machine learning_week2

Notebook:	artificial intelligence		
Created:	18/11/04 06:14	**Updated:**	18/11/07 23:35
Author:	2440027575@qq.com		
URL:	https://www.coursera.org/learn/machine-learning/supplement/WKgbA/multiple-features		

week.2

Multivariate Linear Regression

- **Intuition:**
  - denotation:
    $$\begin{aligned} x_j^{(i)} &= \text{value of feature } j \text{ in the } i^{th} \text{ training example} \\ x^{(i)} &= \text{the input (features) of the } i^{th} \text{ training example} \\ m &= \text{the number of training examples} \\ n &= \text{the number of features} \end{aligned}$$
  - hypothesis:
    $$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \cdots + \theta_n x_n$$
  - by matrix multiplication:
    $$h_\theta(x) = \begin{bmatrix} \theta_0 & \theta_1 & \dots & \theta_n \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix} = \theta^T x$$
    - tips: assume $x_0^{(i)} = 1 \text{ for } (i \in 1, \dots, m).$ (let the two vectors 'θ' and $x^{(i)}$ match each other element-wise (which = n+1)
- **Gradient Descent:**
  - base on knowledge of Multi-variable calculus, we can get the gradient vector by make partial derivative of our hypothesis for every variable.
  - here the algorithm:
    $$\begin{aligned} &\text{repeat until convergence: } \{ \\ &\quad \theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x_j^{(i)} \qquad \text{for j := 0...n} \\ &\} \end{aligned}$$
  - further more: instead of thinking of those 0~n as separate parameters, thinking them as (n+1)-dimensional vector.

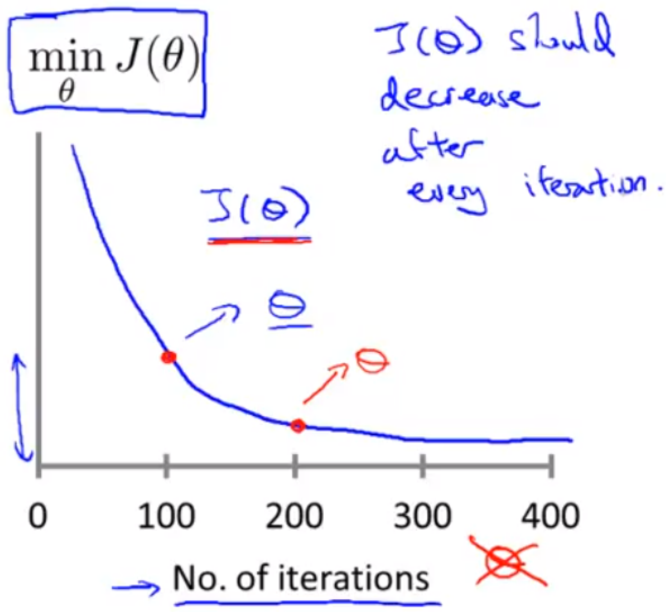
    $\cancel{\theta_0, \theta_1, \dots, \theta}$ vector θ
  - two tricks to make Gradient Descent work more efficency
    - 1 **feature scaling & mean normalization**
      - problem faced: x_i x_j have different ranges,like x_i range from 0 to 10, but x_j range from -1000 to 1000. This difference would need a long time for GD to converge.
      - Feature scaling involves dividing the input values by the range (i.e. the maximum value minus the minimum value) of the input variable, resulting in a new range of just 1.
      - Mean normalization involves subtracting the average value for an input variable from the values for that input variable resulting in a new average value for the input variable of just

zero.

- $$x_i := \frac{x_i - \mu_i}{s_i}$$
- Where μ_i is the average of all the values for feature (i) and s_i is the range of values (max - min), or s_i is the standard deviation.

- when scaling we don't need to fit the range accurately, just close to it is acceptable.

- 2 **debugging gradient descent & choose alpha:**
  - plotting: Make a plot with *number of iterations* on the x-axis.



    - e g.
    - observe the plot.
  - choose alpha:use plots to find the suitable alpha. Usually try each value being about 3x bigger than previous one.
    - Like alpha= 0.001 0.003 0.01 0.03 0.1 0.3 1······and make a plot with each of the value to choose the best.

- **features and Polynomial Regression**
  - **create new features:**we can combine multiple features into one to make a new features which has better relevance to our prediction.
  - **Polynomial Regression：** more function to fit data.
    - **change the behavior or curve** of our hypothesis function by making it a quadratic, cubic or square root function (or any other form).
    - e g. $$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 \sqrt{x_1}$$
    - caution:! **feature scaling** is really important now! Think that we have a n polynomial regression, if the x hadn't been scale, the value of power(x,n) may be gigantic.

Computing Parameters Analytically

- **Normal Equation.**
  - hey guys, still remember what u learn in Linear Algebra? Here we don't want to iterate but to solve the vector θ by matrix.
  - supposed that we make each training example as a (n+1)-dimensional vector(because it has n features. And we add 1 as the first dimension), then we **design matrix X** ,then our problem is trans into **solve a system of linear equations.**

	Size (feet²)	Number of bedrooms	Number of floors	Age of home (years)
x_0	x_1	x_2	x_3	x_4
1	2104	5	1	45
1	1416	3	2	40
1	1534	3	2	30
1	852	2	1	36

$$X = \begin{bmatrix} 1 & 2104 & 5 & 1 & 45 \\ 1 & 1416 & 3 & 2 & 40 \\ 1 & 1534 & 3 & 2 & 30 \\ 1 & 852 & 2 & 1 & 36 \end{bmatrix}$$

    - X is like
  - formula:
    - $$\theta = (X^T X)^{-1} X^T y$$
    - Q:why just compute theta equal to **(x)^(-1)*y**? A:because the matrix X may not always invertible. By (X^TX) make sure we get a invertible matrix.
    - remember: a matrix X is invertible which Is equivalent to det(X) != 0
    - so careful X^TX may be non-invertible when:
      1. Redundant features.where two features are very closely related (i.e. they are linearly dependent)
      2. Too many features (e.g. m ≤ n). In this case, delete some features or use "regularization" (to be explained in a later lesson).
  - what the computer need
  - comparison with GD:
    -

Gradient Descent	Normal Equation
Need to choose alpha	No need to choose alpha
Needs many iterations	No need to iterate
O (kn^2)	O (n^3), need to calculate inverse of X^TX
Works well when n is large	Slow if n is very large

    - if n>10,000, we should use GD.

Sum Up:

- this week we still play with linear regression, but it's a multivariate version. And we come up with two solution,one important concept
  - one **concept: VECTOR.** We should consider every values as a vector or a matrix first(which mean i need to review Linear Algebra -laugh-). By vector operation we can compute faster and simultaneous.
  - solution one is GD(gradient descent) with **more variate**. After review multivariate calculus and linear algebra, we should know that the **essence** of GD is using vector to **simultaneous compute partial derivative.**

        - three tricks: 1. feature scaling 2. polynomial regression to fit data 3.plot the cost function to debug.
    - solution two is **Normal Equation,** which suited for small data set and give final theta without loop.