斯坦福大学公开 Machine Learning 学习笔记

第一章 监督学习

1.监督学习（Supervised Learning）

监督学习的定义：给出一个算法，需要部分已有正确答案的数据集（训练集），算法的结果就是算出更多的正确答案。一般监督学习分为回归问题（Regression）和分类问题（Classification）。回归问题意指要预测一个连续值的输出，比如预测房价。分类问题指离散（Discrete）的输出值，比如垃圾邮件分类器的实现。

2.无监督学习（Unsupervised Learning）

无监督学习：在没有正确的历史数据（训练集）的情况下解决问题，最终预测数据集中存在的某些关系。例如使用所很多用户的消费历史数据，将这些用户依据他们的消费习惯归类为几个大类，如有些用户喜欢购买电子产品，有些喜欢购买衣服，有些喜欢购买书籍等等。

3.线性回归（Linear Regression）模型

概念声明

特征值: 决定某个模型的所有（大部分）属性的值。如在预测房价中，决定房价高低的属性有：房子的面积，房间的数量等等属性。从例子中可以看出每个模型的特征值不止一个，可能有更多，如果使用 x_i 来表示某个特定的属性，那么所有特征值将组成一个向量：（x_1，x_2，x_3, ...），这一组向量就是将模型通过特征值提取转变为数学模型的关键一步。其实特征值提取也是一门复杂的学问，尤其是对一个复杂的模型，存在成百上千的特征值，如何从中挑选出权重高的特征值，绝对不是一件容易的事情，有可能还涉及到人文学科。

特征值向量：与每个特征值对应的值，决定特定特征值在模型中占的比重。其实也就表示了特征值和结果之间的某种关系。例如，房价预测中，特征值房子的面积 X_i 的比重值 θ_i 将决定 X_i 在最终的房价预测中占多大的比重。同上，一组特征值对应一组特征值向量（θ_1，θ_2，θ_3, ...）。如果知道了特征值向量，我们就可以将特征向量用在新特征值上，从而预测出其对应的结果。

训练集：正确的历史数据的集合（样本集合），其中包括很多组特征值和其对应的实际结果，都是由向量组成。例如，一个样本数据 i，其特征值为（X_i^1，X_i^2，X_i^3，...，X_i^n），其对应的实际结果用 y^i 表示，那么由很多这样的训练样本组成的特征值和对应的结果便是两个矩阵，分别表示为：

$$X = \begin{bmatrix} x_1^1 & x_1^2 & x_1^3 & \cdots & x_1^n \\ x_2^1 & x_2^2 & x_2^3 & \cdots & x_2^n \\ x_3^1 & x_3^2 & x_3^3 & \cdots & x_3^n \\ & & \vdots & & \\ x_m^1 & x_m^1 & x_m^3 & \cdots & x_m^n \end{bmatrix}$$

$$y=[\, y^1, y^2, y^3, \ldots, y^m,]^T$$

其中用 n 表示描述样本的特征值数量，也就是描述决定模型结果的特征数量，用 m 表示整个训练集有 m 个样本，那么 $X \in R^{m*n}$，即属于 m 行 n 列的实数矩阵，$y \in R^m$，y 属于 m 行 1 列的矩阵，其实这里我们假设结果值只有一个，但是有时候结果值也是一组向量，比如有 o 个，那么 $y \in R^{m*o}$，视具体模型而定。

假设函数（hypothesis function）：假设函数其实就是我们通过训练集训练得出的用来预测未知数据结果的函数，用 $h_\theta(x)$表示，其实就是由特征向量和特征值组成的关系函数。

以上各个概念在算法中的作用可以用图 1-1 表示出来。首先通过训练集，使用训练算法训练（后者说是让算法自己学习）出特征值和结果之间的关系，即假设函数，然后在使用假设函数预测结果值位置的样本，得出预测的结果。

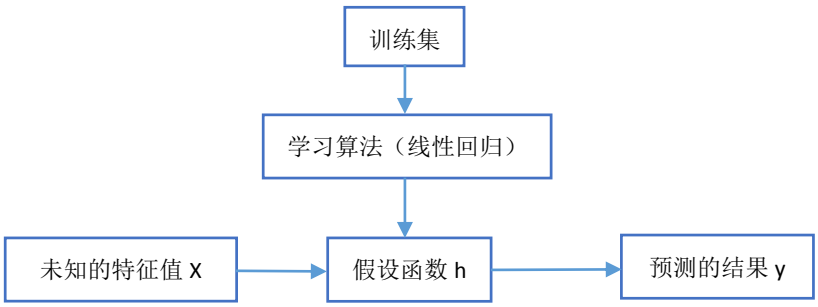

图 1-1 学习算法的简单模型

线性回归算法

线性回归模型应该是监督学习中最简单的算法了，其假设结果 y^i 与特征值 x^i 是线性关系。由此可以将假设函数定义为：

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

其中，θ_0 是 x_0 的权重值（系数），但是 $x_0=1$，所以一般 $\theta \in R^{n+1}$。线性回归的问题就是已知了一部分 x 和对应的 $h_\theta(x)$值，求出（训练出）线性表达式的系数（特征值向量）。那么，如何通过训练集训练出我们想要的线性表达式的系数呢？最简单的方法就是试错，但这里使用的并不是纯粹碰运气的试错，而是使用统计数学工具一步步优化的试错：先给出一个随机的特征值向量初始值 $\Theta' = (\theta_0, \theta_1, \theta_2, \theta_3, \ldots)$，然后将 Θ' 带入假设函数中得到假设函数为 $h_{\Theta'}(x)$，使用这个假设函数来预测已经有正确结果的训练集中的特征值，最后得出预测结果 y'，然后使用 y'和真实的结果 y 比较，得出一个代价值（误差），也就是预测值和真实值之间的差距，然后调整一点点 Θ，再计算对应的预测值，再计算代价值（误差），再调整……, 通过循环上面的方法有限次，最终会得出一个比较好的 Θ，使得假设函数的代价值已经达到最小，也就是再调整 Θ，也不会有更好的预测结果了，由此我们就可以认为该 Θ 对应的假设函数已经完美拟合了我们的训练集，如果我们的训练集有一定的代

表性的话，那么它也就可以用于预测未知特征值对应的结果。

以上只是文字的描述，从中我们可以看出有两个难点需要处理，其一是如何求所谓的代价值，单纯地使用 $y'-y$？，这好像是可以的，但是 $y'-y$ 对应的只是一个样本的代价值，要如何将所有的样本的代价值都涵盖进来呢？简单的办法是使用加和平均值（即代价值的期望值，或均值），也就是将所有样本的代价值加起来除以样本总数，貌似也是可以的。第二个难点是调整 Θ，该如何调整，总不能又随机地调整，这样就是纯粹的靠运气了。直觉告诉我们调整一定要和这个代价值有关，从而使得每次调整后代价值都会减小一些。其实，数学家已经为我们解决了这个问题，使用差平方和来表示代价值（为什么使用差平方和，而不是样本误差的加和平均值？这个问题还是要好好看看这两个的区别，属于数学范围的讨论了），那么就可以使用梯度下降算法来调整 Θ，而且还和代价值有关，其中梯度下降算法下节会讨论。下面我们先以最简单的假设函数（只有一个特征值 x_1, 所以假设函数 $h_\theta(x) = \theta_0 + \theta_1 x_1$）为例，给出其代价函数：

$$\text{cost function} = J(\theta_0, \theta_1) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x_i) - y_i)^2 \qquad \text{（公式 1-1）}$$

从我们之前对算法的描述可以得出，其实我们一直调整 Θ，就是为了使得公式 1-1 的值最小，即 $h_\theta(x)$函数的代价最小，使用数学描述就是求使公式 1-1 的值最小时的 Θ 值，即如下公式：

$$\min_{\theta_0, \theta_1}(J(\theta_0, \theta_1)) \qquad \text{（公式 1-2）}$$

其中 $J(\theta_0, \theta_1)$又被称作平方误差函数，平方误差代价函数（Squared Error Function, Mean Squared Error）。

梯度下降算法（Gradient Descent）

从上一节的分析可以知道，我们的目标就是求得使公式 1-1 的值最小时的 Θ 值，其中有个方法就是梯度下降算法。我们直接给出调整 Θ 的公式：

$$\theta_j := \theta_j - \alpha\frac{\partial}{\partial\theta_j}J(\theta_0, \theta_1) \quad \text{（for j=0 and j=1）} \qquad \text{(公式 1-3)}$$

其中，:=为赋值运算符，α是学习速率，在梯度下降算法中，它控制了下降的步伐，我们后面还会经常使用它，$\frac{\partial}{\partial\theta_j}J(\theta_0, \theta_1)$是$J(\theta_0, \theta_1)$对$\theta_j$的偏导，$\theta_j \in (\theta_0, \theta_1)$。关于过多的数学知识，我就不班门弄斧了，而且也不是我的强项，这里只给出我个人的理解。对于偏导数$\frac{\partial}{\partial\theta_j}J(\theta_0, \theta_1)$我们都知道是代价函数$J(\theta_0, \theta_1)$相对$\theta_j$的斜率（也就是$J(\theta_0, \theta_1)$在$\theta_j$上的方向）。而梯度下降算法就是将$\theta_j$朝其在$J(\theta_0, \theta_1)$上的方向上改变一点点，也就是说$\theta_j$的改动是依赖于其在$J(\theta_0, \theta_1)$上的变化方向的，这样的改动就满足了我们之前的要求，而且是沿着合理的方向改变，最终会找到一个满足最小值的 θ_0, θ_1，因为最终会找到一组特征值向量，使得$\frac{\partial}{\partial\theta_j}J(\theta_0, \theta_1) = 0$，然后再调整$\theta_j$就没有意义，其实从微积分角度

理解就是此时的我们已经求得了$J(\theta_0, \theta_1)$的最小值，也就是说θ_0, θ_1已经达到了最优值。注意，在梯度下降中，要同时更新θ_0, θ_1的值，即在同一个$J(\theta_0, \theta_1)$的基础上更新θ_0和θ_1，因为在每次θ_0和θ_1的更新后，$J(\theta_0, \theta_1)$也会随之更新。

最后，应该注意的就是学习速率α不应该过大也不应该过小，过小会使得下降的步伐太小而迭代太多次，太大则会导致发散，最终无法拟合训练集样本，图 1-2 给出了不同的情况的表现，其中 $h_\theta(x) = \theta_0 + \theta_1 x_1$，左图使用了太多次才到达最优值，右图则越来越离散，离最优值越来越远。

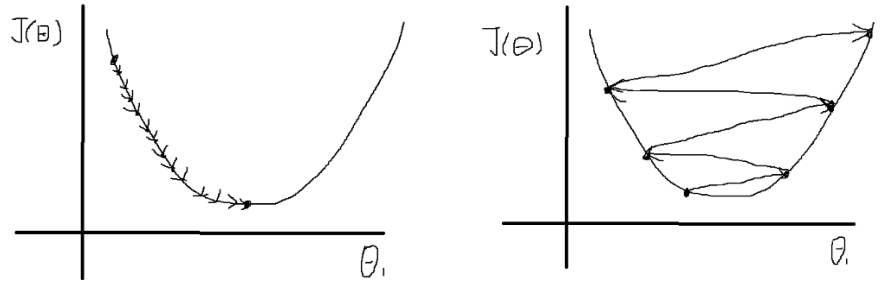

图 1-2　使用太小和太大α，算法的表现

还应该注意一点，先给出课程上 ppt 的英文原话：
Gradient descent can converge to a local minimus, Even with the learning rate α fixed. As we approach a local minimum, gradient descent will automatically take smaller steps. So, no need to decrease α over time。
意思即是说，梯度下降有可能最终拟合在一个局部最优解上，而不是全局最优解，即使我们使用了固定的学习速率α。所以当我们的算法朝向一个局部最优解的时候，梯度下降会自动使用小步伐进行下降，而不用随着时间去减小α的值。意思即是说导致局部最优解的不是学习速率的问题，我们不应该在它身上花费时间，其实导致局部最优解的原因是我们一开始的随机初始化的特征值向量选取的不是很好，所以有时候我们应该多选用几个不同的特征值向量来训练算法。

在线性回归算法中使用梯度下降

本章我们学习的线性回归模型都是只有一个特征值的模型，即假设函数 $h_\theta(x) = \theta_0 + \theta_1 x_1$，其实搞懂了一个特征值的模型，理解多特征值模型也绝不是问题了，而且视频中先使用单特征值作为初学，也是因为其易理解和比较直观。整个单特征值线性回归算法的训练描述如下：
1. 获取训练集
2. 给出假设函数：

$$h_\theta(x) = \theta_0 + \theta_1 x_1 \qquad \text{(公式 1-3)}$$

3. 给出假设函数的代价函数：

$$J(\theta_0, \theta_1) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x_i) - y_i)^2 \qquad \text{(公式 1-4)}$$

4. 求得 J(θ0,θ1)对θ_j（θ0,θ1）的偏导：

$$\frac{\partial}{\partial\theta_j}J(\theta_0,\theta_1)=\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x_i)-y_i)\cdot x_j$$

$$=\frac{1}{m}\sum_{i=1}^{m}(\theta_0+\theta_1x_1^i-y_i)\cdot x_j^i \qquad \text{(公式 1-5)}$$

上式也等于下面的公式，θ_0和θ_1不同的原因是对应的x_0^i都是常数 1，故忽略掉了：

$$\frac{\partial}{\partial\theta_0}J(\theta_0,\theta_1)=\frac{1}{m}\sum_{i=1}^{m}(\theta_0+\theta_1x_1^i-y_i)$$

$$\frac{\partial}{\partial\theta_1}J(\theta_0,\theta_1)=\frac{1}{m}\sum_{i=1}^{m}(\theta_0+\theta_1x_1^i-y_i)\cdot x_1^i$$

4. 循环执行下面的更新，直到代价函数达到最优解，即算法拟合为止。必须同时更新 θ0，θ1的值：

$$\theta_j:=\theta_j-\alpha\frac{\partial}{\partial\theta_j}J(\theta_0,\theta_1) \qquad \text{（for j=0 and j=1）} \qquad \text{(公式 1-6)}$$

上式等价于两步：

$$\theta_0:=\theta_0-\alpha\frac{\partial}{\partial\theta_0}J(\theta_0,\theta_1)=\theta_0-\frac{\alpha}{m}\sum_{i=1}^{m}(\theta_0+\theta_1x_1^i-y_i)$$

$$\theta_1:=\theta_1-\alpha\frac{\partial}{\partial\theta_1}J(\theta_0,\theta_1)=\theta_1-\frac{\alpha}{m}\sum_{i=1}^{m}(\theta_0+\theta_1x_1^i-y_i)\cdot x_1^i$$

最后值得强调的是：对于线性回归的代价函数，使用梯度下降算法求最优解，没有局部最优解，只有全局最优解。

个人理解可能存在偏差或错误，欢迎交流和批评指正： wearyoung@outlook.com