Machine Learning (Andrew Ng) Notes #2

标签（空格分隔）： Coursera ML DL AI Andrew_Ng Stanford

A. Multivariable linear regression

定义: 训练数据有多组feature，进行回归

Hypothesis Function:

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_n x_n$$

For convenience, ***define***: $x_0 = 1$, i.e. $x_0^i = 1$ for each i

Therefore the hypothesis function becomes:

$$h_\theta(x) = \theta_0 x_0 + \theta_1 x_1 + \ldots + \theta_n x_n = \theta^T x$$

θ^T is the transpose of θ_n

Notation:

x_j^i = value of feature j in the ith training sample

x^i = the column vector of all the feature inputs of the ith training example

m = the number of training examples

n = | x^i | -1; (number of features, starting from 0 hence the -1)

Parameters:

$$\theta_0, \theta_1 \ldots \theta_n$$

Cost Function:

$$J(\theta_0, \theta_1, \ldots, \theta_n) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^i) - y^i)^2$$

Taking parameters as θ **vector**, the cost function is:

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (\theta^T x^i - y^i)^2$$

Goal:
Minimize:

$$J(\theta)$$

B. Gradient Descent algorithm

Definition:

The idea is to use a fixed jumping distance * slope ($\frac{\partial}{\partial\theta_j}$) to itereate to the local minimum of θ_j.

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

$$\alpha = learning\_rate$$

Notes: Need to simultaneously update all $\theta_0 \ldots \theta_j$

C. Applying gradient descent into cost function

$$\theta_0 := \theta_0 - \alpha\frac{1}{m}\sum_{i=1}^{m}(h_0(x^i) - y^i)x_0^i$$

$$\theta_j := \theta_1 - \alpha\frac{1}{m}\sum_{i=1}^{m}(h_0(x^i) - y^i) * x_j^i$$

The cost function can also be directly solved with **normal equation method**, however, gradient descent will scale better at large data sets.

D. Feature scaling & mean normalization

For calculation purpose, ideally every feature should be in the -1 <= x^i <= 1 range

Or to replace x^i with $\frac{x_i - \mu_i}{s^i}$, where μ_i is the average of x_i in the training samples and s_i is the range of x_i (max - min)

E. Gradient descent vs normal equation

Normal equation needs to solve for X^TX, depending on the data size, inverting a matrix with 10,000+ size can be demanding in calculation capacity.

F. Octave commands

```
#Basic function
>> a == b  % compare
```

```
>> a ~= b  % not equal
>> a && b  % and
>> a || b  % or
>> a = ones(a,b) % a * b matrix with ones
>> a = A(2,:) % Everything in the second row
>> rand(a,b) % a*b matrix with random numbers
>> eye(a)  % a by a identity matrix
>> help rand % shows functions help
>> a = rand(3,4);
>> hist(a) % generate histogram in GUI
>> A = [A, [vector]] % add a column vector to the right
>> A(:) % put all elements of A into a single vector
>>
>>
# Moving data around
>> size(a) % give the dimension of matrix A
>> length(a) %longest dimension, mostly used in vectors
>> load features.dat %load data file
>> load('features.dat') %file name can be given in string
>> who % prints varaiables in the current scope
>> whos % better formating with variable info
>> clear variable % clears varaible in the current env
>> v = priceY(1:10) % v = the first 10 elements
>> save hello.mat v; % save v data into new file
>> save hello.txt v -ascii % save v as a text file
>>
>>
# Computing data
>> A * B % Matrix multiplication
>> A .* B % Product of each item in matrices of same size
>> A .^ 2 % ^2 on each element
>> abs(V) % absolute elements
>> V + ones(length(v),1) % Add one to each element
>> A' % A transpose
>> [val, ind] = max(A) % Gives the max value / pos of A
>> [r, c] = find(A>=7) % Gives column / row of >=7 element
>> sum(a) % sum of all elements by column
>> prod(a) % product of all elements by column
>> ceil(a) % 向上取整
>> max(A, [], 1) % returns column maximums
>> max(A, [], 2) % returns row maximums
>> sum(A, 1) % sum by column
>> sum(A, 2) % sum by row
>> sum(sum(A.*eye(length(A)))) % diagonal addition
>> flipud(eye(9)) % flip identity matrix
>> pinv(A) % inverting a given matrix A
>>
>>
```

```
# Plotting data
>> t = [0:0.01:0.98];
>> y1 = sin(2*pi*4*t);
>> plot(t,y1)  % Generate plot
>> hold on;  % Generate plot on the same canvas
>> plot(t,y2,'r');  % Generate second plot in red
>> xlabel('time);
>> ylabel('value);
>> legend('sin', 'cos');
>> title('my plot');
>> print -dpng 'myPlot.png'
>> close
>> figure(1); plot(t,y1);  % Plot separate figures
>> figure(2); plot(t,y2);
>> subplot(1,2,1); % Divide canvas by 1 * 2 and access the 1st
>> axis([0.5 1 -1 1]) % sets the x/y axis range for current plot
>> clf % Clear canvas
>> A = magic(5)
>> imagesc('A') % Color plot the matrix, VERY COOL!
>> imagesc('A'), colorbar, colormap gray; % Gray with bar
>> % using , instead of ; lets the terminal excute multiple commands at the same time
>>
>>
# Control statement
>> v = zeros(10,1)
>> for i = 1:10,
>> v(i) = 2^i
>> end;
>> % array subscript starts from 1 instead of 0
>>
>>
>> i = 1; %Octave doesnt care about indention
>> while true % using true instead of True
>>      v(i) = 999;
>>      i = i+1;
>>      if i == 6, % use , to indicate :
>>          break;
>>      end; % each conditional statement needs an 'end'
>> end;
>>
>> function J = costFunctionJ(X, y, theta)
>> m = size(X, 1);
>> predictions = X*theta
>> sqrErrors = (predictions - y) .^ 2;
>> J = 1/(2 * m) * sum(sqrErrors);
```