Supervised learning

笔记：Alvin

2017-04-17

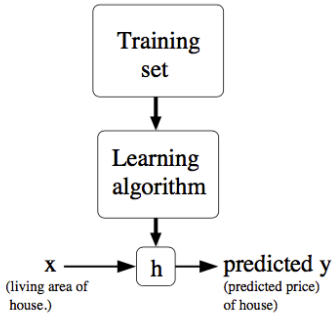
目录

1 引例

监督学习是怎样的一类问题呢，在 Andrew Ng 的讲义中就举了一个例子：假设我们给出一个数据集，该数据集包含了来自 Portland, Oregon 的 47 条住房居住面积和价钱的值。那么如果给定这样的数据，我们如何学习去预测这个城市的其它房子的价钱呢？也就是说如何去预测一个变量为住房面积的价格函数。那么这种类型的问题就是监督学习问题。

在中文的维基百科上是这么定义监督学习的：监督式学习（英语：Supervised learning），是一个机器学习中的方法，可以由训练资料中学到或建立一个模式（函数 / learning model），并依此模式推测新的实例。训练资料是由输入物件（通常是向量）和预期输出所组成。函数的输出可以是一个连续的值（称为回归分析），或是预测一个分类标签（称作分类）。

为了在之后的内容中能够方便表示一些概念，我们用 $x^{(i)}$ 来表示“输入变量”（在上面的例子中就是居住的面积），它们也叫作输入特征，$y^{(i)}$ 来表示“输出变量”或者叫目标变量，它们是我们要预测的（在上面的例子中也就是价钱）。一对数据 $(x^{(i)}, y^{(i)})$ 叫做训练样本，它们是用来学习的，因此 m 个训练数据组成的集合 $(x^{(i)}, y^{(i)}); i = 1, ..., m$ 叫做一个训练集。



我们用 $\mathcal{X}$ 代表输入值的空间，用 $\mathcal{Y}$ 代表输出值的空间，在上面的例子中，$\mathcal{X} = \mathcal{Y} = \mathbb{R}$ 。那么更加形式化地表示监督学习的目的就是给定一个训练集，用它来学习一个函数 $h : \mathcal{X} \mapsto \mathcal{Y}$, 以使得 $h(x)$ 可以称得上是预测 y 值的一个好的预测工具。一般叫这个函数 h 为假设函数 (hypothesis)。

那么当我们想要预测的值是一种连续的值的时候，例如上面例子中的价钱，我们就把这类学习问题叫做回归 (regression) 问题。而当 y 取值是一些离散的值时（例如我们给定房子的居住面积，预测它是普通住房还是公寓），

我们称这类学习问题为分类 (classification) 问题。

2 线性回归

那为了让这样一个例子更有意思，我们为其加上一个特征参数 bedrooms，也就是卧室的间数：

Living area	# bedrooms	Price(1000$ s)
2104	3	400
1600	3	330
2400	3	369
1416	2	232
3000	4	540
.	.	.
.	.	.

因此，这里的 x 是在空间 $\mathbb{R}_2$ 上的二维向量。例如 $x_1^{(i)}$ 就是在训练集中第 i 个房子的居住面积，而 $x_2^{(i)}$ 则是卧室的数量。
那么为了执行监督学习，我们必须确定好怎样在计算机中表示一个函数/假设 h。一般第一个选择的就是用一个关于 x 的线性方程来估计 y 的值：

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

那么这里的 θ_i(也被称作是权值) 就是 $\mathcal{X} \mapsto \mathcal{Y}$ 空间的线性函数的参数，为防止混淆，我们用下表的形式表示，方便起见，这里就按照惯例引入截矩项 $x_0 = 1$（intercept term）：

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

等式最右边的 θ 和 x 都是向量，而 n 就是输入变量的个数（这里不把 x_0 算进去）。

那么现在给定一个训练集，我们该如何挑选或者说，我们该如何学习参数 θ 呢? 其中一个合理的方法应该就是使得函数 $h(x)$ 接近 y, 至少对于我们目前有的训练集而言是这样的。那么形式话的表示就是我们定义一个 cost function：

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

2.1 LMS algorithm

我们现在要决定 θ 来使得 $J(\theta)$ 最小。那么怎么去做呢？我们可以利用一个搜索算法，并且我们可以假设一个初始的"θ" 值，然后不断的重复地改变这个 θ 值从而使得 $J(\theta)$ 的值变小，直到最后得到了一个我们希望的能够使得 $J(\theta)$ 的值最小的那个 θ。那么具体用哪个算法呢？这里就要提到一个叫做梯度下降的算法，该算法拥有一个初始的 θ 值，然后根据函数来不断地更新这个值：

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

这里的 α 叫做 learning rate。这个算法非常自然地在 J 的梯度方向上重复地走一小步。

为了能够实现这个算法，我们需要计算出在等式右边的偏导数的值，我们首先解决只有一个训练集时的情况，这样子可以暂时不考虑 J 的连乘情况，这样我们就有：

$$\begin{aligned}
\frac{\partial}{\partial \theta_j} J(\theta) &= \frac{\partial}{\partial \theta_j} \frac{1}{2} (h_\theta(x) - y)^2 \\
&= (h_\theta(x) - y) \frac{\partial}{\partial \theta_j} (\theta_1 x_1 + ... + \theta_j x_j + ... + \theta_n x_n - y) \\
&= (h_\theta(x) - y) x_j
\end{aligned}$$

这样以来带入到上面的等式就能得到对于已有一个训练样本的时候，更新函数规则就是：

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

这个规则就叫做 LMS 更新规则（LMS 代表的是最小均方），也就是 Widrow-Hoff 学习规则。这个规则有许多比较自然地特性。例如说 error term $(y^{(i)} - h_\theta(x^{(i)}))$；因此例如我们遇到一个训练样本，发现我们的预测已经很接近真实的 $y^{(i)}$ 值了，这样就没有必要再去改变这些参数。当然了，如果我们的预测有很大的误差的话，那么参数就变化很大。

对于只有一个训练样本的规则我们已经得到了，也就是 LMS。那么对于修改大于一个样本的训练集这里有两个方法。方法一就是利用下面的算法：

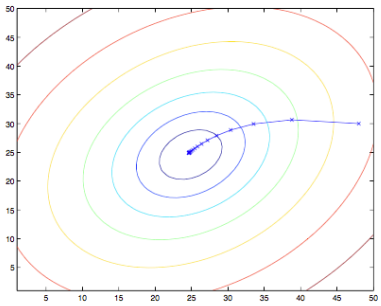
Repeat until convergence {

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

}

这个算法在每一步都对训练集中的参数进行了修改，因此这也叫做 batch 梯度下降。注意到，通常当梯度下降能够容易得到当前最小值，而我们在这儿提出的对于线性回归的最优问题只有一个全局的最优。因此，梯度下降总是收敛于全局的最小值（假设学习率 α 不是特别大）。的确，J 是一个凸的二次函数。这儿有一个梯度下降的例子，最小化二次函数。

那么在讲义上就有两幅图，第一幅是椭圆形状的。它展示了二次函数的一个轮廓，以及梯度下降的一个轨迹。初始化的值在 (48,30)。而打叉的那些就是通过梯度下降成功得到的 θ 值。



运行批处理的梯度下降在前面的数据集中找到适合的 θ，我们得到了 $\theta_0 = 71.27$ 和 $\theta_1 = 0.1345$。如果我们将该函数画出来，并加上训练数据，可以获得以下只有一条直线的图。

如果多了一个参数 bedroom，我们有得到了 $\theta_0 = 89.60$ 、$\theta_1 = 0.1394$ 以及 $\theta_2 = -8.738$

现在再考虑完成另一种梯度下降，看如下算法：

Loop {

for i=1 to m,{

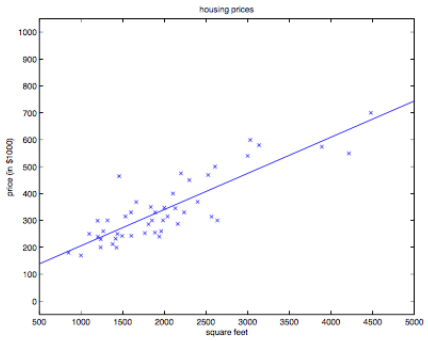
$\theta_j := \theta_j + \alpha(y^{(i)} - h_\theta(x^{(i)}))x_j^i \ (for\ very\ j)$

}

}

在这样的一个算法中，我们重复地在训练集中运行，每次我们得到一个训练样本，我们只根据每一个训练样本的梯度差来更新参数。这样的一个算法就叫做随机梯度下降 (stochastic gradient descent)。而批处理的梯度下降 (batch gradient descent) 在执行每一步之前都要扫描整个训练集，如果训练集的规模很大的话，它的开销也会很大。随机梯度下降会根据每一个样本来更新参数，通常它得到的接近于最小的参数 θ 要花费更短的时间。

（当然，它也有可能无法收敛，而只能在最小值附近震荡。）因此，当训练数据更大的情况下，我们更喜欢用随机梯度下降。



2.2 The normal equations

梯度下降提供了一种最小化 J 的方式。接下来我们讨论第二种方式，这一次的明确地进行了最小化并且没有用到循环算法。在这个方法中，我们通过求 θ_j 的导函数，然后令它们都为 0。那么显然在这种情况下，利用矩阵更适合计算。

2.2.1 Matrix derivatives(矩阵导数)

对于一个函数 $f:\mathbb{R}^{m\times n}\mapsto\mathbb{R}$ 是从 m x n 的矩阵到实数的映射。我们定义 f 对 A 的导数为:

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

因此,梯度 $\nabla_A f(A)$ 是一个 m x n 的矩阵,它的 (i,j) -元素就是 $\frac{\partial f}{\partial A_{ij}}$。例如,假设矩阵 $A=\begin{bmatrix}A_{11} & A_{12}\\ A_{21} & A_{22}\end{bmatrix}$ 是一个 2 乘 2 的矩阵,并且函数 $f:\mathbb{R}^{2\times 2}\mapsto\mathbb{R}$ 如下:

$$f(A)=\tfrac{3}{2}A_{11}+5A_{12}^2+A_{21}A_{22}$$

然后根据矩阵的偏导数公式，我们就能得到下面的等式:

$$\begin{bmatrix}\frac{3}{2} & 10A_{12}\\ A_{22} & A_{21}\end{bmatrix}$$

接着根据矩阵中的迹 trace 的概念，我们就有:

$$trA=\sum_{i=1}^{n}A_{ii}$$

2.2.2 Least squares revisited

那么根据矩阵导数这个方法，现在让我们来找一下令 $J(\theta)$ 最小的参数值 θ, 因此我们需要将 $J(\theta)$ 用矩阵向量的方式重写一下:

首先是训练样本的输入值，那么矩阵 X 就是下面的矩阵，它是一个 m 行 n 列的矩阵，如果加上一个截距项的话就是 m 行 n+1 列的矩阵:

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

自然 $\vec{y}$ 便是一个 m 维的向量，该向量包含了训练集的目标值：

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

因此，由于 $h_\theta(x^{(i)}) = (x^{(i)})^T\theta$

那么我们就能简单地得到：

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

然后运用向量中与自身相乘等于迹，即 $z^Tz = \sum_i z_i^2$ 可以得到：

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

最后最小化 J，则要找到关于 θ 的导数，根据公式

$$\nabla_{A^T} trABA^TC = B^TA^TC^T + BA^TC$$

得到

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

中间的计算过程可以查看 Ng 的讲义。这样令该式子为 0 的话就能够得到一个 normal equations：

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

。

那么这样就能够得到使得 $J(\theta)$ 最小的 θ 值了，即：

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

（结合 lecture 理解）

2.3 概率论解释

当我们面临回归问题时，为何是线性回归并且为何最小二乘的 cost function J 是合理的选择？这一节就会给出一系列概率假设，揭示最小二乘回归的原理。我们通过以下的等式表示目标参数和输入参数

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

, 这里的 $\epsilon^{(i)}$ 是误差项，表示未参与建模的因素产生的误差，并且我们进一步假设该参数是独立同分布的 IID，并且遵从期望为 0，方差为 σ^2 的高斯分布，即 $\epsilon^{(i)} \sim \mathcal{N}(0, \sigma^2)$。其概率密度函数是：

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

这能导出

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

这里的 $p(y^{(i)}|x^{(i)};\theta)$ 表示这是参数 θ 下的 $x^{(i)}$ 条件下的 $y^{(i)}$ 分布。注意到我们不需要在条件 θ 下，因为这里的 θ 不是一个随机变量。我们还可以将其写为 $y^{(i)}|x^{(i)};\theta \sim \mathcal{N}(\theta^T x^{(i)}, \sigma^2)$。

给定 X 和 θ,$y^{(i)}$ 的分布是什么呢？数据的概率是 $p(\vec{y}|X;\theta)$。对于固定的 θ，$\vec{y}$ 的函数。那么我们称其为似然函数（likelihood function）：

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

注意，由于在 $\epsilon^{(i)}$ 上的独立假设，则能写成：

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

如今给定了与 $y^{(i)}$ 和 $x^{(i)}$ 相关的概率模型，那么什么方法可以最好地确定出参数 θ 呢？那么最大似然估计告诉我们需要选择尽可能使得概率更高的 θ 值，也就是选择 θ 使得 $L(\theta)$ 最大。

我们可以求 $L(\theta)$ 的对数似然的最大值，

$$\begin{aligned}\ell(\theta) &= logL(\theta)\\ &= log\prod_{i=1}^{m}\frac{1}{\sqrt{2\pi}\sigma}exp(-\frac{y^{(i)}-\theta^T x^{(i)}}{2\sigma^2})\\ &= \sum_{i=1}^{m}\frac{1}{\sqrt{2\pi}\sigma}exp(-\frac{y^{(i)}-\theta^T x^{(i)}}{2\sigma^2})\\ &= mlog\frac{1}{\sqrt{2\pi}\sigma}-\frac{1}{\sigma^2}\frac{1}{2}\sum_{i=1}^{m}(y^{(i)}-\theta^T x^{(i)})^2\end{aligned}$$

因此，想要最大化 $\ell(\theta)$ 就转为了最小化

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

这就是我们的 $J(\theta)$，即最开始的最小二乘代价函数，所以会选择计算它。

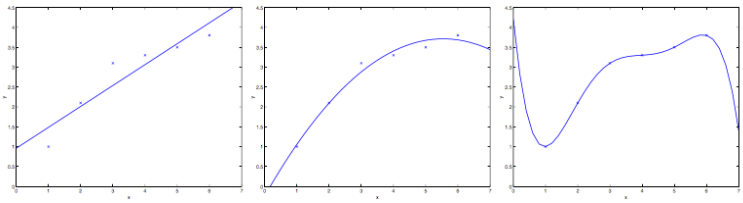
总结：在之前数据的概率假设中，最小二乘回归寻找了 θ 最大似然估计。因此在一系列假设下，最小二乘回归可以被证明为非常自然的函数以用来计算最大似然估计。

还要注意的是，在我们先前的讨论中，最后的对 θ 的选择不依赖于 σ^2，在不知道 σ^2 的情况下，我们的确可以得到同样的结果。在之后的指数家族和生产线性模型还会讨论到它们。

2.4 局部权重线性回归

考虑问题：在 $x \in \mathbb{R}$ 上预测 y。下面最左边的图展现了一个拟合 $y = \theta_0 + \theta_1 x$ 的数据。我们可以看到，其实数据并没有真正全落在这个直线上，因此最左边的不是十分适合。如果能够在刚才的式子上增加一个额外的特征 x^2，使其拟合 $y = \theta_0 + \theta_1 x + \theta^2 x^2$，那么我们就能够获得一个更适合数据的曲线。所以，一般都会天真地认为增加的特征越多，得到的函数就越拟合数据。显然，增加太多的特征是很危险的：在最右边的图中是拟合了 5 次的函数 $y = \sum_{j=0}^{5}\theta_j x^j$。虽然我们在图中看出该函数完全覆盖了这些数据，但我们不指望它是一个好的预测器，比如根据不同的公寓面积来预测它的价格。用更加专业的术语来描述它们，最左边的称为是欠拟合（underfitting）——不能够捕获到数据的结构，那么最右边的称为是过拟合（overfitting）。在之

后的的学习理论中会我们将会给出这些概念更形式化的表示，并且会更仔细的定义出一个好的假设函数与一个不好的假设函数的含义是什么。



在之前的例子中，我们会发现，特征（feature）的选择对于确保一个学习算法的高性能是十分重要的。（当我们讨论模型选择时，我们会发现算法会自动的选择一系列的特征）在这一节中，我们会讨论一下局部权重线性回归（LWR）算法，它假设有充足的训练数据，并且能够降低特征的选择对模型性能的影响。在作业中你会碰到的。

在之前的线性回归算法中，在一组点 x 上预测，我们会：

1. 得到 θ，最小化 $\sum_i (y^{(i)} - \theta^T x^{(i)})^2$；
2. 输出 $\theta^T x$。

相反，局部权重线性回归做一下的事情：

1. 得到 θ，最小化 $\sum_i \omega^{(i)} (y^{(i)} - \theta^T x^{(i)})^2$；
2. 输出 $\theta^T x$。

这里的 $\omega^{(i)}$ 是非负值 weights。直观地看，如果 $\omega^{(i)}$ 对某个 i 很大，那么在挑选 θ 时，我们会很难使得 $(y^{(i)} - \theta^T x^{(i)})^2$ 变小。如果 $\omega^{(i)}$ 较小，那么误差项 $(y^{(i)} - \theta^T x^{(i)})^2$ 在拟合中就会被忽略。

一个标准的权值为：

$$\omega^{(i)} = exp(-\frac{(x^{(i)} - x)^2}{2\tau^2})$$

注意，权值 weights 依赖于特定想要预测的对象 x。如果 $(x^{(i)} - x)^2$ 较小，那么 $\omega^{(i)}$ 就接近于 1；如果 $(x^{(i)} - x)^2$ 较大，那么 $\omega^{(i)}$ 就小。因此，在预测点 x 附近的训练样本，挑选 θ 会有更高的权值。（注意，权值的形式看上去像高斯分布，但是它与高斯没有任何直接的联系，并且 $\omega^{(i)}$ 并不是随机变量，正态分布什么的。）参数 τ 控制了训练样本的权值随它的 $x^{(i)}$ 到预测点 x 的距离的滑落的速度。τ 被称为是带宽参数（bandwidth），作业中也会有实验。

局部权重线性回归是我们见过的第一个非参数化（non-parametric）算

法的例子。之前我们见过的回归算法是带参数的（parametric）学习算法，因为其有固定的有限数量的参数（θ_i）来拟合数据。一旦我们拟合了 θ_i 并存储，我们就不再需要保留训练样本来进行预测了。相反，利用局部权重线性回归（locally weighted linear regression），我们需要整个训练集。非参数化指的是，为了代表假设函数 h，所要保留的 stuff 的数量与训练集随着训练集的大小，呈线性增长，

3 分类与逻辑回归

现在我们来讨论下逻辑回归问题。这个问题与回归问题相似，只不过现在预测的 y 值是一些离散的值。现在我们先考虑二分类问题，也就是 y 只能取 0 和 1。（我们讲到的大多数都可以适用于多分类的问题。）例如垃圾邮件分类器，那么 $x^{(i)}$ 就是邮件的一些特征，如果是垃圾邮件，那么 y 就会是 1，否则 y 就是 0。0 也叫做 negative class，1 叫做 positive class，它们有时也会用“-”和“+”表示。给定 $x^{(i)}$，那么对应的 $y^{(i)}$ 就会称为是训练样本的标签。

3.1 逻辑回归 Logistic regression

当我们解决分类问题时，可以忽视 y 是离散的值，利用我们之前的线性回归算法给定 x 来尝试预测 y。然而却很容易构造一些例子，在这些例子中这样的方法执行起来的效果很差。同样，只是取大于 1 或者小于 0 的 $h_\theta(x)$ 并没有什么意义。

因此，我们修改假设函数 $h_\theta(x)$ 的形式，我们选择

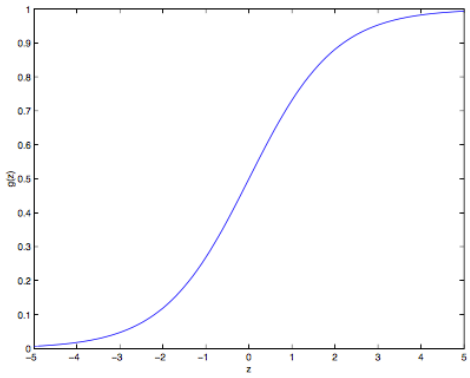
$$h_\theta(x) = g(\theta^T x) = \frac{1}{1 + e^{-\theta^T x}}$$

，而其中的

$$g(z) = \frac{1}{1 + e^{-z}}$$

叫做时逻辑函数（logistic function）或者是 sigmoid 函数（sigmoid function）。下面是 $g(z)$ 的图像：

注意到，当 z 趋向于正无穷的时候，$g(z)$ 趋向于 1，而当 z 趋向于负无穷的时候，$g(z)$ 趋向于 0。因此，$g(z)$，同时也就是 $h(x)$ 就是在界 0 和 1 之间。和之前一样，先令 $x_0 = 1$，那么 $\theta^T x = \theta_0 + \sum_{j=1}^{n} \theta_j x_j$。



现在我们假设 g 是给定了。其他界从 0 到 1 的函数也能用，但是由于一些原因，logistic 函数是更好的（之后会提到，关于 GLMS 生成学习算法）。在这之前，给出一个 sigmoid 函数的导数 g' 的性质：

$$\begin{aligned} g'(z) &= \frac{d}{dz}\frac{1}{1+e^{-z}} \\ &= \frac{1}{(1+e^{-z})^2}(e^{-z}) \\ &= \frac{1}{(1+e^{-z})}\cdot(1-\frac{1}{1+e^{-z}}) \\ &= g(z)(1-g(z)) \end{aligned}$$

那么给定了 logsitic 模型，我们要怎样拟合 θ 呢？根据在一系列假设下的最大似然估计得到的最小二次回归，在一系列概率假设下建立我们的分类模型，然后通过最大似然估计参数。

我们假设

$$P(y=1|x;\theta) = h_\theta(x)$$

$$P(y=0|x;\theta) = 1 - h_\theta(x)$$

这个可以写在一起:

$$p(y|x;\theta) = (h_{\theta(x)})^y(1-h_{\theta(x)})^{1-y}$$

假设 m 个训练样本是独立的，我们就能写出参数的概率：

$$\begin{aligned} L(\theta) &= p(\vec{y}|X;\theta) \\ &= \prod_{i=1}^{m} p(y^{(i)}|x^{(i)};\theta) \\ &= \prod_{i=1}^{m} (h_\theta(x^{(i)}))^{y^{(i)}} (1-h_\theta(x^{(i)}))^{1-y^{(i)}} \end{aligned}$$

跟之前一样，取 log:

$$\begin{aligned} \ell(\theta) &= logL(\theta) \\ &= \sum_{i=1}^{m} y^{(i)} logh(x^{(i)}) + (1-y^{(i)}) log(1-h(x^{(i)})) \end{aligned}$$

我们如何最大化概率？与线性回归的偏导类似，我们可以用梯度上升。使用向量符号，因此有 $\theta := \theta + \alpha \nabla_\theta \ell(\theta)$。注意，这里用的是“+“号，那是因为现在是最大化，梯度上升。我们先用一个训练样本 (x, y)，然后求随机梯度上升:

$$\begin{aligned} \frac{\partial}{\partial \theta_j} \ell(\theta) &= (y \frac{1}{g(\theta^T x)} - (1-y) \frac{1}{1-g(\theta^T x)}) \frac{\partial}{\partial \theta_j} g(\theta^T x) \\ &= (y - h_\theta(x)) x_j \end{aligned}$$

那么利用 $g'(z) = g(z)(1-g(z))$ ，我们就能够得到一个随机梯度上升的规则:

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

如果我们将其与 LMS 的更新规则比较，我们会发现它看起来是相同的。但是却不是同一个算法，因为这里的 $h_\theta(x^{(i)})$ 被定义成 $\theta^T x^{(i)}$ 的非线性的函数。不过惊讶的是，对于不同的算法和学习问题，竟然能够有相似的结果。那么在这样的结果后面是否有更深层次的缘由的？在 GLM 模型中，我们会回答这个问题。（在 problem set 1 中的 Q3 也有。）

3.2 题外话：感知器学习算法 The perceptron learning algorithm

我们先简单介绍一下一个有趣的算法，在之后的学习理论中会回到这一话题。考虑修改 logistic 回归方法来强制输出 0 或者 1 的值，这样一来就自然需要修改 g 的定义：

$$g(z) = \begin{cases} 1 & \text{if } z \geq 0, \\ 0 & \text{if } z < 0 \end{cases}$$

如果用之前的 $h_\theta(x) = g(\theta^T x)$ ，但是 g 用修改后的定义，并且使用更新规则

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

那么我们就有感知器学习算法。

注意，表面上感知器算法与其他的算法相似，但是它实际上和 logistic 回归和最小二乘线性回归不同。特别是在有意义的概率假设解释下建立该算法，或者得到基于最大似然估计算法的感知器算法。

3.3 另一个最大化 $\ell(\theta)$ 的算法

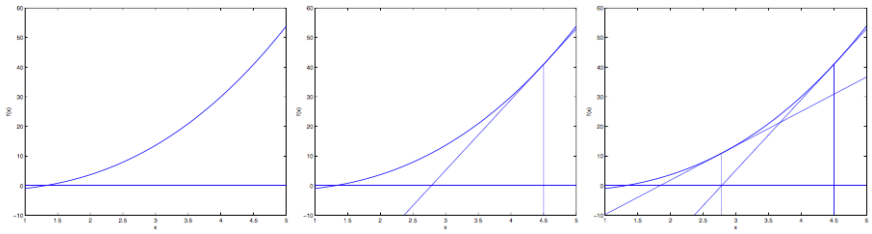
回到带有 $g(z)$ sigmoid 函数的逻辑回归，现在我们讨论一个不同的计算 $\ell(\theta)$ 最大值的算法。

首先我们考虑找出一个函数值为 0 的牛顿方法（Newton's method）。具体地说就是，假设我们有函数 $f : \mathbb{R} \mapsto \mathbb{R}$，然后我们想要找到一个值 θ 使得 $f(\theta) = 0$。这里 $\theta \in \mathbb{R}$，牛顿方法执行下面的迭代更新：

$$\theta := \theta - \frac{f(\theta)}{f'(\theta)}$$

。对于这个方法的一个直观的解释就是，随机猜测一个 θ 值，通过一个线性函数，估计函数 f，当线性函数等于 0 时，将此时的 θ 值作为下一次的猜测值。下面的图就是该方法的过程：

从最左边的图中我们可以看到函数 f 以及直线 $y = 0$。我们想要找到 θ 使得 $f(\theta) = 0$。那么我们知道大概是 θ 为 1.3 时。假设初始值 θ 为 4.5，牛顿方法就会在 $\theta = 4.5$ 处得到一个切线，然后找到哪边为 0（第二幅图）。同时就给出下一个 θ，大概为 2.8。最右边的图可以看出，这是可以更新到 θ 值为 1.8。经过几次循环后就能到达 1.3。



因此牛顿方法提供了一个求值 θ 使得 $f(\theta) = 0$ 的方式。那么将其用在对函数 $\ell(\theta)$ 上，使得 $f(\theta) = \ell'(\theta)$，我们就能用以下的更新规则：

$$\theta := \theta - \frac{\ell'(\theta)}{\ell''(\theta)}$$

(这里可以思考一下，怎样用牛顿方法来求最小值？)

最后，由于在我们的逻辑回归设定中 θ 是一个向量值，因此我们需要泛化牛顿方法到该设定。泛化牛顿方法到多维下（也叫做 Newton-Raphson method)：

$$\theta := \theta - H^{-1}\nabla_\theta \ell(\theta)$$

这里的 $\nabla_\theta \ell(\theta)$ 是 $\ell(\theta)$ 关于 θ_i 的偏导；H 是一个 n*n 的矩阵（如果加上截距项就是 n+1*n+1)，叫做 Hessian：

$$H_{ij} = \frac{\partial^2 \ell(\theta)}{\partial \theta_i \partial \theta_j}$$

牛顿方法要比（批处理）梯度下降有更快的收敛速度，并且到达最小值需要的循环次数更少。但是牛顿方法中的一次循环的代价要高于梯度下降，因为它要找到并得到 n*n 矩阵 Hessian 的转置；但是只要 n 不是太大的话，牛顿方法还是快许多的。当该方法应用到最大化逻辑回归的似然函数 $\ell(\theta)$ 的 log 时，得到结果的方法也被称作是 Fisher scoring。

4 Generalized Linear Models

迄今为止，我们讲过了一个回归的例子和一个分类的例子。在回归的例子中，我们有 $y|x;\theta \sim \mathcal{N}(\mu, \sigma^2)$，在分类问题中，我们有 $y|x;\theta \sim Bernoulli(\phi)$，而 μ ϕ 都是关于 x 和 θ 的适当的定义。在这一节中，我们会知道这些方法是模型家族中的特殊的方法，这个更丰富的模型家族叫做广义线性模型 GLMs。（注：本节展示的内容受以下两份作品的启发：

Michael I. Jordan, Learning in graphical models (unpublished book draft), 以及 McCullagh and Nelder, Generalized Linear Models (2nd ed.)。）我们还会展示 GLMs 家族中的其他模型是怎样获得并运用到其他的分类和回归问题中的。

4.1 指数家族

在开始研究 GLMs 的时候，我们会开始定义指数家族的分布，我们说如果能写成如下的形式，那么一个分布的类属于指数家庭:

$$p(y;\eta) = b(y)exp(\eta^T T(y) - a(\eta))$$

在这里 η 叫做该分布的自然参数（natural parameter，同时也叫做 canonical parameter)。$T(y)$ 是 sufficient statistic（对于我们考虑的分布，通常都是 $T(y) = y$ 的情况)；$a(\eta)$ 是 log 分割函数（log partition function)；等式 $e^{-a(\eta)}$ 对于归一化常数十分重要，它确保了分布 $p(y;\eta)$ 在 y 上总和/整合到 1。

给定 T，a，b 定义一个参数为 η 的分布家族 (family)，根据不同的 η 就会得到家庭中不同的分布。

现在我们证明伯努利和高斯分布是指数家族分布的例子。（这里就是一些分布该如何写成指数家族分布的形式）

对于伯努利分布:

$$\begin{aligned} p(y;\phi) &= \phi^y(1-\phi)^{1-y} \\ &= exp(ylog\phi + (1-y)log(1-\phi)) \\ &= exp((log(\frac{\phi}{1-\phi}))y + log(1-\phi)) \end{aligned}$$

因此就能得到 $\eta = log(\phi/(1-\phi))$，那么这就能得到 $\phi = 1/(1+e^{-\eta})$，这就是我们熟悉的 sigmoid 函数! 在之后作为 GLM 得到逻辑回归还会遇到。用指数家族来对伯努利完成计算，还有:

$$\begin{aligned} T(y) &= y \\ a(\eta) &= -log(1-\phi) \\ &= log(1+e^{\eta}) \\ b(y) &= 1 \end{aligned}$$

因此，给定适当的 T，a，b，伯努利分布就能够写成指数家族的形式。

那么高斯分布呢？得到一个线性回归，σ^2 的值与最终的 θ 和 $h_\theta(x)$，令 $\sigma^2 = 1$，我们有：

$$\begin{aligned} p(y;\phi) &= \frac{1}{\sqrt{2\pi}} exp(-\frac{1}{2}((y-\mu))^2 \\ &= \frac{1}{\sqrt{2\pi}} exp(-\frac{1}{2}y^2) exp(\mu y - \frac{1}{2}\mu^2) \end{aligned}$$

因此，我们看出高斯分布也在指数家族中，其中：

$$\begin{aligned} \mu &= \eta \\ T(y) &= y \\ a(\eta) &= \eta^2/2 \\ &= \mu^2/2 \\ b(y) &= 1/\sqrt{2\pi} exp(-y^2/2) \end{aligned}$$

还有许多的分布是指数家族的，多重正轨分布、珀松；在下一节中将会讨论关于构造来自某些分布的通用的配方。

4.2 构造 GLMs

假设你想要建立一个模型来估计建立在诸如促销活动、最近广告、天气、星期的特征 x，在给定的时间内来到你的商店的人数 y。那么我们会知道泊松分布是比较适合的。那么知道了这个我们如何来建立模型呢？幸运的是，泊松是在指数家族分布中的，因此我们可以应用广义线性模型。这一节中就描述类似这些问题的构造方法。

更普遍的就是说，考虑一个分类问题或者回归问题，预测的是一个关于 x 的函数的随机变量值 y。用 GLM 来解决这类问题，我们需要为我们的模型关于给定的 x 的 y 的条件分布作出一些假设：

1. $y|x;\theta \sim ExponentialFamily(\eta)$,；例如给定 x 和 θ，y 的分布跟随着参数 η 的指数家族分布。
2. 给定 x，我们的目标是预测给定 x 下的 $T(y)$ 的期望值。在大多数例子中，

我们的 $T(y) = y$，这也就意味着我们想要通过学习好的假设 h 的预测输出 $h(x)$ 能够满足 $h(x) = E[y|x]$(注意，这一假设在 $h_\theta(x)$ 下的逻辑回归和线性回归都是满足的，例如在逻辑回归中，我们有 $h_\theta(x) = p(y = 1|x;\theta) = 0 \cdot p(y = 0|x;\theta) + 1 \cdot p(y = 1|x;\theta) = E[y|x;\theta]$)。
3. 自然参数 η 和输入 x 是线性相关的：$\eta = \theta^T x$。(或者，如果 η 是向量值，那么 $\eta_i = \theta_i^T x$)

第三个假设最不容易证明，因此最好就当作是设计的选项，而不是一个前提假设。(?) 这三个设计对象会让我们得到一类非常优雅的学习算法——GLMs，它有许多的优点，比如说容易学习。其次，最后的结果模型挺长对于建立基于 y 的不同分布的模型是非常有效的，例如我们可以简单的展示一下逻辑回归和最小二乘都能转为 GLMs。

4.2.1 Ordinary Least Squares

为了证明最小二乘是 GLM 家族中模型，考虑目标变量 y（response variable in GLM terminology）是连续的，并且我们建立基于 x 的条件分布 y 是一个高斯分布 $\mathcal{N}(\mu, \sigma^2)$。(这里的 μ 也许依赖于 x。）然后我们令上面的 $ExponentialFamily(\eta)$ 分布是高斯分布。正如之前的过程，我们得到了结果，即 $\mu = \eta$。因此我们有：

$$\begin{aligned} h_\theta(x) &= E[y|x;\theta] \\ &= \mu \\ &= \eta \\ &= \theta^T x \end{aligned}$$

第一个等号是来自于假设 2，第二个等式来自 $y|x;\theta \sim \mathcal{N}(\mu, \sigma^2)$，它的期望值就是给定的 μ。第三个等式来自于假设 1；最后一个等式来自来于假设 3。

4.2.2 Logistic Regression

现在我们考虑逻辑回归。这里指考虑二分情况，因此 $y \in \{0, 1\}$。因为 y 只有两个值，那么选择伯努利分布来建立给定 x 的 y 的条件分布，我们有 $\phi = \frac{1}{1+e^{-\eta}}$。还要注意到，如果 $y|x;\theta \sim Bernoulli(\phi)$，那么 $E[y|x;\theta] = \phi$，因此，有：

$$\begin{aligned} h_\theta(x) &= E[y|x;\theta] \\ &= \phi \\ &= \frac{1}{1+e^{-\eta}} \\ &= \frac{1}{1+e^{-\theta^T x}} \end{aligned}$$

因此，这就给了我们形式为 $h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$ 的假设函数，如果你在想知道我们怎么想到逻辑回归的形式 $\frac{1}{1+e^{-z}}$ 的，那么答案是：一旦我们假设建立在 x 上的 y 的条件分布是伯努利，那就联想到了 GLMs 以及其指数家庭分布。

引入一些新的术语，将函数的自然参数作为分布的均值的函数 g，其中 g 为 $g(\eta) = E[T(y);\eta]$ 称为 canonical response 函数。它的反函数 g^{-1} 叫做 canonical link 函数。因此，g 与高斯家族相等，而 g^{-1} 则为 logistic 函数。

4.2.3 Softmax Regression

现在我们再多介绍一个 GLM 的例子。考虑一个分类问题，response 变量 y 可以取 k 个值中的一个，因此 $y \in \{1, 2, ..., k\}$。例如，不像之前只分两类的垃圾邮箱分类器，这个例子可以将邮件分为三类——垃圾邮件、个人邮件、工作邮件。因此 y 依旧是离散的，但是可以取大于两个的值。因此我们将其分布称为多项分布（multinomial distribution）。

让我们为这样的多项分布数据建立模型得到 GLM。所以我们要将多项分布表示为指数家族分布。

对于一个多项分布，我们可能需要 k 个参数来表示其不同的 k 个可能的输出值，但是这样有些冗余，因为它们不一定是相互独立的，根据这些参数最后的概率和为 1，我们可以知道其实只需要 k-1 个参数就可以。因此我们设定 k-1 个参数：$\phi_1, ..., \phi_{k-1}$，这里 $\phi_i = p(y = i;\phi)$，并且 $p(y = k;\phi) = 1 - \sum_{i=1}^{k-1} \phi_i$，为了表示简单，我们令 $\phi_k = 1 - \sum_{i=1}^{k-1} \phi_i$，但是我们要知道，这不是一个参数，它其实是用前面 k-1 个参试表示的。

为了将多项分布表示为指数家族分布，我们需要定义 $T(y) \in \mathbb{R}^{k-1}$：

$$\text{T(1)}=\begin{bmatrix}1\\0\\0\\\vdots\\0\end{bmatrix},\text{T(2)}=\begin{bmatrix}0\\1\\0\\\vdots\\0\end{bmatrix},\text{T(3)}=\begin{bmatrix}0\\0\\1\\\vdots\\0\end{bmatrix},...,\text{T(k-1)}=\begin{bmatrix}0\\0\\0\\\vdots\\1\end{bmatrix},\text{T(k)}=\begin{bmatrix}0\\0\\0\\\vdots\\0\end{bmatrix},$$

这里的 $T(y)$ 不再等于 y，这里的 $T(y)$ 是一个 k-1 维的向量，而不是一个实数。我们写成 $(T(y))_i$ 来表示向量 $T(y)$ 的第 i 个元素。

我们引入一个更加有用的符号。一个指示函数 $1\{\cdot\}$，如果后面是 true，则等于 1；否则为 0（$1\{True\}=1$，$1\{False\}=0$。）例如说 $1\{2=3\}=0$，$1\{3=5-2\}=1$。因此我们可以将 T(y) 和 y 的关系写 $(T(y))_i=1\{y=i\}$。接着，$E[(T(y))_i]=P(y=i)=\phi_i$，接下来我们证明多项分布也是指数家族。我们有：

$$\begin{aligned}
p(y;\phi) &= \phi_1^{1\{y=1\}}\phi_2^{1\{y=2\}}...\phi_k^{1\{y=k\}}\\
&= \phi_1^{1\{y=1\}}\phi_2^{1\{y=2\}}...\phi_k^{1-\sum_{i=1}^{k-1}1\{y=i\}}\\
&= \phi_1^{(T(y))_1}\phi_2^{(T(y))_2}...\phi_k^{1-\sum_{i=1}^{k-1}(T(y))_i}\\
&= exp((T(y))_1 log(\phi_1/\phi_k)+(T(y))_2 log(\phi_2/\phi_k)+...+(T(y))_{k-1}log(\phi_{k-1}/\phi_k))\\
&= b(y)exp(\eta^T T(y)-a(\eta))
\end{aligned}$$

这里的 $\eta=\begin{bmatrix}log(\phi_1/\phi_k)\\log(\phi_2/\phi_k)\\\vdots\\log(\phi_{k-1}/\phi_k)\end{bmatrix}$，

$a(\eta)=-log(\phi_k)$

$b(y)=1$

这样就完成了将多项分布转换成指数家族分布，那么 link 函数（对于 i=1,2,...,k）就是:

$$\eta_i=log\frac{\phi_i}{\phi_k}$$

方便起见，我们定义 $\eta_k=log(\phi_k/\phi_k)=0$。求它的转置并得到 response 函数，我们因此有：

$$
\begin{aligned}
e^{\eta_i} &= \frac{\phi_i}{\phi_k} \\
\phi_k e^{\eta_i} &= \phi_i \\
\phi_k \sum_{i=1}^{k} e^{\eta_i} &= \sum_{i=1}^{k} \phi_i = 1
\end{aligned}
$$

那么就能导出

$$\phi_i = \frac{e^{\eta_i}}{\sum_{j=1}^{k} e^{\eta_j}}$$

该函数从 η 映射到 ϕ 被称为是 softmax 函数。

那么要完成我们的模型，就要用到假设 3，也即是 η_i 与 x 是线性相关的。因此，$\eta_i = \theta_i^T x$ ，对于 i=1,2,...,k-1，这里的 $\theta_1, ..., \theta_k \in \mathbb{R}^{n+1}$ 是模型的参数。为了方便，我们还需定义 $\theta_k = 0$，以便 $\eta_k = \theta_i^T x = 0$。因此我们的模型假设下面的分布：

$$
\begin{aligned}
p(y = i|x; \theta) &= \phi_i \\
&= \frac{e^{\eta_i}}{\sum_{j=1}^{k} e^{\eta_j}} \\
&= \frac{e^{\theta_i^T x}}{\sum_{j=1}^{k} e^{\theta_j^T x}}
\end{aligned}
$$

这个可以应用到多分类问题的模型叫做 softmax 回归，它是 logistic 回归的泛化。

我们的假设函数就是：

$$
\begin{aligned}
h_\theta(x) &= E[T(y)|x;\theta] \\
&= E\begin{bmatrix} 1\{y=1\} \\ 1\{y=2\} \\ \vdots \\ 1\{y=k-1\} \end{bmatrix} \quad | \quad x;\theta \\
&= \begin{bmatrix} \phi_1 \\ \phi_2 \\ \vdots \\ \phi_{k-1} \end{bmatrix} \\
&= \begin{bmatrix} \frac{exp(\theta_1^T x)}{\sum_{j=1}^k exp(\theta_j^T x)} \\ \frac{exp(\theta_2^T x)}{\sum_{j=1}^k exp(\theta_j^T x)} \\ \vdots \\ \frac{exp(\theta_{k-1}^T x)}{\sum_{j=1}^k exp(\theta_j^T x)} \end{bmatrix}
\end{aligned}
$$

换句话说，我们的假设会输出估计的概率值 $p(y=i|x;\theta)$（i=1,...,k; 虽然 $h_\theta(x)$ 只有 k-1 维，但是 $p(y=k|x;\theta)$ 可以用 $1-\sum_{i=1}^{k-1}\phi_i$ 表示）。

最后，我们讨论一下参数拟合。与之前提到的最小二乘法呀逻辑回归相似，如果我们有一个规模为 m 个样本的训练集 $\{(x^{(i)},y^{(i)});i=1,...,m\}$，然后想来学习这个模型的参数 θ，那么我们得先写出 log 似然函数：

$$
\begin{aligned}
\ell(\theta) &= \sum_{i=1}^m logp(y^{(i)}|x^{(i)};\theta) \\
&= \sum_{i=1}^m log \prod_{l=1}^k (\frac{e^{\theta_l^T x^{(i)}}}{\sum_{j=1}^k e^{\theta_j^T x^{(i)}}})^{\{y^{(i)}=l\}}
\end{aligned}
$$

为了得到第二个等式，我们用到了上面的公式，我们因此可以利用类似于梯度下降或者牛顿方法通过最大化 $\ell(\theta)$ 来计算关于 θ 的最大似然估计。