Week 2 Lecture Notes

机器学习：多变量线性回归 - ML:Linear Regression with Multiple Variables

多变量线性回归（Linear regression with multiple variables）也称为“多元线性回归（multivariate linear regression）”。

现在我们引入一些符号，现在可以有任意数量的变量了：

$$
\begin{aligned}
x_j^{(i)} &= \text{第}i\text{个训练数据中，特征}j\text{的值} \\
x^{(i)} &= \text{第}i\text{个训练数据，也是一个列向量} \\
m &= \text{训练数据的规模} \\
n &= \left| x^{(i)} \right|; (\text{每个训练数据包函的特征数})
\end{aligned}
$$

现在定义多变量形式的假设函数如下，匹配多个特征：

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \cdots + \theta_n x_n$$

为了给出一些直觉上的理解，我们可以将θ_0作为房屋的起步价，作θ_1为每平方米的价格，θ_2作为每层价格，等等。x_1对应房子有多少平方米，x_2对应房子的有几层，等等。

利用矩阵乘法的定义，我们的多变量假设函数可以简明地表示为：

$$h_\theta(x) = \begin{bmatrix} \theta_0 & \theta_1 & \cdots & \theta_n \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ \cdots \\ x_n \end{bmatrix} = \theta^T x$$

这是针对一个训练数据的假设函数的向量化；参见向量化的课程以了解更多。

备注：在本课程中，出于方便的原因，吴恩达先生假设 $x_0^{(i)} = 1 \text{ 对于 } (i \in 1, \ldots, m)$

[注意：为了用θ和x进行矩阵运算，对于所有的i，使$x_0^{(i)} = 1$，这使得两个向量“θ”和$x_{(i)}$在元素上彼此匹配（即，具有相同元素的数量n+1）。]

训练数据以X行存储，例如：

$$X = \begin{bmatrix} x_0^{(1)} & x_1^{(1)} \\ x_0^{(2)} & x_1^{(2)} \\ x_0^{(3)} & x_1^{(3)} \end{bmatrix}, \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \end{bmatrix}$$

你可以把这个假设作为一个列向量（m x 1）来计算：

$h_\theta(X) = X\theta$

对于笔记的剩余部分和其他讲稿，X表示训练数据以行存储的矩阵。

代价函数 - Cost function

For the parameter vector θ (of type $\mathbb{R}^{n+1}$ or in $\mathbb{R}^{(n+1)\times 1}$, the cost function is:

对于参数向量θ（$\mathbb{R}^{n+1}$或$\mathbb{R}^{(n+1)\times 1}$ }中，代价函数为：

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

矢量化的版本是：

$$J(\theta) = \frac{1}{2m}(X\theta - \vec{y})^T(X\theta - \vec{y})$$

$\vec{y}$ 表示所有y值组成的列向量。

多变量的梯度下降 - Gradient Descent for Multiple Variables

梯度下降方程本身就是通用的形式：

$$\theta_j := \theta_j - \alpha\frac{d}{d\theta_j}J(\theta_0, \theta_1) \quad (for\ j = 0, 1, 2 \ldots\ n)$$

我们只需要重复计算“N”个特征：

$$\begin{aligned}
&\text{repeat until convergence} : \{ \\
&\theta_0 := \theta_0 - \alpha\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)}) \cdot x_0^{(i)} \\
&\theta_1 := \theta_1 - \alpha\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)}) \cdot x_1^{(i)} \\
&\theta_2 := \theta_2 - \alpha\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)}) \cdot x_2^{(i)} \\
&\ldots \\
&\}
\end{aligned}$$

也可以写成：

$$\begin{aligned}
&\text{repeat until convergence} : \{ \\
&\theta_j := \theta_j - \alpha\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)}) \cdot x_j^{(i)} \quad for\ j := 0 \ldots n \\
&\}
\end{aligned}$$

公式推导（译者注）

$\frac{\partial}{\partial\theta_j}J(\theta)$ 是分别对每个θ_j求偏导数

$$\frac{\partial}{\partial\theta_j}J(\theta) = \frac{\partial}{\partial\theta_j} \cdot \frac{1}{2m} \cdot \sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$$

推导：

$$
\begin{aligned}
&\frac{\partial}{\partial \theta_j} J(\theta) \\
=&\frac{\partial}{\partial \theta_j} \cdot \frac{1}{2m} \cdot \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 \\
=&\frac{1}{2m} \cdot 2 \cdot \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \cdot \frac{\partial}{\partial \theta_j} (h_\theta(x^{(i)}) - y^{(i)}) \\
=&\frac{1}{m} \cdot \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \cdot \frac{\partial}{\partial \theta_j} (\theta_0 x_0^{(i)} + \theta_1 x_1^{(i)} + \theta_2 x_2^{(i)} + \ \dots\ \theta_n x_n^{(i)} - y^{(i)}) \\
=&\frac{1}{m} \cdot \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x_j^{(i)}
\end{aligned}
$$

那么梯度下降公式整理为：

$$
\theta_j := \theta_j - \alpha \cdot \frac{1}{m} \cdot \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x_j^{(i)} \quad (for\ j = 0, 1, 2 \dots\ n)
$$

矩阵表示 - Matrix Notation

梯度下降规则可以表示为：

$\theta := \theta - \alpha \nabla J(\theta)$

$\nabla J(\theta)$ 是一个列向量：

$$
\nabla J(\theta) = \begin{bmatrix} \frac{\partial J(\theta)}{\partial \theta_0} \\ \frac{\partial J(\theta)}{\partial \theta_1} \\ \dots \\ \frac{\partial J(\theta)}{\partial \theta_2} \end{bmatrix}
$$

梯度下降的第j个分量是两个项乘积的求和：

$$
\begin{aligned}
\frac{\partial J(\theta)}{\partial \theta_j} &= \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x_j^{(i)} \\
&= \frac{1}{m} \sum_{i=1}^{m} x_j^{(i)} \cdot (h_\theta(x^{(i)}) - y^{(i)})
\end{aligned}
$$

有时，两个项乘积再求和可以表示为一个向量的转置与另一向量的乘积。

对于i = 1，...，m，$x_j^{(i)}$ 表示训练集X的第j列，共m个元素。

另一部分 $\left(h_\theta(x^{(i)}) - y^{(i)}\right)$ 是假设$h_\theta(x^{(i)})$与实际值$y^{(i)}$的差，也是一个矢量. 重写$\frac{\partial J(\theta)}{\partial \theta_j}$部分，于是有：

$$
\begin{aligned}
\frac{\partial J(\theta)}{\partial \theta_j} &= \frac{1}{m} \vec{x}_j^{\,T} (X\theta - \vec{y}) \\
\nabla J(\theta) &= \frac{1}{m} x^T (X\theta - \vec{y})
\end{aligned}
$$

最后，梯度下降规则的矩阵表示法（矢量化）是：

$$\theta := \theta - \frac{\alpha}{m} X^T (X\theta - \vec{y})$$

特征归一化 - Feature Normalization

使每个变量保持在大致相同的范围内，可以加速梯度下降。这是因为θ在小范围内会快速下降，而在大范围内会缓慢下降，所以当变量非常不均匀时，θ会振荡，低效率的下降到最优。

防止这种情况的方法是修改输入变量的范围，使它们大致相同。理想情况：

$-1 \le x_{(i)} \le 1$

或者

$-0.5 \le x_{(i)} \le 0.5$

因为我们只是加快计算速度，所以这个范围并不是严格要求。所以目标是把所有的输入变量大致映射到一个范围内，以上列出的范围即可，或者任取一个。

有两个步骤：**特征缩放**和**均值归一化**。

均值归一化：将每个输入变量的值中减去平均值，得到新的值，这样新值的平均值就是0。

特征缩放：将每个输入值除以所有变量的范围（即，最大值减去最小值），从而新的值分布范围1。

这两个步骤用公式表示：

$x_i := \frac{x_i - \mu_i}{s_i}$

其中μ_i是所有特征(i)的**平均值**，s_i是值的范围(max-min)，或者是标准差。

注意，除以范围或除以标准差，会不同的结果。本课程的测验（Quiz）用范围，而编程练习（Programming Exercise）使用标准差。

例：x_i 代表房价， 范围100到2000，平均值1000， 那么， $x_i := \frac{price - 1000}{1900}$.

关于梯度下降的提示 - Gradient Descent Tips

Debugging gradient descent. Make a plot with *number of iterations* on the x-axis. Now plot the cost function, J(θ) over the number of iterations of gradient descent. If J(θ) ever increases, then you probably need to decrease α.

Automatic convergence test. Declare convergence if J(θ) decreases by less than E in one iteration, where E is some small value such as 10−3. However in practice it's difficult to choose this threshold value.

It has been proven that if learning rate α is sufficiently small, then J(θ) will decrease on every iteration. Andrew Ng recommends decreasing α by multiples of 3.

调试梯度下降。用迭代次数表示x轴，绘制一个图。现在，对应不同的迭代次数，绘制代价函数J(θ)。如果J(θ)在增加，那么梯度下降不收敛，你可能需要减少α。

程序收敛测试。假设E是10～3的一些小值，如果J(θ)在一次迭代中减小的值小于E，则可以认为已经收敛。然而，在实际应用中很难选择这个阈值。

已经证明，如果学习速率α足够小，则在每次迭代中J(θ)都会减小。吴恩达教授建议每次以3倍将α减小或增大。

特征与多项式回归 - Features and Polynomial Regression

我们可以用几种不同的方式来改进我们的特征和假设函数的形式。

我们可以将多个特征组合为一个。例如，我们可以将x_1和x_2结合到x_3中，作为新特征，$x_3 = x_1 \cdot x_2$。

多项式回归 - Polynomial Regression

如果假设函数不能很好地拟合数据，那么假设函数不一定必须是线性的（直线）。

我们可以使假设函数成为二次、三次或平方根函数（或任何其他形式）来**改变它的图像或曲线形式**。

例如，如果我们的假设函数是$h_\theta(x) = \theta_0 + \theta_1 x_1$，那么我们可以基于$x_1$创建额外的特征，以得到二次函数$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_1^2$或三次函数$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_1^2 + \theta_3 x_1^3$。

在三次函数的版本中，我们创建了新的特征x_2和x_3，其中$x_2 = x_1^2$和$x_3 = x_1^3$。

要使其成为平方根函数，我们也可以：$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 \sqrt{x_1}$。

请注意，在“Features and Polynomial Regression”视频的2:52和6:22中，吴恩达教授所讨论的“doesn't ever come back down”曲线指的是使用sqrt()函数（用实紫线表示）的假设函数，而不是使用$size^2$（用蓝色点的线表示）的假设函数。如果θ_2是负的，假设函数的二次形式将具有蓝色虚线所示的形状，。

One important thing to keep in mind is, if you choose your features this way then feature scaling becomes very important.

要记住的一件重要事情是，如果你这样添加新的特征，那么特征缩放就变得非常重要。

eg. if x_1 has range 1 - 1000 then range of x_1^2 becomes 1 - 1000000 and that of x_1^3 becomes 1 - 1000000000.

例如：如果x_1的范围为1—1000，则x_1^2的范围变成1～1000000，x_1^3的范围变成1—1000000000。

正规方程 - Normal Equation

“正规方程”是一种**不用迭代**就能求出最优θ的方法。

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

正规方程也**不需要**进行特征缩放。

正规方程的数学证明需要线性代数的知识，而且相当复杂，所以不必担心细节。

有兴趣的人可以在这些链接上找到证明：

https://en.wikipedia.org/wiki/Linear_least_squares_(mathematics)

http://eli.thegreenplace.net/2014/derivation-of-the-normal-equation-for-linear-regression

下面是梯度下降与正规方程的比较：

梯度下降	正规方程
需要挑选α值	No need to choose alpha
需要多次迭代	No need to iterate
时间复杂度O (kn^2)	时间复杂度O (n^3)，需要计算X^TX的逆矩阵
当n很大的时候，性能依然不错	n很大的时候，会变慢

用正态方程计算逆矩阵的复杂度为$\mathcal{O}(n^3)$。因此，如果我们有大量的特征，正规方程将会非常缓慢。在实践中，当N超过10000时，就该考虑使用梯度下降了。

正规方程的可逆性 - Normal Equation Noninvertibility

在Octave中求正规方程时，我们希望使用“pinv”函数而不是“inv”。

X^TX may be **noninvertible**. The common causes are:

- Redundant features, where two features are very closely related (i.e. they are linearly dependent)
- Too many features (e.g. m ≤ n). In this case, delete some features or use "regularization" (to be explained in a later lesson).

Solutions to the above problems include deleting a feature that is linearly dependent with another or deleting one or more features when there are too many features.

X^TX可能是不可逆的。常见的原因有：

- 有冗余特征，其中两个特征非常密切相关（即它们是线性相关的）。
- 有太多的特征（例如m ≤ n）。在这种情况下，删除一些特征或使用“正则化（regularization）”（在后面的课中解释）。

上述问题的解决方案包括删除与另一个特性线性相关的特性，或者当特性太多时删除一个或多个特性。

（译者注：后文均为Octave的教程，Octave可以认为是Matlab的开源免费版，语法和基本函数库相同，考虑到Matlab中文资料已经够多，就不再翻译。）

机器学习：Octave教程 - ML:Octave Tutorial

基本操作 - Basic Operations

```
%% Change Octave prompt
PS1('>> ');
%% Change working directory in windows example:
cd 'c:/path/to/desired/directory name'
%% Note that it uses normal slashes and does not use escape characters for
the empty spaces.
```

```
%% elementary operations
5+6
3-2
5*8
1/2
2^6
1 == 2 % false
1 ~= 2 % true.  note, not "!="
1 && 0
1 || 0
xor(1,0)


%% variable assignment
a = 3; % semicolon suppresses output
b = 'hi';
c = 3>=1;

% Displaying them:
a = pi
disp(a)
disp(sprintf('2 decimals: %0.2f', a))
disp(sprintf('6 decimals: %0.6f', a))
format long
a
format short
a


%%  vectors and matrices
A = [1 2; 3 4; 5 6]

v = [1 2 3]
v = [1; 2; 3]
v = 1:0.1:2   % from 1 to 2, with stepsize of 0.1. Useful for plot axes
v = 1:6       % from 1 to 6, assumes stepsize of 1 (row vector)

C = 2*ones(2,3) % same as C = [2 2 2; 2 2 2]
w = ones(1,3)   % 1x3 vector of ones
w = zeros(1,3)
w = rand(1,3) % drawn from a uniform distribution
w = randn(1,3)% drawn from a normal distribution (mean=0, var=1)
w = -6 + sqrt(10)*(randn(1,10000));  % (mean = -6, var = 10) - note: add
the semicolon
hist(w)    % plot histogram using 10 bins (default)
hist(w,50) % plot histogram using 50 bins
% note: if hist() crashes, try "graphics_toolkit('gnu_plot')"
```

```
I = eye(4)   % 4x4 identity matrix

% help function
help eye
help rand
help help
```

操作数据 - Moving Data Around

Data files used in this section: featuresX.dat, priceY.dat

```
%% dimensions
sz = size(A) % 1x2 matrix: [(number of rows) (number of columns)]
size(A,1) % number of rows
size(A,2) % number of cols
length(v) % size of longest dimension



%% loading data
pwd   % show current directory (current path)
cd 'C:\Users\ang\Octave files'  % change directory
ls    % list files in current directory
load q1y.dat   % alternatively, load('q1y.dat')
load q1x.dat
who   % list variables in workspace
whos  % list variables in workspace (detailed view)
clear q1y      % clear command without any args clears all vars
v = q1x(1:10); % first 10 elements of q1x (counts down the columns)
save hello.mat v;  % save variable v into file hello.mat
save hello.txt v -ascii; % save as ascii
% fopen, fread, fprintf, fscanf also work  [[not needed in class]]

%% indexing
A(3,2)  % indexing is (row,col)
A(2,:)  % get the 2nd row.
        % ":" means every element along that dimension
A(:,2)  % get the 2nd col
A([1 3],:) % print all  the elements of rows 1 and 3

A(:,2) = [10; 11; 12]     % change second column
A = [A, [100; 101; 102]]; % append column vec
A(:) % Select all elements as a column vector.

% Putting data together
A = [1 2; 3 4; 5 6]
B = [11 12; 13 14; 15 16] % same dims as A
C = [A B]  % concatenating A and B matrices side by side
C = [A, B] % concatenating A and B matrices side by side
```

```
C = [A; B] % Concatenating A and B top and bottom
```

计算数据 - Computing on Data

```
%% initialize variables
A = [1 2;3 4;5 6]
B = [11 12;13 14;15 16]
C = [1 1;2 2]
v = [1;2;3]

%% matrix operations
A * C  % matrix multiplication
A .* B % element-wise multiplication
% A .* C  or A * B gives error - wrong dimensions
A .^ 2 % element-wise square of each element in A
1./v   % element-wise reciprocal
log(v)  % functions like this operate element-wise on vecs or matrices
exp(v)
abs(v)

-v  % -1*v

v + ones(length(v), 1)
% v + 1  % same

A'  % matrix transpose

%% misc useful functions

% max  (or min)
a = [1 15 2 0.5]
val = max(a)
[val,ind] = max(a) % val -  maximum element of the vector a and index -
index value where maximum occur
val = max(A) % if A is matrix, returns max from each column

% compare values in a matrix & find
a < 3 % checks which values in a are less than 3
find(a < 3) % gives location of elements less than 3
A = magic(3) % generates a magic matrix - not much used in ML algorithms
[r,c] = find(A>=7)  % row, column indices for values matching comparison

% sum, prod
sum(a)
prod(a)
floor(a) % or ceil(a)
max(rand(3),rand(3))
max(A,[],1) -  maximum along columns(defaults to columns - max(A,[]))
```

```
max(A,[],2) - maximum along rows
A = magic(9)
sum(A,1)
sum(A,2)
sum(sum( A .* eye(9) ))
sum(sum( A .* flipud(eye(9)) ))


% Matrix inverse (pseudo-inverse)
pinv(A)        % inv(A'*A)*A'
```

用数据绘图 - Plotting Data

```
%% plotting
t = [0:0.01:0.98];
y1 = sin(2*pi*4*t);
plot(t,y1);
y2 = cos(2*pi*4*t);
hold on;  % "hold off" to turn off
plot(t,y2,'r');
xlabel('time');
ylabel('value');
legend('sin','cos');
title('my plot');
print -dpng 'myPlot.png'
close;           % or,  "close all" to close all figs
figure(1); plot(t, y1);
figure(2); plot(t, y2);
figure(2), clf;  % can specify the figure number
subplot(1,2,1);  % Divide plot into 1x2 grid, access 1st element
plot(t,y1);
subplot(1,2,2);  % Divide plot into 1x2 grid, access 2nd element
plot(t,y2);
axis([0.5 1 -1 1]);  % change axis scale

%% display a matrix (or image)
figure;
imagesc(magic(15)), colorbar, colormap gray;
% comma-chaining function calls.
a=1,b=2,c=3
a=1;b=2;c=3;
```

控制语句for, while, if - Control statements: for, while, if statements

```
v = zeros(10,1);
```

```
for i=1:10,
    v(i) = 2^i;
end;
% Can also use "break" and "continue" inside for and while loops to control
execution.

i = 1;
while i <= 5,
  v(i) = 100;
  i = i+1;
end

i = 1;
while true,
  v(i) = 999;
  i = i+1;
  if i == 6,
    break;
  end;
end

if v(1)==1,
  disp('The value is one!');
elseif v(1)==2,
  disp('The value is two!');
else
  disp('The value is not one or two!');
end
```

函数 - Functions

To create a function, type the function code in a text editor (e.g. gedit or notepad), and save the file as "functionName.m"

Example function:

```
function y = squareThisNumber(x)

y = x^2;
```

To call the function in Octave, do either:

1) Navigate to the directory of the functionName.m file and call the function:

```
% Navigate to directory:
cd /path/to/function

% Call the function:
functionName(args)
```

2) Add the directory of the function to the load path and save it:**You should not use addpath/savepath for any of the assignments in this course. Instead use 'cd' to change the current working directory. Watch the video on submitting assignments in week 2 for instructions.**

```
 % To add the path for the current session of Octave:
 addpath('/path/to/function/')

 % To remember the path for future sessions of Octave, after executing
addpath above, also do:
 savepath
```

Octave's functions can return more than one value:

```
function [y1, y2] = squareandCubeThisNo(x)
y1 = x^2
y2 = x^3
```

Call the above function this way:

```
[a,b] = squareandCubeThisNo(x)
```

矢量化 - Vectorization

Vectorization is the process of taking code that relies on **loops** and converting it into **matrix operations**. It is more efficient, more elegant, and more concise.

As an example, let's compute our prediction from a hypothesis. Theta is the vector of fields for the hypothesis and x is a vector of variables.

With loops:

```
prediction = 0.0;
for j = 1:n+1,
  prediction += theta(j) * x(j);
end;
```

With vectorization:

```
prediction = theta' * x;
```

If you recall the definition multiplying vectors, you'll see that this one operation does the element-wise multiplication and overall sum in a very concise notation.

编写和提交编程作业 - Working on and Submitting Programming Exercises

1. Download and extract the assignment's zip file.
2. Edit the proper file 'a.m', where a is the name of the exercise you're working on.
3. Run octave and cd to the assignment's extracted directory
4. Run the 'submit' function and enter the assignment number, your email, and a password (found on the top of the "Programming Exercises" page on coursera)

视频课程索引 - Video Lecture Table of Contents

Basic Operations

```
0:00    Introduction
3:15    Elementary and Logical operations
5:12    Variables
7:38    Matrices
8:30    Vectors
11:53   Histograms
12:44   Identity matrices
13:14   Help command
```

Moving Data Around

```
0:24    The size command
1:39    The length command
2:18    File system commands
2:25    File handling
4:50    Who, whos, and clear
6:50    Saving data
8:35    Manipulating data
12:10   Unrolling a matrix
12:35   Examples
14:50   Summary
```

Computing on Data

```
0:00    Matrix operations
0:57    Element-wise operations
4:28    Min and max
5:10    Element-wise comparisons
5:43    The find command
6:00    Various commands and operations
```

Plotting data

```
0:00    Introduction
0:54    Basic plotting
2:04    Superimposing plots and colors
3:15    Saving a plot to an image
4:19    Clearing a plot and multiple figures
4:59    Subplots
6:15    The axis command
6:39    Color square plots
8:35    Wrapping up
```

Control statements

```
0:10    For loops
1:33    While loops
3:35    If statements
4:54    Functions
6:15    Search paths
7:40    Multiple return values
8:59    Cost function example (machine learning)
12:24   Summary
```

Vectorization

```
0:00    Why vectorize?
1:30    Example
4:22    C++ example
5:40    Vectorization applied to gradient descent
9:45    Python
```