Softmax 回归

摘要：Softmax 回归用于处理多分类问题，是 Logistic 回归的一种推广。这两种回归都是用回归的思想处理分类问题。这样做的一个优点就是输出的判断为概率值，便于直观理解和决策。下面我们介绍它的原理和实现。

1 原理

考虑 K 类问题，假设已知训练样本集 D 的 n 个样本 $\{(x_i, y_i) \mid i = 1, 2, 3, \ldots m\}$，其中，$x_i \in R^d$ 为特征向量，y_i 为样本类别标签，和一般而分类问题不同，Softmax 回归采用了标签向量来定义类别，其定义如下：

$$y_i = \begin{pmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{pmatrix} \tag{1}$$

标签向量为 0 – 1 的 K 维向量,若属于 k 类,则向量的 k 分量为 1,其他分量均为 0

为计算每个样本的所属类别概率，首先定义回归函数：

$$P(y_i=k\mid x_i,\theta)=\frac{e^{\theta_k^T x_i}}{\sum_{j=1}^{K} e^{\theta_j^T x_i}} \tag{2}$$

其中θ_k 为第k类的回归参数。根据回归函数，样本x_i的概率：

$$P(x_i\mid\theta_1,\theta_2,\ldots\theta_K)=\prod_{k=1}^{K} P(y=k\mid x,\theta)^{y_{ik}} \tag{3}$$

其中,$y_i=(y_{i1},y_{i2},\ldots,y_{iK})^T$为$x_i$的标签向量。

对于给定样本x_i，样本属于类别k的概率为：

$$P(y_i=k\mid x_i,\theta)=\frac{e^{\theta_k^T x_i}}{\sum_{j=1}^{K} e^{\theta_j^T x_i}} \tag{4}$$

可将概率罗列成矩阵形式:

$$\begin{bmatrix} P(y_i=1\mid x_i,\theta) \\ P(y_i=2\mid x_i,\theta) \\ P(y_i=3\mid x_i,\theta) \\ \vdots \\ P(y_i=K\mid x_i,\theta) \end{bmatrix}=\frac{1}{\sum_{j=1}^{K} e^{\theta_j^T x_i}}\begin{bmatrix} e^{\theta_1^T x_i} \\ e^{\theta_2^T x_i} \\ e^{\theta_3^T x_i} \\ \vdots \\ e^{\theta_K^T x_i} \end{bmatrix} \tag{5}$$

假设样本

$$x=\{x_i\mid i=1,2,3,\ldots m\}=\begin{bmatrix} 1 & x_{11} \cdots x_{1n} \\ 1 & x_{11} \cdots x_{1n} \\ & \vdots \\ 1 & x_{m1} \cdots x_{mn} \end{bmatrix} \tag{6}$$

将θ写在一个矩阵里

$$\theta=\begin{bmatrix} \theta_1^T \\ \theta_2^T \\ \vdots \\ \theta_K^T \end{bmatrix}=\begin{bmatrix} \theta_{10}\ \theta_{11} \cdots \theta_{1n} \\ \theta_{20}\ \theta_{21} \cdots \theta_{2n} \\ \vdots \\ \theta_{K0}\ \theta_{K1} \cdots \theta_{Kn} \end{bmatrix} \tag{7}$$

我们采用极大似然法估计回归参数θ。我们的目标是期望所有样本的获得概率最大化，因此构造如下似然函数：

$$L(\theta)=\prod_{i=1}^{m}P(y_i=k\mid x_i,\theta)=\prod_{i=1}^{m}\frac{e^{\theta_k^T x_i}}{\sum_{j=1}^{K}e^{\theta_j^T x_i}} \tag{8}$$

对第k类回归参数，定义损失函数：

$$J(\theta)=-\ln L(\theta)=\sum_{i=1}^{m}(-\theta_k^T x_i+\ln(\sum_{j=1}^{K}e^{\theta_j^T x_i})) \tag{9}$$

$$\nabla_\theta J(\theta)=\frac{\partial J(\theta)}{\partial\theta}=\begin{pmatrix}\frac{\partial J(\theta)}{\partial\theta_{10}} & \cdots & \frac{\partial J(\theta)}{\partial\theta_{1n}}\\ \vdots & \ddots & \vdots\\ \frac{\partial J(\theta)}{\partial\theta_{K1}} & \cdots & \frac{\partial J(\theta)}{\partial\theta_{Kn}}\end{pmatrix}=\sum_{i=1}^{m}[-\frac{\partial(\theta_k^T x_i)}{\partial\theta}+\frac{\partial\ln(\sum_{j=1}^{K}e^{\theta_j^T x_i})}{\partial\theta}] \tag{10}$$

引入符号函数 $sign_i^k=\begin{Bmatrix}1 & y_i=k\\ 0 & y_i\neq k\end{Bmatrix}$

则 $\theta_k^T x_i=sign_i^1\theta_1^T x_i+sign_i^2\theta_2^T x_i+\ldots+sign_i^K\theta_K^T x_i$

(11)

则

$$\frac{\partial(\theta_k^T x_i)}{\partial\theta}=\begin{bmatrix}\frac{\partial(\theta_k^T x_i)}{\partial\theta_{10}} & \cdots & \frac{\partial(\theta_k^T x_i)}{\partial\theta_{1n}}\\ \vdots & & \vdots\\ \frac{\partial(\theta_k^T x_i)}{\partial\theta_{K0}} & \cdots & \frac{\partial(\theta_k^T x_i)}{\partial\theta_{Kn}}\end{bmatrix}$$

$$=\begin{bmatrix}sign_i^1 & sign_i^1x_i^{(1)} & sign_i^1x_i^{(2)}\cdots sign_i^1x_i^{(n)}\\ sign_i^2 & sign_i^2x_i^{(1)} & sign_i^2x_i^{(2)}\cdots sign_i^2x_i^{(n)}\\ & \vdots & \\ sign_i^K & sign_i^Kx_i^{(1)} & sign_i^Kx_i^{(2)}\cdots sign_i^Kx_i^{(n)}\end{bmatrix}= \tag{12}$$

$$\begin{bmatrix}sign_i^1\\ sign_i^2\\ \vdots\\ sign_i^K\end{bmatrix}\begin{bmatrix}1 & x_i^{(1)} & x_i^{(2)}\cdots x_i^{(n)}\end{bmatrix}=\begin{bmatrix}sign_i^1\\ sign_i^2\\ \vdots\\ sign_i^K\end{bmatrix}x_i^T$$

$$\frac{\partial \ln(\sum_{j=1}^{K} e^{\theta_j^T x_i})}{\partial \theta} = \frac{1}{\sum_{j=1}^{K} e^{\theta_j^T x_i}} \begin{bmatrix} e^{\theta_1^T x_i} & e^{\theta_1^T x_i} x_i^{(1)} & \cdots & e^{\theta_1^T x_i} x_i^{(n)} \\ e^{\theta_2^T x_i} & e^{\theta_2^T x_i} x_i^{(1)} & \cdots & e^{\theta_2^T x_i} x_i^{(n)} \\ & & \vdots & \\ e^{\theta_K^T x_i} & e^{\theta_K^T x_i} x_i^{(1)} & \cdots & e^{\theta_K^T x_i} x_i^{(n)} \end{bmatrix} \tag{13}$$

$$= \frac{1}{\sum_{j=1}^{K} e^{\theta_j^T x_i}} \begin{bmatrix} e^{\theta_1^T x_i} \\ e^{\theta_2^T x_i} \\ \vdots \\ e^{\theta_K^T x_i} \end{bmatrix} x_i^T$$

所以

$$\nabla_\theta J(\theta) = \sum_{i=1}^{m} [\frac{1}{\sum_{j=1}^{K} e^{\theta_j^T x_i}} \begin{bmatrix} e^{\theta_1^T x_i} \\ e^{\theta_2^T x_i} \\ \vdots \\ e^{\theta_K^T x_i} \end{bmatrix} - \begin{bmatrix} sign_i^1 \\ sign_i^2 \\ \vdots \\ sign_i^K \end{bmatrix}] x_i^T \tag{14}$$

采用随机梯度下降，在 m 个样本中，随机挑选一个做梯度下降，

则

$$\nabla_\theta J(\theta) = [\frac{1}{\sum_{j=1}^{K} e^{\theta_j^T x_i}} \begin{bmatrix} e^{\theta_1^T x_i} \\ e^{\theta_2^T x_i} \\ \vdots \\ e^{\theta_K^T x_i} \end{bmatrix} - \begin{bmatrix} sign_i^1 \\ sign_i^2 \\ \vdots \\ sign_i^K \end{bmatrix}] x_i^T \tag{15}$$

对梯度加入权重：

$$\nabla_\theta J(\theta) = [\frac{1}{\sum_{j=1}^{K} e^{\theta_j^T x_i}} \begin{bmatrix} e^{\theta_1^T x_i} \\ e^{\theta_2^T x_i} \\ \vdots \\ e^{\theta_K^T x_i} \end{bmatrix} - \begin{bmatrix} sign_i^1 \\ sign_i^2 \\ \vdots \\ sign_i^K \end{bmatrix}] x_i^T + \lambda \theta_j \tag{16}$$

$\{(x_i, y_i) \mid i = 1,2,3,...m\}$ $\theta = \theta - \alpha * \nabla_\theta J(\theta)$

2 算法

1. 训练样本集 D 的 n 个样本$\{(x_i, y_i) \mid i = 1, 2, 3, \ldots m\}$

2. 随机选取一个进行梯度下降

3. 迭代更新梯度$\theta = \theta - \alpha * \nabla_{\theta} J(\theta)$

4. 输出迭代后的系数矩阵