Multiple Variable Linear Regression

Shen Hengheng

2017

该笔记是来自 Andrew Ng 的 Machine Learning 课程的第二周: **多变量线性回归**的课堂记录，其实就是将单变量线性回归推广到多变量中去，主要讲解了以下几个内容:

- 模型的表示
- 梯度下降算法
- 运算中的实用技巧
    1. 特征缩放
    2. 学习率 α

0.1 模型表达

在单变量线性回归的房屋预测模型例子中，我们忽略了一个显示，往往购房者不仅仅考虑一个因素，因此多变量（特征）的引入是必需的，比如房屋的价格往往和卧室的数量（numbers of bedrooms），有多少个楼层 (numbers of floors)，房屋的年龄 (age of home) 等等，我们选取这些特征来预测价格，实例数据如图1

Size (feet²) x_1	Number of bedrooms x_2	Number of floors x_3	Age of home (years) x_4	Price ($1000) y
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
852	2	1	36	178
...	...	...	...	...

图 1: 房屋价格预测模型-多变量

为了后面描述的方便，特别对符号作出声明:

- n 特征的数目
- m 训练集中实例的数量
- $x's$ 输入变量或者特征
- $y's$ 输出变量或目标变量
- $x^{(i)}$ 第 i 个训练样本的输入特征
- $x_j^{(i)}$ 第 i 个训练样本的第 j 个特征

- x_j 样本数据的第 j 个特征

如图1中，第一个训练样本为 $x^{(1)} = \begin{bmatrix} 2104 & 5 & 1 & 45 & 460 \end{bmatrix}$ 则第一个训练样本的第一个特征为 $x_1^{(1)} = 21040$ 样本数据的第一个特征 $x_1 = \begin{bmatrix} 2104 \\ 1416 \\ 1534 \\ 852 \end{bmatrix}$

同样地，支持多变量的假说函数表示为

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

为了更方便的表示，令 define $x_0 = 1$

$$X = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ \cdots \\ x_n \end{bmatrix} \in \mathbb{R}^{n+1}, \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \theta_3 \\ \cdots \\ \theta_n \end{bmatrix}$$

所以假说函数 h_θ 可以表示为

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + ... = \theta^T X$$

0.2 梯度下降算法

本节主要讲解如何设定该假设的参数，特别地，讲解如何使用梯度下降算法处理多元线性回归模型。现在我们已经知道：

假说函数：$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + ... = \theta^T X$

参数：$\theta \in \mathbb{R}^{n+1}$

代价误差函数：$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$

Goals：$minimize_\theta J(\theta)$

梯度下降算法伪代码：

Algorithm 1 梯度下降算法多变量线性回归版本

Require: α 学习率; $\theta \in \mathbb{R}$ 参数; J 代价误差函数; := 赋值;

1: **repeat** $\quad for\ j = 0, ..., n$

2: $\quad \theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$;

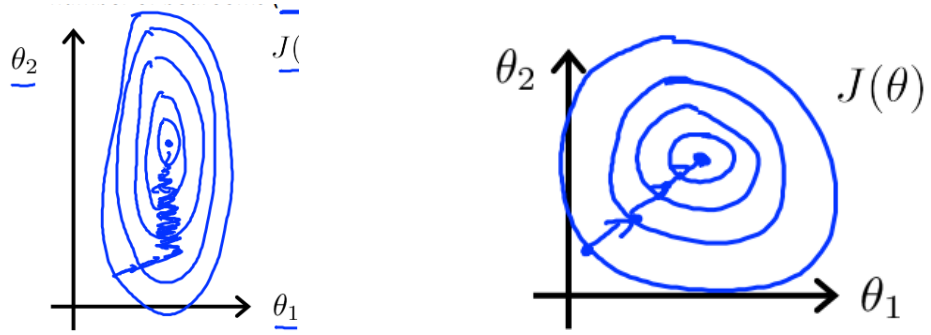
3: **until** 收敛

0.3 运算中的适用技巧

0.3.1 特征缩放-特征归一化

特征缩放（feature scaling）可以把特征缩放到一个相近的范围内，是一种能使得梯度下降算法快速收敛的一个技巧。

例子：图2中，θ_2 表示房屋的大小（0 2000 $feets$），θ_2 表示卧室的数量 (0 5)，从图2(a) 可以看出等高线很扁，如果不进行特征的缩放，梯度下降算法需要经过很多次迭代才能到达最小值点。经过缩放后，收敛到最小值速度比不缩放的快。



(a) 没有经过特征缩放处理，算法的收敛情况　　(b) 经过特征缩放处理后，算法的收敛情况

图 2: 特征缩放对算法收敛的影响

为了实现特征缩放将会使用到一种叫做**均值归一化**的技术，使得 θ_1,θ_2 的范围在 [-1,1] 之间。主要的转换公式为：

$$X_i = \frac{X_i - \mu_n}{\mathbb{S}_n}$$

其中 μ_n 表示 X 的均值，$\mathbb{S}_n$ 表示 X 的范围，即 $max - min$.

比如 θ_1 归一化的结果为 $\theta_1 = \frac{\theta_1 - 1000}{2000}$

0.3.2 学习率 α

复习：梯度下降算法的迭代公式如下.

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial_{\theta_j}} J(\theta)$$

通过上面的公式，我们可以绘制出迭代次数与代价函数 $J(\theta)$ 的关系曲线. 通过曲线观测算发在何时起开始收敛。也有一些自动测试是否收敛的方法，例如将代价函数的变化与某个阈值（例如 0.01）进行比较，通常图3能直观的表示出 $J(\theta)$ 的变化情况。

那么上面的例子与学习率有什么关系？关系就是如果学习率 α 选择不合适将会导致梯度算法不能工作，即不能到达最小值或者将会使得梯度下降算法运行效率低迟迟不能收敛！

总结：

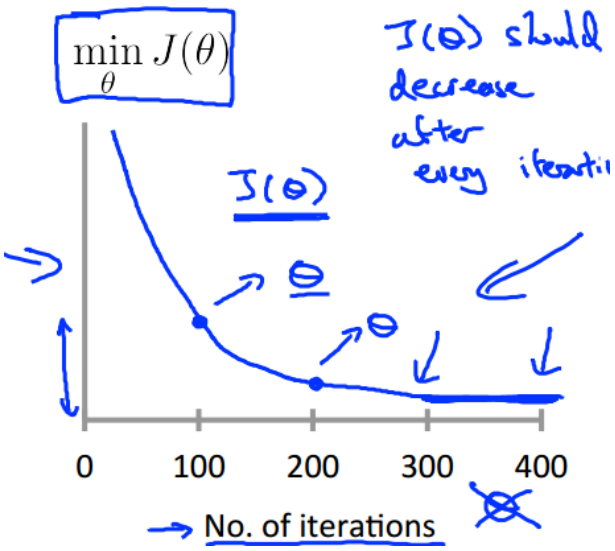


图 3: $J(\theta)$ 与迭代次数的关系

1. 如果 α 过小，将会导致算法的收敛速度慢;
2. 如果 α 过大，在梯度下降过程中，很容易掠过局部最小值，导致无法收敛甚至发散。如图??.

技巧：可以尝试下面的学习率:

$$\alpha = 0.001, 0.003, 0.01, 0.03, 0.1, 0.3, 1$$