回归

YangXiao

2018 年 3 月 27 日

1　引理：向量的导数

$\boldsymbol{A}$ 为 $m \times n$ 的矩阵，$\boldsymbol{x}$ 为 $n \times 1$ 的列向量,
记 $\vec{y} = \boldsymbol{A} \cdot \vec{x}$
思考：$\frac{\partial \vec{y}}{\partial \vec{x}} =?$
向量导数的推导
令

$$\boldsymbol{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

$$\vec{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

$$\boldsymbol{A} \cdot \vec{x} = \begin{bmatrix} a_{11}x_1 + a_{12}x_2 \cdots + a_{1n}x_n \\ a_{21}x_2 + a_{22}x_2 \cdots + a_{2n}x_n \\ \vdots \\ a_{n1}x_1 + a_n 2x_2 \cdots + a_{nn}x_n \end{bmatrix}$$

从而，

$$\frac{\partial \vec{y}}{\partial \vec{x}} = \frac{\partial \boldsymbol{A} \cdot \vec{x}}{\partial \vec{x}} = \begin{bmatrix} a_{11} & a_{21} & \cdots & a_{m1} \\ a_{12} & a_{22} & \cdots & a_{m2} \\ \vdots & \vdots & \ddots & \vdots \\ a_{1n} & a_{2n} & \cdots & a_{mn} \end{bmatrix} = \boldsymbol{A}^T$$

结论与直接推广

向量偏导公式：

$\frac{\partial \boldsymbol{A} \cdot \vec{x}}{\partial \vec{x}} = \boldsymbol{A}^T$

$\frac{\partial \boldsymbol{A} \cdot \vec{x}}{\partial \vec{x}^T} = \boldsymbol{A}$

$\frac{\partial (\vec{x}^T \cdot \boldsymbol{A})}{\partial \vec{x}} = \boldsymbol{A}$

引理：标量对向量求导

$\boldsymbol{A}$ 为 $n \times n$ 的矩阵，$\boldsymbol{x}$ 为 $n \times 1$ 的列向量,

记 $y = \vec{x}^T \cdot \boldsymbol{A} \cdot \vec{x}$

同理可得：

$\frac{\partial y}{\partial \vec{x}} = \frac{\vec{x}^T \cdot \boldsymbol{A} \cdot \vec{x}}{\partial \vec{x}} = (\boldsymbol{A} + \boldsymbol{A}^T) \cdot \vec{x}$

若 $\boldsymbol{A}$ $\frac{\vec{x}^T \cdot \boldsymbol{A} \cdot \vec{x}}{\partial \vec{x}} = 2\boldsymbol{A}\vec{x}$

2 线性回归

$y = ax + b$ –单变量

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

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

使用极大似然估计解释最小二乘

$$y^{(i)} = \theta^T x^{(i)} + \epsilon^{(i)} \tag{2}$$

误差 $\epsilon^{(i)}(1 \leq i \leq m)$ 是独立同分布的，服从均值为 0，方差为某定值的 σ^2 的高斯分布

原因：中心极限定理。众多因素的独立影响的综合反应，往往近似服从正态分布。

似然函数

$$p(\epsilon^{(i)}) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(\epsilon^{(i)})^2}{2\sigma^2}\right) \tag{3}$$

$$p(y^{(i)}|x^{(i)};\theta) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}\right) \tag{4}$$

$$\begin{aligned} \mathcal{L}(\theta) &= \prod_{i=1}^{m} p(y^{(i)}|x^{(i)};\theta) & (5) \\ &= \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}\right) & (6) \end{aligned}$$

在得到似然函数后，使似然函数值最大，即假定给定 x 和 θ 后，使得 y 出现的可能性最大, 事实上 θ 未知，4 表示在 i 那个点处给定 x 和 θ $y^{(i)}$ 出现的概率，所以连乘所有的点就可以得到所有概率，然后求最大。我们可以对似然函数 L 求导，使其等于 0，得到极值点。
为社么在求导等于 0 可以保证似然函数值最大？
因为，指数族函数都是凹函数，凹函数极值点便是最值点。且为最大值点，我们平时看到的各种分布：高斯分布，泊松分布等可以写成指数族函数形式。下面即开始研究5 高斯对数与最小二乘的关系：

$$\begin{aligned} \mathcal{L}(\theta) &= \log L(\theta) & (7) \\ &= \log \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi}\sigma} exp\left(-\frac{(y^{(i)} - \theta^T x^{(i)})}{\sigma^2}\right) & (8) \\ &= \sum_{i=1}^{m} \log \frac{1}{\sqrt{2\pi}\sigma} exp\left(-\frac{(y^{(i)} - \theta^T x^{(i)})}{\sigma^2}\right) & (9) \\ &= m \log \frac{1}{\sqrt{2\pi}\sigma} - \frac{1}{\sigma^2} \cdot \frac{1}{2} \sum_{i=1}^{m} (y^{(i)} - \theta^T x^{(i)})^2 & (10) \end{aligned}$$

去掉无关常数项，所以目标函数变换为：

$$\mathcal{J}(\theta) = \frac{1}{2} \sum_{i=1}^{m} (h_(\theta))(x^{(i)}) - y^{(i)})^2 \tag{11}$$

假设 $\boldsymbol{X}$ 为 m 行，n 列的矩阵。m 表示有 m 个样本，n 表示每个样本有多少个参数（特征）。

目标函数11 转变为如下：

$$\mathcal{J}(\boldsymbol{\theta})=\frac{1}{2}(\boldsymbol{X}\boldsymbol{\theta}-\boldsymbol{y})^T(\boldsymbol{X}\boldsymbol{\theta}-\boldsymbol{y}) \tag{12}$$

对上式12 求导等于 0，过程如下：

$$\begin{aligned}
\nabla_{\boldsymbol{\theta}}\mathcal{J}(\boldsymbol{\theta}) &= \nabla_{\boldsymbol{\theta}}\left(\frac{1}{2}(\boldsymbol{\theta}^T\boldsymbol{X}^T-\boldsymbol{y}^T)(\boldsymbol{\theta}\boldsymbol{X}-\boldsymbol{y})\right) && (13)\\
&= \nabla_{\boldsymbol{\theta}}\left(\frac{1}{2}\boldsymbol{\theta}^T\boldsymbol{X}^T\boldsymbol{X}\theta-\theta^T\boldsymbol{X}^T\boldsymbol{y}-\boldsymbol{y}^T\boldsymbol{X}\theta+\boldsymbol{y}^T\boldsymbol{y}\right) && (14)\\
&= \frac{1}{2}\left(2\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{\theta}-\boldsymbol{X}^T\boldsymbol{y}-(\boldsymbol{y}^T\boldsymbol{X})^T\right) && (15)\\
&= \boldsymbol{X}^T\boldsymbol{X}\boldsymbol{\theta}-\boldsymbol{X}^T\boldsymbol{y} && (16)
\end{aligned}$$

令上式等于 0，可得 $\boldsymbol{\theta}$ 解析式：

$$\boldsymbol{\theta}=(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y} \tag{17}$$

若 $\boldsymbol{X}^T\boldsymbol{X}$ 不可逆，则需要加入 λ 扰动，即正则，如下：

$$\boldsymbol{\theta}=(\boldsymbol{X}^T\boldsymbol{X}+\lambda\boldsymbol{I})^{-1}\boldsymbol{X}^T\boldsymbol{y} \tag{18}$$

梯度下降算法最小化目标函数

- 初始化 $\boldsymbol{\theta}$(随机初始化）
- 迭代，新的 θ 能够使得 $\boldsymbol{J}(\boldsymbol{\theta})$ 更小
- 如果 $\boldsymbol{J}(\boldsymbol{\theta})$ 能继续减少，继续迭代
- $\boldsymbol{\theta}_j=\boldsymbol{\theta}_j-\alpha\frac{\partial\boldsymbol{J}(\boldsymbol{\theta})}{\partial\boldsymbol{\theta}_j}$
- 其中，α 称为学习率/步长

$$\begin{aligned}
\alpha\frac{\partial\boldsymbol{J}(\boldsymbol{\theta})}{\partial\theta_j} &= \alpha\frac{\partial}{\partial\boldsymbol{\theta}_j}\frac{1}{2}\left(h_\theta(x)-y\right)^2 && (19)\\
&= 2\cdot\frac{1}{2}\left(h_\theta(x)-y\right)\cdot\frac{\partial}{\partial\theta_j}\left(h_\theta(x)-y\right) && (20)\\
&= \left(h_\theta(x)-y\right)\cdot\frac{\partial}{\partial\theta_j}\left(\sum_{i=0}^{n}\theta_i x_i-y\right) && (21)\\
&= \left(h_\theta(x)-y\right)\boldsymbol{x}_j && (22)
\end{aligned}$$

其中 $\boldsymbol{x}_j$ 表示某一个向量的第 j 维，是标量

批处理梯度下降算法

$$\theta_j = \theta_j - \alpha \sum_{i=0}^{m} (h_\theta(x^i) - y^{(i)}) x_j^{(i)} \tag{23}$$

其中 $x_j^{(i)}$ 表示第 i 个向量的第 j 维，没更新一个 θ 需要遍历所有的样本，当样本很大时，显然需要的操作太多，下面介绍随机梯度下降

随机梯度下降——SGD

Loop

for i=1 to m:

$$\theta_j = \theta_j - \alpha (h_\theta(x^i) - y^{(i)}) x_j^{(i)} \tag{24}$$

3 logistic 回归

logistic 分布

设 X 是连续随机变量，若 X 满足 logistic 分布，则 X 具有下列分布函数：

$$F(x) = P(X \leq x) = \frac{1}{1 + exp^{-(x-\mu)/\gamma}} \tag{25}$$

其中 μ 为位置参数，$\gamma > 0$ 为形状参数

logistic 函数

$$g(z) = \frac{1}{1 + exp^{-z}} h_\theta(\boldsymbol{x}) = g(\boldsymbol{\theta}^T \boldsymbol{x}) = \frac{1}{1 + exp^{-\boldsymbol{\theta}^T \boldsymbol{x}}} \tag{26}$$

对 g(z) 求导得：

$$g^{'}(z) = g(z)\left(1 - g(z)\right) \tag{27}$$

假定：

$$P(y = 1 \mid \boldsymbol{x}; \boldsymbol{\theta}) = h_{\boldsymbol{\theta}}(\boldsymbol{x}) \tag{28}$$

$$P(y = 0 \mid \boldsymbol{x}; \boldsymbol{\theta}) = 1 - h_{\boldsymbol{\theta}}(\boldsymbol{x}) \tag{29}$$

给定 $\boldsymbol{x}$ 和 θ 分布律为：

$$p(\boldsymbol{y} \mid \boldsymbol{x}; \boldsymbol{\theta}) = \left(h_{\boldsymbol{\theta}}(\boldsymbol{x})\right)^y \left(1 - h_{\boldsymbol{\theta}}(\boldsymbol{x})\right)^{1-y} \tag{30}$$

似然函数：

$$
\begin{aligned}
\mathcal{L} &= p(\vec{y}|\boldsymbol{X};\boldsymbol{\theta}) && (31)\\
&= \prod_{i=1}^{m} p(y^{(i)}|x^{(i)};\boldsymbol{\theta}) && (32)\\
&= \prod_{i=1}^{m} p\left(h_\theta(x^{(i)})\right)^{y^{(i)}}\left(1-h_\theta(x^{(i)})\right)^{1-y^{(i)}} && (33)
\end{aligned}
$$

对数似然函数：

$$
\mathcal{L} = \log L(\boldsymbol{\theta}) \quad = \sum_{i=1}^{m} y^{(i)} \log h\left(x^{(i)}\right) + \left(1-y^{(i)}\right)\log\left(1-h(x^{(i)})\right) \qquad (34)
$$

$$
\begin{aligned}
\frac{\partial \mathcal{L}(\boldsymbol{\theta})}{\partial \theta_j} &= \left(y\frac{1}{g(\boldsymbol{\theta}^T\boldsymbol{X})} + (1-y)\frac{-1}{1-g(\boldsymbol{\theta}^T\boldsymbol{X})}\right)\frac{\partial g(\boldsymbol{\theta}^T\boldsymbol{X})}{\partial \theta_j} && (35)\\
&= \left(y\frac{1}{g(\boldsymbol{\theta}^T\boldsymbol{X})} + (1-y)\frac{-1}{1-g(\boldsymbol{\theta}^T\boldsymbol{X})}\right) g(\boldsymbol{\theta}^T\boldsymbol{X})(1-g(\boldsymbol{\theta}^T\boldsymbol{X}))\frac{\partial \theta^T\boldsymbol{X}}{\partial \boldsymbol{\theta}_j} && (36)\\
&= \left(y(1-g(\boldsymbol{\theta}^T\boldsymbol{X})) + (1-y)g(\boldsymbol{\theta}^T\boldsymbol{X})\right) x_j && (37)\\
&= (y-(\boldsymbol{\theta}^T\boldsymbol{X}))x_j && (38)\\
&= (y-h_\theta(\boldsymbol{X}))x_j && (39)
\end{aligned}
$$

logistic 回归参数学习规则：

批处理梯度下降算法

$$
\theta_j = \theta_j - \alpha\sum_{i=0}^{m}(y^{(i)} - h_\theta(x^i))x_j^{(i)} \qquad (40)
$$

其中 $x_j^{(i)}$ 表示第 i 个向量的第 j 维，没更新一个 θ 需要遍历所有的样本，当样本很大时，显然需要的操作太多，下面介绍随机梯度下降

随机梯度下降——SGD

Loop

for i=1 to m:

$$
\theta_j = \theta_j - \alpha(y^{(i)} - h_\theta(x^i))x_j^{(i)} \qquad (41)
$$

对比23,40和24,41。可知线性回归和 logistic 回归有相同形式。

对数线性模型

一个事件的几率 odds，是指该事件发生的概率与该事件不发生的概率的比值。

对数几率：logit 函数

$P(y=1 \mid \boldsymbol{x};\boldsymbol{\theta}) = h_\theta(\boldsymbol{x})$

$P(y=0 \mid \boldsymbol{x};\boldsymbol{\theta}) = 1 - h_\theta(x)$

$$
\begin{aligned}
logit(p) &= \log\frac{p}{1-p} &(42)\\
&= \log\frac{h_\theta(x)}{1-h_\theta(x)} &(43)\\
&= \log\left(\frac{\frac{1}{1+exp^{(-w^Tx)}}}{\frac{exp^{(-w^Tx)}}{1+exp^{(-w^Tx)}}}\right) &(44)\\
&= w^T\boldsymbol{x} &(45)
\end{aligned}
$$

所以 logistic 回归模型是，对输入 $\boldsymbol{x}$ 的数线性模型 $w^T\boldsymbol{x}$，由26可知，当线性模型 $w^T\boldsymbol{x}$ 越趋近无穷大，其概率值越接近 1，越接近负无穷，其概率值越趋近 0.

4 实验——多项式曲线拟合

分别使用直接求解析解、加入正则求解析解和随机梯度下降求解 问题背景： 在 [0,1] 的区间里，利用正弦函数加上均值为 0，方差为 0.2 的噪声。sin(2*πx)+$\mathcal{N}(0, 0.2)$ 生成 10 个点。

直接求解析解： 代码如下：

```
#定义多项式函数
def phi(self, x, m):
    _phi = [1]
    for i in range(m):
        _phi.append(x**(i+1))
    return _phi
```

```
        def Phi(self, X, m):
        _Phi = []
        for x in X:
            _Phi.append(self.phi(x, m))
        return np.array(_Phi)
    #y = w^T *x + w_0
    def predict(self, omgea, x):
        return omgea.dot(self.phi(x, len(omgea) - 1))
    #画预测函数
    def plot_predict(self, func, range, label = '',\\resolution = 0.02,color
        _x = np.arange(range[0], range[1], resolution)
        _y = [func(x) for x in _x]
        plt.plot(_x, _y, label = label, color = color)
    #分别在2，3，5，9维下进行多项式拟合
    def run(self, X, t):
        dimesion = [2, 3, 5, 9]
        range = [0 ,1]
        for idx, dim in enumerate(dimesion):
            _Phi = self.Phi(X, dim)
            #无正则，求解析解
            omega = inv(_Phi.T.dot(_Phi)).dot(_Phi.T).dot(t)
            #正则，求解析解
            lamda = 1e-3
            omega_re = inv(lamda*np.eye(dim + 1, dim + 1) +\\
            _Phi.T.dot(_Phi)).dot(_Phi.T).dot(t)
            #无正则预测
            _predict = partial(self.predict, omega)
            #加入正则预测
            _predict_re = partial(self.predict, omega_re)

            ax = plt.subplot(2, 2, idx +1)
            #w无正则画图
            self.plot_predict(_predict, range, label = 'no regularizaition',
```

```
                resolution = 0.015,color='red')
            #正则画图
            self.plot_predict(_predict_re, range, label='regularzaition',\\
                resolution = 0.015, color = 'green')
            plt.scatter(X, t, c = '', marker='o',edgecolors='blue')
            plt.title('Dim = %d'%dim)
            plt.legend()
        plt.show()
```

随机梯度下降：

```
    #随机梯度下降
    def sgd(self, X, t, omega, lr = 1e-2):
        for i in range(len(X)):
            for j in range(1,len(omega)):
                omega[j] = omega[j] - lr*(self.predict \\
                (omega, X[i]) - t[i])*(X[i]**j)
        return omega
    #训练
    def train(self,X, t, dim, lr = 1e-2 ,epch = 20000):
        omega = np.zeros(dim)
        for i in range(epch):
            omega = self.sgd(X, t, omega)
        return omega
    #分别在2，3，5，9维训练
    def run_sgd(self,X ,t):
        demsion = [2, 3, 5, 9]
        range = [0, 1]
        for idx, dim in enumerate(demsion):
            omega = self.train(X, t, dim)

            _predict_sgd = partial(self.predict, omega)

            ax = plt.subplot(2, 2, idx + 1)
```

```
    # 随机梯度下降画图
    self.plot_predict(_predict_sgd, range, label='gradient_sgd',\ \\
    resolution=0.015, color='red')
    plt.title('dim %d'%dim)
    plt.scatter(X, t, c ='',marker='o' ,edgecolors='blue')
plt.show()
```

5 最大熵模型