Note for cs229-note1

iSea @ Jan. 4th, 2015

Content

---

1 线性回归（Linear Regression）

1.1 问题

为了预测房价模型，抽出的feature有房间大小和卧室数目两个，如果假定目标模型为线性的，那么

$$h(x) = \theta_0 + \theta_1 x_1 + \theta_1 x_2 = \sum_{i=0}^{n} \theta_i x_i = \theta^T x$$

学习θ的过程就是**线性回归**。定义cost function为：

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

学习的目标就是使cost function的值最小。

1.2 LMS：最小均方

梯度下降法（gradient descent）是选择θ的一个基本方法，它从一个起始值θ开始，不断用这个式子更新：

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

这种更新规则叫做LMS规则，即目标是least mean square。对于这个式子，在实际应用中有两种：

- **BGD（batch gradient descent）**，每次在遍历完所有数据后更新θ，也就是累加 $\sum_{i=1}^{m} (y^{(i)} - h_\theta(x^{(i)})) x_j^{(i)}$
- **SGD（stochastic gradient descent）**，每遍历一个数据项就更新一次θ

一旦数据规模（m）很大，那么BGD的收敛速度就会很慢，而SGD可以较快的收敛到一个目标值。但SGD可能会在最小值之间震荡，如果数据波动较大的话。

1.3 Normal Equations

除了迭代的方式来求θ，这里从矩阵的定义上来求推导θ的normal形式，也就是数学意义上的最优解。定义两个矩阵

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

那么

$$X\theta - \overrightarrow{y} = \begin{bmatrix} (x^{(1)})^T\theta \\ (x^{(2)})^T\theta \\ \vdots \\ (x^{(m)})^T\theta \end{bmatrix} - \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{bmatrix} = \begin{bmatrix} h_\theta(x^{(1)}) - y^{(1)} \\ h_\theta(x^{(2)}) - y^{(2)} \\ \vdots \\ h_\theta(x^{(m)}) - y^{(m)} \end{bmatrix}$$

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

目标是使这个cost function最小，对矩阵求导，求得

$$\nabla_\theta J(\theta) = X^TX\theta - X^T\overrightarrow{y}$$

令该式为0，得到

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

这就是求解θ的直接公式。但是求矩阵的逆比较慢，在m很大时，这也只能作为理论上的参考。

需要注意的一点是，当feature不是线性无关的，矩阵不是满秩的，自然也没有逆。如果矩阵仍然没有逆，需要用到后面讲到的regulaization。

1.4 误差函数

这里解释为什么选用LMS，也就是误差的平方和来作为误差的cost function。

可以假定数据集是满足

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

其中$\epsilon^{(i)} \sim \mathcal{N}(0, \sigma^2)$，即均值为0的高斯分布（正态分布）。

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

也就是$y^{(i)}$在给定$x^{(i)}$,θ的分布是

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

似然函数就是这些分布的乘积，极大似然原则要求最大化似然函数。而似然函数的log函数

$$l(\theta) = logL(\theta) = mlog\,\frac{1}{\sqrt{2\pi}\,\sigma} - \frac{1}{\sigma^2}\cdot\frac{1}{2}\sum_{i=1}^{m}(h_\theta(x^{(i)}-y^{(i)}))^2$$

因此cost function使用$\frac{1}{2}\sum_{i=1}^{m}(h_\theta(x^{(i)}-y^{(i)}))^2$是符合极大似然原则的。

1.5 带权值的线性回归

前面的回归模型中权重都是1，这里考虑带权重的线性回归。这里的权重是基于需要预测的query的数据决定的。在θ的学习过程中，前面方法的目标都是选取使$\sum_{i=1}^{m}(h_\theta(x^{(i)}-y^{(i)}))^2$最小的，而这里选取使$\sum_{i=1}^{m}w^{(i)}(h_\theta(x^{(i)}-y^{(i)}))^2$最小的。

其中

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

含义就是离目标样本越接近的样本权重就越大。

在带权重的线性回归中，参数θ的结果不是固定的，是会根据query样本的变化而变化的。这种方法叫做**非参数学习**。与**参数学习**不同点在于，后者一旦确定了参数，就不再需要training data，可以直接预测。而非参数学习需要始终持有这些数据来预测。

2 对数回归（Logistic Regression）

2.1 问题

在机器学习中，分类（classification）是一个很核心的应用。使用上一节的线性回归来处理这个问题的结果是很差的，因为分类问题的取值空间非常小，通常的，1和0。为了修正这个问题，引入

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

这里的

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

通常被称作**logistic函数**，或者**sigmoid函数**。使用这个函数的原因会在后面的GLM中给出。

对数回归的θ选取过程也可以使用梯度下降法，定义对数形式的cost function：

$$cost(h_\theta(x),y)=\begin{cases}log(h_\theta(x)) & if\ y=1\\ -log(1-h_\theta(x)) & if\ y=0\end{cases}$$
$$=-ylog(h_\theta(x))-(1-y)log(1-h_\theta(x))$$

这个定义可以这样理解：当$y=1$时，$h_\theta(x)$越大越好，反之则越小越好。

那么有

$$J(\theta)=-[\frac{1}{m}\sum_{i=1}^{m}-ylog(h_\theta(x))-(1-y)log(1-h_\theta(x))]$$

求导得到

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

利用这个来进行梯度下降过程的迭代。可以发现，这个公式与线性回归的迭代公式只有评估差距计算的区别，是很相似的。

2.2 枝桠：感知机学习算法

将前面的sigmoid函数替换成一个离散函数，强制输出结果为0或1，即

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

然后用梯度下降法来学习的过程就叫做**感知机学习算法**。这个算法最早提出于上世纪60年代，是一个很原始的分类模型。

2.3 牛顿法求解极大似然

除了梯度下降的迭代方法，牛顿法也是一个很有用的迭代方法。

牛顿迭代的过程是

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

应用到这里

$$\because f(\theta)=\frac{\partial}{\partial\theta_j}J(\theta)$$
$$\therefore\theta:=\theta-\frac{J'(\theta)}{J''(\theta)}$$

继续推导

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

其中$H_{ij} = \frac{\partial^2 l(\theta)}{\partial\theta_i \partial\theta_j}$，叫做**Hessian矩阵**。相比梯度下降法，牛顿法的收敛速度非常快，但是当feature数目较多时，求矩阵的逆的消耗增加（$O(N^3)$），导致耗时显著增加。

3 一般线性模型（Generalized Linear Model）

3.1 Exponential family

如果一个概率分布可以表示成

$$p(y|\theta) = b(y)exp(\eta^T \Phi(x) - A(\theta))$$

那么它属于Exponential family。前面应用的Bernoulli分布（指数回归）和Gaussian分布（线性回归）都属于Exponential family。可以将Bernoulli分布写作

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

这里$\eta = log(\phi/(1-\phi))$，转化一下就是前面的sigmoid函数：

$$\phi = \frac{1}{1+e^{-\eta}}$$

这也是指数回归使用sigmoid函数的来源。

3.2 构造GLM

为了构建GLM（Generalized Linear Model），需要满足三个假设：

1. $y|x;\theta \sim \mathrm{ExponetialFamily}(\eta)$
2. 模型的目标是为了在给定x时预测$\Phi(y)$。通常$\Phi(y) = y$，也就是预测目标$h(x) = E[y|x;\theta]$（比如对数回归中的$h_\theta(x)$就是y为1的期望）
3. $\eta = \theta^T x$

对于线性回归中用到的Gaussian分布$\mathcal{N}(\mu, \sigma^2)$，简单起见，假设$\sigma^2 = 1$，有

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

属于Exponetial Family，符合第一条。其中$\eta = \mu$，那么

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

第一个等式符合第二条，最后一个则符合第三条。

3.3 Softmax回归

假设在分类问题中，结果不止0和1，而是$y \in \{1, 2, \ldots, k\}$。比如分类邮件为垃圾邮件，订阅邮件、工作邮件和个人邮件。

使用k个参数$\phi_1, \phi_2, \ldots, \phi_k$来表示每种结果的概率，由于$\sum_{i=1}^{k} \phi_i = 1$，所以最后一个可以用其他表示，只需要$k-1$个。

定义$\varPhi(y) \in \mathbb{R}^{k-1}$如下：

$$
\varPhi(1) = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \varPhi(2) = \begin{bmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix}, \ldots, \varPhi(k-1) = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}, \varPhi(k) = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}
$$

这里$\varPhi(y) \neq y$，而是一个$k-1$维的向量。可以得到

$$
E[(\varPhi(y))_i] = P(y = i) = \phi_i
$$

转化成Exponetial Family，即

$$
\begin{aligned}
p(y;\phi) &= \phi_1^{(\varPhi(y))_1} \phi_2^{(\varPhi(y))_2} \ldots \phi_k^{1-\sum(\varPhi(y))_i} \\
&= exp((\varPhi(y))_1 log(\phi_1/\phi_k) + \ldots + (\varPhi(y))_{k-1} log(\phi_{k-1}/\phi_k) + log(\phi_k))
\end{aligned}
$$

其中

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

也即

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

计算出ϕ，即

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

得到概率分布

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

用极大似然的log函数

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

利用梯度下降或者牛顿迭代处理就可以了。这个过程就是**softmax回归。**

Reference

[1] http://cs229.stanford.edu/notes/cs229-notes1.pdf

[2] http://openclassroom.stanford.edu/MainFolder/CoursePage.php?course=MachineLearning