\theta)}{d\theta}$
}
```

More complex case: there are two parameters to predict $\{\theta_0, \theta_1\}$, which means the input data X is 1D. The above parabola curve is extended to a curly surface.



Repeat {

$$\theta_0 = \theta_0 - \frac{\partial J(\theta_0, \theta_1)}{\partial \theta_0}\alpha$$

$$\theta_1 = \theta_1 - \frac{\partial J(\theta_0, \theta_1)}{\partial \theta_1}\alpha$$

}

α - speed controller

Attention: You need to make sure for each iteration, the two parameters θ_0 and θ_1 are updated by using the partial derivative on the cost function J(θ_0, θ_1) based on the (θ_0, θ_1) values in last iteration, i.e. at iteration i, update θ_0 using the value of θ_0 and θ_1 from iteration (i-1); similarly update the θ_1 using the value of θ_0 and θ_1 from iteration (i-1). Don't used the currented updated θ_0

For a problem, we have the initial guess $\theta_0 = 5$, $\theta_1 = 12$. We also know the partial derivatives:

$\partial J(\theta_0, \theta_1)/ \quad \partial \theta_0 \;= 0.3\, \theta_0 + 0.6\, \theta_1$

$\partial J(\theta_0, \theta_1)/ \quad \partial \theta_1 \;= -0.2\, \theta_0 + 0.5\, \theta_1$

What are the (θ_0, θ_1) after 2 iterations? Assuming $\alpha = 1$

Iteration 1:

Theta_0 = theta_0 – partial_derivative(theta_0) = 5 – (0.3 * 5 + 12 * 0.6) = -3.7

Theta_1 = theta_1 – partial_derivative(theta_1) = 12 – (-0.2 * 5 + 12 * 0.5) = 7

Iteration 2:

Theta_0 = theta_0 – partial_derivative(theta_0) = -3.7 – (0.3 * (-3.7) + 0.6 * 7) = -4.248

Theta_1 = theta_1 – partial_derivative(theta_1) = 7 – ((-0.2) * (-3.7) + 0.5 * 7) = 2.76

…

Consider a problem, let's try to predict the price of a car. The price can depends on many factors, such as brand name, model, years, …. We just make a very simple case, let's just consider the car speed as the main factor for the price perdition. We have training data:

(100m/h, \$30,000)
(120m/h, \$45,000)
input (160m/h, \$60,000) output

Assume, we only have three data for training. Now, (1) could you write down the cost equation for this prediction? (2) How many parameters should I predict? (3) If I initially predict **θ** as {2, 5}, then what are the new **θ** in the next iteration?

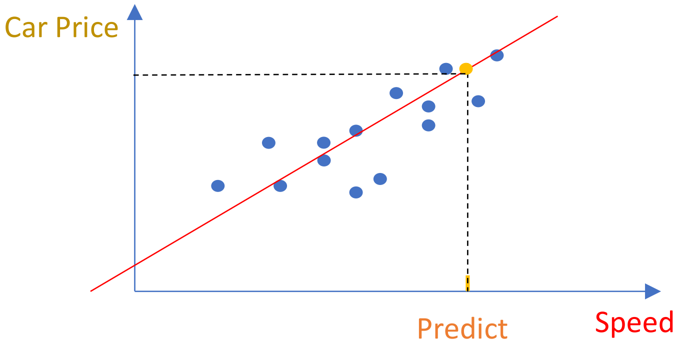
X – 1D, which is just the speed
y – 1D, which is the price
θ – 2D, which are $\{\theta_0, \theta_1\}$

$$J(\theta_0, \theta_1) = (1/6)\,[(\theta_0 + 100\theta_1 - 30000)^2 + (\theta_0 + 120\theta_1 - 45000)^2 + (\theta_0 + 160\theta_1 - 60000)^2]$$

$$\frac{\partial J(\theta_0, \theta_1)}{\partial \theta_0} = (1/6)\,[\,2 * \theta_0 + 2 * \theta_0 + 2 * \theta_0] = \theta_0 = 2$$

$$\frac{\partial J(\theta_0, \theta_1)}{\partial \theta_1} = (1/6)[2 * 100 * \theta_1 + 2 * 120 * \theta_1 + 2 * 160 * \theta_1]$$

The visualization of Linear Regression – fit a hyper line on the distributed points.


Car Price
Predict
Speed