2_广义线性模型

2_2_逻辑斯谛回归

2_2_1_sigmoid函数与二项逻辑斯谛回归模型

sigmoid函数：

$$sigmoid\left(z\right)=\sigma\left(z\right)=\frac{1}{1+e^{-z}}$$

其中，$z\in\mathbb{R}$，$sigoid\left(z\right)\in\left(0,1\right)$。

sigmoid函数的导数：

$$\sigma'\left(z\right)=\sigma\left(z\right)\left(1-\sigma\left(z\right)\right)$$

二项逻辑斯谛回归模型是如下的条件概率分布：

$$\begin{aligned}P\left(y=1|\mathbf{x}\right)&=\sigma\left(\mathbf{w}\cdot\mathbf{x}+b\right)\\&=\frac{1}{1+\exp\left(-\left(\mathbf{w}\cdot\mathbf{x}+b\right)\right)}\\&=\frac{\exp\left(\mathbf{w}\cdot\mathbf{x}+b\right)}{1+\exp\left(\mathbf{w}\cdot\mathbf{x}+b\right)}\\P\left(y=0|\mathbf{x}\right)&=1-\sigma\left(\mathbf{w}\cdot\mathbf{x}+b\right)\\&=\frac{1}{1+\exp\left(\mathbf{w}\cdot\mathbf{x}+b\right)}\end{aligned}$$

其中，$\mathbf{x}\in\mathbb{R}^n$，$y\in\{0,1\}$，$\mathbf{w}\in\mathbb{R}^n$是权值向量，$b\in\mathbb{R}$是偏置，$\mathbf{w}\cdot\mathbf{x}$为向量内积。

可将权值权值向量和特征向量加以扩充，即增广权值向量$\hat{\mathbf{w}}=\left(w^{(1)},w^{(2)},\cdots,w^{(n)},b\right)^\top$，增广特征向量$\hat{\mathbf{x}}=\left(x^{(1)},x^{(2)},\cdots,x^{(n)},1\right)^\top$，则逻辑斯谛回归模型：

$$\begin{aligned}P\left(y=1|\hat{\mathbf{x}}\right)&=\frac{\exp\left(\hat{\mathbf{w}}\cdot\hat{\mathbf{x}}\right)}{1+\exp\left(\hat{\mathbf{w}}\cdot\hat{\mathbf{x}}\right)}\\P\left(y=0|\hat{\mathbf{x}}\right)&=\frac{1}{1+\exp\left(\hat{\mathbf{w}}\cdot\hat{\mathbf{x}}\right)}\end{aligned}$$

2_2_2_二项逻辑斯谛回归参数学习——最大似然估计

给定训练数据集

$$D=\left\{\left(\hat{\mathbf{x}}_1,y_1\right),\left(\hat{\mathbf{x}}_2,y_2\right),\cdots,\left(\hat{\mathbf{x}}_N,y_N\right)\right\}$$

其中，$\hat{\mathbf{x}}_i\in\mathbb{R}^{n+1},y_i\in\left\{0,1\right\},i=1,2,\cdots,N$。

设

$$P\left(y=1|\hat{\mathbf{x}}\right)=\sigma\left(\hat{\mathbf{w}}\cdot\hat{\mathbf{x}}\right), \quad P\left(y=0|\hat{\mathbf{x}}\right)=1-\sigma\left(\hat{\mathbf{w}}\cdot\hat{\mathbf{x}}\right)$$

似然函数

$$\begin{aligned} L\left(\hat{\mathbf{w}}\right) &= \prod_{i=1}^{N} P\left(y_i|\hat{\mathbf{x}}_i\right) \\ &= \prod_{i=1}^{N} \left[\sigma\left(\hat{\mathbf{w}}\cdot\hat{\mathbf{x}}_i\right)\right]^{y_i}\left[1-\sigma\left(\hat{\mathbf{w}}\cdot\hat{\mathbf{x}}_i\right)\right]^{1-y_i} \end{aligned}$$

对数似然函数

$$\begin{aligned} l\left(\hat{\mathbf{w}}\right) &= \log L\left(\hat{\mathbf{w}}\right) \\ &= \sum_{i=1}^{N}\left[y_i\log\sigma\left(\hat{\mathbf{w}}\cdot\hat{\mathbf{x}}_i\right)+\left(1-y_i\right)\log\left(1-\sigma\left(\hat{\mathbf{w}}\cdot\hat{\mathbf{x}}_i\right)\right)\right] \end{aligned}$$

最大似然估计

$$\hat{\mathbf{w}}^{*}=\arg\max_{\hat{\mathbf{w}}} l\left(\hat{\mathbf{w}}\right)$$

令$\hat{y}_i=\sigma\left(\hat{\mathbf{w}}\cdot\hat{\mathbf{x}}_i\right)$，则对数似然函数$l\left(\hat{\mathbf{w}}\right)$关于$\hat{\mathbf{w}}$的偏导数

$$\begin{aligned} \frac{\partial l\left(\hat{\mathbf{w}}\right)}{\partial\hat{\mathbf{w}}} &= \sum_{i=1}^{N}\left(y_i\frac{\hat{y}_i\left(1-\hat{y}_i\right)}{\hat{y}_i}\hat{\mathbf{x}}_i+\left(1-y_i\right)\frac{\hat{y}_i\left(1-\hat{y}_i\right)}{1-\hat{y}_i}\hat{\mathbf{x}}_i\right) \\ &= \sum_{i=1}^{N}\left(y_i\left(1-\hat{y}_i\right)\hat{\mathbf{x}}_i+\left(1-y_i\right)\hat{y}_i\hat{\mathbf{x}}_i\right) \\ &= \sum_{i=1}^{N}\hat{\mathbf{x}}_i\left(y_i-\hat{y}_i\right) \end{aligned}$$

采用梯度上升法，初始化$\hat{\mathbf{w}}=\mathbf{0}$，进行迭代

$$\hat{\mathbf{w}}_{t+1}\leftarrow\hat{\mathbf{w}}_t+\alpha\sum_{i=1}^{N}\hat{\mathbf{x}}_i\left(y_i-\hat{y}_i^{\hat{\mathbf{w}}_t}\right)$$

其中，α是学习率，$\hat{y}_i^{\hat{\mathbf{w}}_t}$是当参数$\hat{\mathbf{w}}_t$时模型的输出。

2_2_3_softmax函数与多项逻辑斯谛回归模型

对于K个标量$x_1,x_2,\ldots,x_K$,softmax函数：

$$z_k=softmax\left(x_k\right)=\frac{\exp(x_k)}{\sum_{i=1}^{K}\exp(x_i)}$$

其中，$z_k\in[0,1]$，且$\forall k, \sum_{i=1}^{K}z_k=1$。

对于K维向量$\mathbf{x}=[x_1,x_2,\ldots,x_K]$，softmax函数：

$$\mathbf{z}=softmax\left(\mathbf{x}\right)=\frac{\exp(\mathbf{x})}{\sum_{i=1}^{K}\exp(x_i)}=\frac{\exp(\mathbf{x})}{\mathbf{1}_K^{\top}\exp(\mathbf{x})}$$

其中，$\mathbf{1}_K=[1,\ldots,1]_{K\times1}$是$K$维的全1向量。

K维向量$\mathbf{x}$的softmax函数的导数：

$$\frac{\partial softmax\left(\mathbf{x}\right)}{\partial \mathbf{x}} = diag\left(softmax\left(\mathbf{x}\right)\right) - softmax\left(\mathbf{x}\right) softmax\left(\mathbf{x}\right) \top$$

多项逻辑斯谛回归模型是如下的条件概率分布：

$$P\left(y=k|\hat{\mathbf{x}}\right) = \frac{\exp\left(\hat{\mathbf{w}}_k \cdot \hat{\mathbf{x}}\right)}{\sum_{i=1}^{K} \exp\left(\hat{\mathbf{w}}_i \cdot \hat{\mathbf{x}}_i\right)}$$

其中，$\hat{\mathbf{x}} \in \mathbb{R}^{n+1}$是特征增广向量，$y \in \{1, 2, \ldots, K\}$，$\hat{\mathbf{w}} \in \mathbb{R}^{n}$是权值增广向量，$\hat{\mathbf{w}} \cdot \hat{\mathbf{x}}$为向量内积。

多项式逻辑回归模型的向量表示

$$\hat{\mathbf{y}} = softmax\left(W^{\top}\hat{\mathbf{x}}\right) = \frac{\exp\left(W^{\top}\hat{\mathbf{x}}\right)}{\mathbf{1}_K^{\top}\exp\left(W^{\top}\hat{\mathbf{x}}\right)}$$

其中，$W = \left[\hat{\mathbf{x}}_1, \ldots, \hat{\mathbf{w}}_K\right]$是由$K$个类别权重向量组成的权重矩阵，$\mathbf{1}$是全1向量，$\hat{\mathbf{y}} \in \mathbb{R}^{K}$是所有K个类别的预测输出向量，第$k$维的值是第$k$类别的输出。

2_2_4_多项逻辑斯谛回归参数学习

给定训练数据集

$$D = \left\{\left(\hat{\mathbf{x}}_1, y_1\right), \left(\hat{\mathbf{x}}_2, y_2\right), \cdots, \left(\hat{\mathbf{x}}_N, y_N\right)\right\}$$

其中，$\hat{\mathbf{x}}_i \in \mathbb{R}^{n+1}, y_i \in \left\{1, 2\ ,\ \ldots, K\right\}, i = 1, 2, \cdots, N$。

对标签y使用K维的one-hot向量$\mathbf{y} \in \{0, 1\}^K$表示。对于类别$c$，其向量表示为

$$\mathbf{y} = \left[I\left(k=1\right), I\left(k=2\right), \ldots, I\left(k=K\right)\right] \top$$

其中，$I\left(\cdot\right)$是指示函数。

使用交叉熵损失函数，softmax回归模型的风险函数

$$\mathcal{L}\left(W\right) = -\frac{1}{N}\sum_{i=1}^{N}\mathbf{y}_i \cdot \log\hat{\mathbf{y}}_i = -\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K}\mathbf{y}_{ik}\log\hat{\mathbf{y}}_{ik}$$

其中，$\mathbf{y}_{ik}$是第i个标签one-hot向量表示的第k个维度元素值。

风险函数$\mathcal{L}\left(W\right)$关于权值矩阵$W$的梯度为

$$\frac{\partial \mathcal{L}\left(W\right)}{\partial W} = -\frac{1}{N}\sum_{i=1}^{N}\hat{\mathbf{x}}_i\left(\mathbf{y}_i - \hat{\mathbf{y}}_i\right)^{\top}$$

采用梯度下降法，初始化$\mathbf{W} = \mathbf{0}$，进行迭代

$$\mathbf{W}_{t+1} \leftarrow \mathbf{W}_t + \alpha\left(\frac{1}{N}\sum_{i=1}^{N}\hat{\mathbf{x}}_i\left(y_i - \hat{y}_i^{\mathbf{W}_t}\right)^{\top}\right)$$

其中，α是学习率，$\hat{y}_i^{\mathbf{W}_t}$是当参数$\mathbf{w}_t$时模型的输出。

2_2_5_逻辑斯谛回归模型应用

In [2]:
```
import matplotlib.pyplot as plt
import numpy as np
from sklearn import datasets, linear_model
from sklearn import model_selection

def load_data():
    iris=datasets.load_iris()
    X_train=iris.data
    y_train=iris.target
    return model_selection.train_test_split(X_train, y_train,test_size=0.25,ran
dom_state=0,stratify=y_train)

def test_LogisticRegression(*data):
    X_train,X_test,y_train,y_test=data
    regr = linear_model.LogisticRegression()
    regr.fit(X_train, y_train)
    print('Coefficients:%s, intercept %s'%(regr.coef_,regr.intercept_))
    print('Score: %.2f' % regr.score(X_test, y_test))

def test_LogisticRegression_multinomial(*data):
    X_train,X_test,y_train,y_test=data
    regr = linear_model.LogisticRegression(multi_class='multinomial',solver='lb
fgs')
    regr.fit(X_train, y_train)
    print('Coefficients:%s, intercept %s'%(regr.coef_,regr.intercept_))
    print('Score: %.2f' % regr.score(X_test, y_test))

def test_LogisticRegression_C(*data):
    X_train,X_test,y_train,y_test=data
    Cs=np.logspace(-2,4,num=100)
    scores=[]
    for C in Cs:
        regr = linear_model.LogisticRegression(C=C)
        regr.fit(X_train, y_train)
        scores.append(regr.score(X_test, y_test))

    fig=plt.figure()
    ax=fig.add_subplot(1,1,1)
    ax.plot(Cs,scores)
    ax.set_xlabel(r"C")
    ax.set_ylabel(r"score")
    ax.set_xscale('log')
    ax.set_title("LogisticRegression")
    plt.show()

if __name__=='__main__':
    X_train,X_test,y_train,y_test=load_data()
    test_LogisticRegression(X_train,X_test,y_train,y_test)
    test_LogisticRegression_multinomial(X_train,X_test,y_train,y_test)
    test_LogisticRegression_C(X_train,X_test,y_train,y_test)
```

```
Coefficients:[[ 0.38705175  1.35839989 -2.12059692 -0.95444452]
 [ 0.23787852 -1.36235758  0.5982662  -1.26506299]
 [-1.50915807 -1.29436243  2.14148142  2.29611791]], intercept [ 0.23950369
1.14559506 -1.0941717 ]
Score: 0.97
Coefficients:[[-0.38353372  0.86198376 -2.26983035 -0.97472053]
 [ 0.34380687 -0.37905055 -0.03125303 -0.86849329]
 [ 0.03972686 -0.48293322  2.30108338  1.84321382]], intercept [  8.75854734
2.49443506 -11.2529824 ]
Score: 1.00
```