机器学习笔记

Le Yang
yangle0125@qq.com

目录

[1]通常叫最小二乘，但是我认为最小二乘这个表述不如最小平方直白。

第一部分 监督学习

记号说明：我们将使用 $x^{(i)}$ 表示“输入”变量, 也叫做特征; 使用 $y^{(i)}$ 表示“输出”或者说目标变量。$(x^{(i)}, y^{(i)})$ 称为一个训练样本, 而用来训练的数据集,$\{(x^{(i)}, y^{(i)}); i = 1, \ldots, m\}$ 称为训练集合。注意这里的 (i) 仅仅作为索引使用，m 是训练集合中训练样本的数量。我们也将使用 $\mathcal{X}$ 表示输入变量所在的空间，用 $\mathcal{Y}$ 表示输出变量所在的空间。

为了更正式一点地描述监督学习的问题，我们的目标是，给定一个训练集合，尝试学习一个函数 $h : \mathcal{X} \mapsto \mathcal{Y}$, 使得 $h(x)$ 相对于对应的 y 来说是一个“好”的预测。因为某些历史上的原因，这个函数 h 被称为一个 *hypothesis*（假设）。

当要预测的目标变量是连续的时，称这个学习问题是一个回归问题; 当要预测的目标只取少量的离散值时，称这个学习问题是一个分类问题。

1 线性回归

作为初始尝试，让我们假定 y 是 x 的线性函数：

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

这里，这些 θ_i 被称为参数 (或称权重)。在不会引起误解的情况下，我们可以扔掉 $h_\theta(x)$ 的下标 θ，将其简单地写作 $h(x)$。为了简化记号，我们再引入一个约定，令 $x_0 = 1$(即截距项)，所以：

$$h(x) = \sum_{i=0}^{n} \theta_i x_i = \theta^T x \tag{1}$$

在最右侧，请将 θ 和 x 都看作向量; 这里 n 是输入变量的个数 (不包括 x_0)。现在，给定一个训练集合，我们应该如何选择，或者说学习，参数 θ 呢？一个合理的方法是，选择这样的 θ，使得 $h(x)$ 尽量接近 y，至少在训练集合内应该这样。为了说地更正式一些，我们将定义一个函数用来衡量对于每一个 θ 的值 $h(x^{(i)})$ 和对应的 $y^{(i)}$ 到底有多接近。我们定义损失函数:

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

1.1 LMS(最小均方) 算法

我们要选择 θ 来最小化 $J(\theta)$。为了这样做，让我们使用一种搜索算法，其始于对 θ 的某些“初始猜测”，然后重复地改变 θ 来让 $J(\theta)$ 越来越小。对于这个问题，让我们考虑梯度下降算法。它基于某些初始的 θ 值，重复地执行如下的更新 (这个更新是对所有的 $j = 0, \ldots, n$ 同时进行的)：

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

这里 α 被称为学习速率。这是一个非常自然的算法，重复地向着 $J(\theta)$ 减少最大的方向更新 θ。为了实现这个算法，我们需要求出最右边一项的偏导数。让我们先假设训练集合中只有一个训练样本 (x, y)，这样我们就不用考虑 J 的定义中的求和符号了，我们有：

$$\begin{aligned}\frac{\partial}{\partial\theta_j}J(\theta) &= \frac{\partial}{\partial\theta_j}\frac{1}{2}(h_\theta(x)-y)^2 \\ &= 2\cdot\frac{1}{2}(h_\theta(x)-y)\cdot\frac{\partial}{\partial\theta_j}(h_\theta(x)-y) \\ &= (h_\theta(x)-y)\cdot\frac{\partial}{\partial\theta_j}(\sum_{i=0}^{n}\theta_i x_i - y) \\ &= (h_\theta(x)-y)x_j\end{aligned} \tag{3}$$

对于单个训练样本，这就给出了更新规则：[2]

$$\theta_j := \theta_j + \alpha(y^{(i)} - h_\theta(x^{(i)}))x_j^{(i)}$$

这个规则被称为 *LMS* 更新规则 (LMS 代表 "least mean squares"), 在某些地方，也被称为 *Widrow-Hoff* 学习规则。这个规则有一些特点使其看起来很自然，例如，更新的幅度是正比于误差项 $(y^{(i)} - h_\theta(x^{(i)}))$ 的。因此，如果我们遇到一个样本，我们的预测和实际的值很接近，那么按照这个规则，θ 就几乎不需要做什么更新；反之，如果我们的预测和实际值相差很远，参数就会有较大幅度的更新。

我们在只有一个样本的情况下推导出了 LMS 规则。有两种方法来修改这一规则使其适用于多于一个样本的训练集合，第一种方法是使用下面的算法：

Algorithm 1 批量梯度下降法 (BGD)

repeat

$\theta_j := \theta_j + \alpha\sum_{i=1}^{m}(y^{(i)} - h_\theta(x^{(i)}))x_j^{(i)}$ (for every j)

until Convergence

该算法在做每一步更新时，都需要查看所有的数据，因此被称为 *batch gradient descent* (批量梯度下降)。还有另外一种算法工作地也很好:

Algorithm 2 随机梯度下降法 (SGD)

repeat

$\theta_j := \theta_j + \alpha(y^{(i)} - h_\theta(x^{(i)}))x_j^{(i)}$ (for every j)

$i := (i+1) \mod m$

until Convergence

[2]我们使用记号 "$a := b$" 表示一个 (计算机程序中的) 操作，含义是我们将设置一个变量 a 的值，使其等于 b 的值。换句话说，这个操作将使用 b 的值来覆盖 a。相反的，我们使用记号"$a = b$" 来断言一个事实，即 a 的值和 b 的值是相等的。

该算法在做每一步更新时，只需查看训练集合中的下一个样本即可。被称为 *stochastic gradient descent*（随机梯度下降）或者是 *incremental gradient descent*（增量梯度下降）。

1.2 正则方程

梯度下降是一种最小化 $J(\theta)$ 的方法，现在让我们来讨论另外一种。这次我们将直接求解 $J(\theta)$ 的最小值，而不需要求助于迭代算法。在这种方法中，我们将求得 $J(\theta)$ 相对于 θ_j 的导数，令其得零。首先让我们来引入一些矩阵求导的记号。

1.2.1 矩阵求导

对于一个将 $m \times n$ 的矩阵映射到实数的函数 $f : \mathbb{R}^{m\times n} \mapsto \mathbb{R}$，我们定义函数 f 对矩阵 A 的导数为:

$$\nabla_A f(A) = \begin{bmatrix} \frac{\partial f}{\partial A_{11}} & \cdots & \frac{\partial f}{\partial A_{1n}} \\ \vdots & \ddots & \vdots \\ \frac{\partial f}{\partial A_{m1}} & \cdots & \frac{\partial f}{\partial A_{mn}} \end{bmatrix} \tag{4}$$

因此，$\nabla_A f(A)$ 本身也是个 $m \times n$ 的矩阵，它的第 (i, j) 元素是 $\partial f/\partial A_{ij}$。我们还要引入“迹”的概念，写做 “tr”。对一个 $n \times n$ 的 (方) 矩阵 A，A 的迹被定义为其对角线上元素的和：

$$\operatorname{tr} A = \sum_{i=1}^{n} A_{ii} \tag{5}$$

如果 a 是一个实数 (也就是，一个 1×1 的矩阵)，那么 $\operatorname{tr} a = a$。迹的定义导致其有这样的属性，即对于两个矩阵 A 和 B，如果 AB 是方阵，那么 $\operatorname{tr} AB = \operatorname{tr} BA$。因此，我们还将有，比如说：

$$\operatorname{tr} ABC = \operatorname{tr} CAB = \operatorname{tr} BCA$$
$$\operatorname{tr} ABCD = \operatorname{tr} DABC = \operatorname{tr} CDAB = \operatorname{tr} BCDA$$

迹的如下属性也很容易验证，这里 A 和 B 都是方阵，a 是一个实数

$$\operatorname{tr} A = \operatorname{tr} A^T \tag{6}$$
$$\operatorname{tr}(A + B) = \operatorname{tr} A + \operatorname{tr} B \tag{7}$$
$$\operatorname{tr} aA = a \operatorname{tr} A \tag{8}$$

我们不加证明地[3]给出矩阵求导的如下属性，(12)只适用于非奇异的方阵 A，其中 $|A|$ 代表 A 的行列式。我们有：

$$\nabla_A \operatorname{tr} AB = B^T \tag{9}$$

$$\nabla_{A^T} f(A) = (\nabla_A f(A))^T \tag{10}$$

$$\nabla_A \operatorname{tr} ABA^TC = CAB + C^TAB^T \tag{11}$$

$$\nabla_A |A| = |A|(A^{-1})^T \tag{12}$$

1.2.2 最小平方[4]

让我们继续求解 $J(\theta)$ 的最小值，首先我们需要将 J 写成矩阵-向量记号的形式。给定一个训练集合，定义设计矩阵 X 为一个 $m \times n$ 的矩阵 (实际上是 $m \times (n+1)$，如果包括截距项的话)。该矩阵的每一行都是一个输入变量：

$$X = \begin{bmatrix} —(x^{(1)})^T— \\ —(x^{(2)})^T— \\ \vdots \\ —(x^{(m)})^T— \end{bmatrix}$$

再定义 $\vec{y}$ 为一个 m 维的向量，包含训练集合中的所有目标变量：

$$\vec{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{bmatrix}$$

---

[3]记笔记的人表示偏要证明一下。

(9)$(\partial/\partial A_{ij}) \operatorname{tr} AB = (\partial/\partial A_{ij}) \sum_i \sum_j A_{ij} B_{ji} = B_{ji}$

(10)按照定义，$(\nabla_{A^T} f(A))_{ij} = (\partial/\partial A_{ij}^T) f(A) = (\partial/\partial A_{ji}) f(A)$

(11)$(\partial/\partial A) \operatorname{tr} ABA^TC$

$= (\partial/\partial A) \sum_{i,j,k,l} A_{ij} B_{jk} A_{lk} C_{li}$

$= (\partial/\partial A_{ij}) \sum_{i,j,k,l} A_{ij} B_{jk} A_{lk} C_{li} + (\partial/\partial A_{lk}) \sum_{i,j,k,l} A_{ij} B_{jk} A_{lk} C_{li}$

$= B_{jk} A_{lk} C_{li} + A_{ij} B_{jk} C_{li} = (BA^TC)_{ji} + (CAB)_{lk} = (C^TAB^T)_{ij} + (CAB)_{lk}$。

(12)定义 A 关于第 i 列第 j 行的余子式（记作 M_{ij}）是去掉 A 的第 i 行第 j 列之后得到的 $(n-1) \times (n-1)$ 矩阵的行列式。A 关于第 i 列第 j 行的代数余子式是：$A'_{ij} = (-1)^{i+j} M_{ij}$。$A$ 的余子矩阵是一个 $n \times n$ 的矩阵 A'，使得其第 i 行第 j 列的元素是 A 关于第 i 行第 j 列的代数余子式。矩阵 A 的伴随矩阵是 A 的余子矩阵的转置矩阵 A'^T。作为拉普拉斯公式的推论，关于 $n \times n$ 矩阵 A 的行列式, 有 $AA'^T = |A|I$。这表明 $A' = |A|(A^{-1})^T$。注意到 $|A| = \sum_j A_{ij} A'_{ij}$。因为 A'_{ij} 与 A_{ij} 无关 (这可以从定义看出来)，这就意味着 $(\partial/\partial A_{ij})|A| = A'_{ij}$，而这就是所要证的。

[4]通常叫最小二乘，但是我认为最小二乘这个表述不如最小平方直白。

现在，因为 $h_\theta(x^{(i)}) = (x^{(i)})^T\theta$，很容易验证：

$$\begin{aligned} X\theta - \vec{y} &= \begin{bmatrix} (x^{(1)})^T\theta \\ \vdots \\ (x^{(m)})^T\theta \end{bmatrix} - \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(m)} \end{bmatrix} \\ &= \begin{bmatrix} h_\theta(x^{(1)}) - y^{(1)} \\ \vdots \\ h_\theta(x^{(m)}) - y^{(m)} \end{bmatrix} \end{aligned}$$

注意到对于向量 z，我们有 $z^Tz = \sum_i z_i^2$，利用这一事实，就得到：

$$\begin{aligned} \frac{1}{2}(X\theta - \vec{y})^T(X\theta - \vec{y}) &= \frac{1}{2}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 \\ &= J(\theta) \end{aligned}$$

最后，为求得 J 的最小值，让我们对 J 求导：

$$\begin{aligned} \nabla_\theta J(\theta) &= \nabla_\theta \frac{1}{2}(X\theta - \vec{y})^T(X\theta - \vec{y}) \\ &= \frac{1}{2}\nabla_\theta(\theta^TX^TX\theta - \theta^TX^T\vec{y} - \vec{y}^TX\theta + \vec{y}^T\vec{y}) \\ &= \frac{1}{2}\nabla_\theta \operatorname{tr}(\theta^TX^TX\theta - \theta^TX^T\vec{y} - \vec{y}^TX\theta + \vec{y}^T\vec{y}) \\ &= \frac{1}{2}\nabla_\theta(\operatorname{tr}\theta^TX^TX\theta - 2\operatorname{tr}\vec{y}^TX\theta) \\ &= \frac{1}{2}(X^TX\theta + X^TX\theta - 2X^T\vec{y}) \\ &= X^TX\theta - X^T\vec{y} \end{aligned}$$

令该导数为零，我们就得到了*正则方程*：

$$X^TX\theta = X^T\vec{y} \tag{13}$$

因此，使得 $J(\theta)$ 最小的 θ 就是：

$$\theta = (X^TX)^{-1}X^T\vec{y} \tag{14}$$

1.3 概率解释

当我们面对一个回归问题时，为什么线性回归，特别是，为什么最小平方损失函数 J，会是一个较为合理的选择呢？在这一小节里，我们将给出一些概率假设，在这些假设下，最小平方回归可以被很自然地推导出来。

让我们假定目标变量和输入变量之间，由下式联系：

$$y^{(i)} = \theta^T x^{(i)} + \epsilon^{(i)} \tag{15}$$

这里的 $\epsilon^{(i)}$ 是误差项，用来捕获那些未被纳入到模型中的影响，或者是训练集合中的随机噪声。进一步地，让我们假设 $\epsilon^{(i)}$ 之间是互相独立的，且都服从均值为零，方差为 σ^2 的高斯分布 (也叫正态分布)。换句话说，$\epsilon^{(i)} \sim \mathcal{N}(0, \sigma^2)$ 是独立同分布 (IID, independently and identically distributed) 的。这就是说，$\epsilon^{(i)}$ 的概率密度由下式给出：

$$p(\epsilon^{(i)}) = \frac{1}{\sqrt{2\pi}\sigma} exp\left(-\frac{(\epsilon^{(i)})^2}{2\sigma^2}\right)$$

因此

$$p(y^{(i)}|x^{(i)};\theta) = \frac{1}{\sqrt{2\pi}\sigma} exp\left(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}\right) \tag{16}$$

$p(y^{(i)}|x^{(i)};\theta)$ 表示以 θ 为参数，在给定 $x^{(i)}$ 时，$y^{(i)}$ 的概率密度。注意我们不能以 θ 为概率条件 ("$p(y^{(i)}|x^{(i)},\theta)$")，因为 θ 不是随机变量。我们也可以将 $y^{(i)}$ 的概率分布写作 $y^{(i)}|x^{(i)};\theta \sim \mathcal{N}(\theta^T x^{(i)}, \sigma^2)$。

给定 X(设计矩阵，它包含所有的 $x^{(i)}$) 和 θ，所有的 $y^{(i)}$ 服从什么分布呢？整体数据的概率应该由 $p(\vec{y}|X;\theta)$ 给出，对于一个固定的 θ 来说，该量通常被视为 $\vec{y}$(也许还应该包括 X) 的一个函数。当我们希望把它看作是关于 θ 的函数时，我们称之为似然函数：

$$L(\theta) = L(\theta; X, \vec{y}) = p(\vec{y}|X;\theta)$$

注意到 $\epsilon^{(i)}$ 之间是相互独立的 (因此，给定 $x^{(i)}$ 时 $y^{(i)}$ 也是相互独立的)，上式还可以被写作：

$$\begin{aligned} L(\theta) &= \prod_{i=1}^{m} p(y^{(i)}|x^{(i)};\theta) \\ &= \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi}\sigma} exp\left(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}\right) \end{aligned} \tag{17}$$

现在，给定了关于 $y^{(i)}$ 和 $x^{(i)}$ 的概率模型，什么是选择我们对参数 θ 的最好猜测的合理方式呢？极大似然法则说，我们应该选择这样的 θ，其使得整体数据的概率越大越好。也就是说，我们需要选择 θ 去最大化 $L(\theta)$。

除了最大化 $L(\theta)$，我们也可以最大化任何一个关于 $L(\theta)$ 的单调递增函数。特别是，如

果我们选择最大化对数似然函数 $l(\theta)$ 的话，推导会更为简单一些：

$$\begin{aligned} l(\theta) &= \log L(\theta) \\ &= \log \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi}\sigma} exp\left(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}\right) \\ &= \sum_{i=1}^{m} \log \frac{1}{\sqrt{2\pi}\sigma} exp\left(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}\right) \\ &= m \log \frac{1}{\sqrt{2\pi}\sigma} - \frac{1}{\sigma^2}\frac{1}{2}\sum_{i=1}^{m}(y^{(i)} - \theta^T x^{(i)})^2 \end{aligned} \tag{18}$$

因此，最大化 $l(\theta)$ 也就是最小化 $\frac{1}{2}\sum_{i=1}^{m}(y^{(i)} - \theta^T x^{(i)})^2$，而这就是 $J(\theta)$，我们原始的最小平方损失函数。

总结一下：在对于数据集的前述概率假设之下，最小平方回归对应于寻找 θ 的最大似然估计。因此在这种假设之下，最小平方回归可以由寻找最大似然估计这种非常自然的方式导出。(然而，需要注意的是这些概率假设对于选用最小平方作为合理的损失函数来说并不是必须的，实际上有很多其他的自然的假设也可以导出最小平方回归。)

另外也请注意，在我们前面的讨论中，我们对于 θ 的最终选择并不依赖于 σ^2，实际上如果不知道 σ^2 是多少，我们仍然会得到关于 θ 的相同的结果。稍后讨论到指数分布族和广义线性模型时，我们会用到这一事实。

1.4 局部权重线性回归

有的时候，数据并不是那样直直地躺在一条线上。在这种情况下，如果使用线性回归去拟合数据集时，结果吻合得并不会特别好。此时，如果我们加入一个额外的特征 x^2，使用 $y = \theta_0 + \theta_1 x + \theta_2 x^2$，那么我们就能够得到一个稍好一些的对于数据的拟合。然而，加入太多的特征也有坏处：虽然对于现有的数据可以拟合的很好，但是对于新数据，也许就并不是一个好的预测了。我们称这两种情况 (使用的特征过少和使用的特征过多) 分别为欠拟合和过拟合。

在前面的讨论中可以看到，特征的选择对保证算法的表现来说是很重要的。在这一节中，我们会简要地讨论局部加权线性回归 (locally weighted linear regression, LWR)。这种算法假定有足够多的数据，因此会使得特征选择稍显次要一些。

在原始的线性回归算法中，要对一个查询点 x 做一个预测，我们需要：

1. 拟合 θ 使其能最小化 $\sum_i (y^{(i)} - \theta^T x^{(i)})^2$
2. 输出 $\theta^T x$

与之对应地，在局部加权线性回归算法中，我们应该这样做：

1. 拟合 θ 使其能最小化 $\sum_i w^{(i)} (y^{(i)} - \theta^T x^{(i)})^2$
2. 输出 $\theta^T x$

这里，$w^{(i)}$ 是一个非负值的权重。就直观感觉而言，如果对于某个 i 来说 $w^{(i)}$ 较大，那么在选择 θ 时，我们就需要更努力地使 $(y^{(i)}-\theta^T x^{(i)})^2$ 更小。如果 $w^{(i)}$ 较小，那么 $(y^{(i)}-\theta^T x^{(i)})^2$ 这个误差项就更倾向于在拟合过程中被忽略。

对于权重的一个较为标准的选择是[5]

$$w^{(i)} = exp\left(-\frac{(x^{(i)}-x)^2}{2\tau^2}\right) \tag{19}$$

应注意到这个权重依赖于我们对于查询点 x 的选择。更进一步说，如果 $|x^{(i)}-x|$ 较小，那么 $w^{(i)}$ 就较接近 1；如果 $|x^{(i)}-x|$ 较大，那么 $w^{(i)}$ 就较小。因此，拟合 θ 时，查询点 x 附近的训练样本 (的误差) 会被给予一个高得多的权重。(也应注意到虽然权重和高斯分布的概率密度形式上非常接近，$w^{(i)}$ 却并不与高斯分布有什么直接的联系，特别是 $w^{(i)}$ 并不是服从正则分布或是其他什么分布的随机变量。) 参数 τ 控制着一个训练样本的权重随其 $x^{(i)}$ 距查询点 x 的距离增加下降的快慢，被称为*带宽*参数。

局部加权线性回归是我们见到的第一个*非参数化*算法的例子。我们之前见到的 (无加权的) 线性回归算法被认为是一个*参数化*的算法，因为它有固定的，有限个参数 (所有的 θ_i) 要与数据进行拟合。一旦我们拟合好了 θ_i 并将其储存起来，对于下一步的预测来说，我们就不需要保留训练数据了。相反地，要使用局部加权线性回归做预测，我们需要保留整个训练集合。术语“非参数化”(大致上) 是指这样一个事实：为了表达我们的假设 h，我们需要保留的东西的数量与训练集合的大小成正比。

2 分类和 logistic 回归

现在让我们来讨论分类问题。这与回归问题非常相似，除了我们现在想要预测的 y 值只取很少的几个离散值。现在，我们将专注于 *binary classification*（二分类）问题，也就是 y 只取两个值，0 和 1。(我们这里所讨论的大部分内容也可以推广到多分类的情形。)

[5]如果 x 取向量值，该式就应该推广为 $w^{(i)} = exp(-(x^{(i)}-x)^T(x^{(i)}-x)/(2\tau^2))$ 或是 $w^{(i)} = exp(-(x^{(i)}-x)^T\Sigma^{-1}(x^{(i)}-x)/2)$。

3 广义线性模型