2_广义线性模型

2_1_线性回归

2_1_1_多元线性回归模型

给定训练数据集

$$D=\{(\mathbf{x}_1,y_1),(\mathbf{x}_2,y_2),\cdots,(\mathbf{x}_i,y_i),\ldots,(\mathbf{x}_N,y_N)\}$$

其中，$\mathbf{x}_i\in\mathcal{X}\subseteq\mathbb{R}^n, y_i\in\mathcal{Y}\subseteq\mathbb{R}$。

线性回归模型：

$$f(\mathbf{x})=\mathbf{w}\cdot\mathbf{x}+b=\sum_{i=1}^{n}w^{(i)}\cdot x^{(i)}+b$$

其中，$\mathbf{x}\in\mathcal{X}\subseteq\mathbb{R}^n$是输入，$\mathbf{w}=\left(w^{(1)},w^{(2)},\ldots,w^{(n)}\right)^\top\in\mathbb{R}^n$和$b\in\mathbb{R}$是参数，$\mathbf{w}$称为权值向量，$b$称为偏置，$\mathbf{w}\cdot\mathbf{x}$为$\mathbf{w}$和$\mathbf{x}$的内积。

令

$$\hat{\mathbf{w}}=(\mathbf{w},b)^\top$$
$$\hat{\mathbf{x}}=(\mathbf{x},1)^\top$$

则多元线性回归模型可简化为

$$f\left(\hat{\mathbf{x}}\right)=\hat{\mathbf{w}}\cdot\hat{\mathbf{x}}$$

其中，$\hat{\mathbf{x}}$为增广特征向量，$\hat{\mathbf{w}}$为增广权重。

2_1_2_多元线性回归参数学习——经验风险最小化与结构风险最小化

损失函数：平方损失损失函数

$$L(y,f(\mathbf{x}))=(y-f(\mathbf{x}))^2$$

经验风险

$$R_{emp}(f)=\frac{1}{N}\sum_{i=1}^{N}L(y_i,f(\mathbf{x}_i))$$

模型参数最优解：

$$\hat{\mathbf{w}}^*=\arg\min_{\hat{\mathbf{w}}}\sum_{i=1}^{N}\left(y_i-f\left(\hat{\mathbf{x}}_i\right)\right)^2$$

基于均方误差最小化来进行模型求解的方法称为“最小二乘法”（least square method）。

等价的，模型参数最优解：

$$\hat{\mathbf{w}}^* = \underset{\hat{\mathbf{w}}}{\arg\min} \left(\mathbf{y} - \mathbf{X}\hat{\mathbf{w}}\right)^\top \left(\mathbf{y} - \mathbf{X}\hat{\mathbf{w}}\right)$$

其中，

$$\mathbf{X} = \begin{pmatrix} \mathbf{x}_1^\top & 1 \\ \mathbf{x}_2^\top & 1 \\ \vdots & \vdots \\ \mathbf{x}_N^\top & 1 \end{pmatrix}$$

$$\mathbf{y} = (y_1, y_2, \ldots, y_N)^\top$$

令$E_{\hat{\mathbf{w}}} = \left(\mathbf{y} - \mathbf{X}\hat{\mathbf{w}}\right)^\top \left(\mathbf{y} - \mathbf{X}\hat{\mathbf{w}}\right)$，对$\hat{\mathbf{w}}$求偏导，得

$$\frac{\partial E_{\hat{\mathbf{w}}}}{\partial \hat{\mathbf{w}}} = 2\mathbf{X}^\top \left(\mathbf{X}\hat{\mathbf{w}} - \mathbf{y}\right)$$

当$\mathbf{X}^\top\mathbf{X}$为满秩矩阵或正定矩阵时，令上式为零可得最优闭式解

$$\hat{\mathbf{w}}^* = \left(\mathbf{X}^\top\mathbf{X}\right)^{-1}\mathbf{X}^\top\mathbf{y}$$

当上述条件不满足时，可使用主成分分析（PCA）等方法消除特征间的线性相关性，再使用最小二乘法求解。
或者通过梯度下降法，初始化$\hat{\mathbf{w}}_0 = \mathbf{0}$，进行迭代

$$\hat{\mathbf{w}} \leftarrow \hat{\mathbf{w}} + \alpha\mathbf{X}^\top \left(\mathbf{X}\hat{\mathbf{w}} - \mathbf{y}\right)$$

其中，α是学习率。

岭回归（Ridge Regression）正则化项：

$$\alpha\|\mathbf{w}\|^2, \alpha \geq 0.$$

套索回归（Lasso Regression）正则化项：

$$\alpha\|\mathbf{w}\|_1, \alpha \geq 0.$$

弹性网络回归（Elastic Net）正则化项：

$$\alpha\rho\|\mathbf{w}\|_1 + \frac{\alpha(1-\rho)}{2}\|\mathbf{w}\|^2, \alpha \geq 0, 1 \geq \rho \geq 0.$$

2_1_3_多元线性回归参数学习——最大似然估计

设标记y为服从均值为$f\left(\hat{\mathbf{x}}\right) = \hat{\mathbf{w}} \cdot \hat{\mathbf{x}}$，方差为$\sigma^2$的高斯分布

$$\begin{aligned} p\left(y|\hat{\mathbf{x}}, \hat{\mathbf{w}}, \sigma\right) &= \mathcal{N}\left(y|\hat{\mathbf{w}} \cdot \hat{\mathbf{x}}, \sigma^2\right) \\ &= \frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{\left(y - \hat{\mathbf{w}} \cdot \hat{\mathbf{x}}\right)^2}{2\sigma^2}\right) \end{aligned}$$

参数$\hat{\mathbf{w}}$在训练集D上的似然函数（Likelihood）

$$\begin{aligned} p(\mathbf{y}|X, \hat{\mathbf{w}}, \sigma) &= \prod_{i=1}^{N} p\left(y_i|\mathbf{x}_i, \hat{\mathbf{w}}, \sigma\right) \\ &= \prod_{i=1}^{N} \mathcal{N}\left(y_i|\hat{\mathbf{w}} \cdot \hat{\mathbf{x}}_i, \sigma^2\right) \end{aligned}$$

参数$\hat{\mathbf{w}}$在训练集D上的对数似然函数（Log Likelihood）

$$\log p(\mathbf{y}|X, \hat{\mathbf{w}}, \sigma) = \sum_{i=1}^{N} \log \mathcal{N}\left(y_i|\hat{\mathbf{w}} \cdot \hat{\mathbf{x}}_i, \sigma^2\right)$$

最大似然估计

$$\hat{\mathbf{w}}^* = \arg\max_{\hat{\mathbf{w}}} p(\mathbf{y}|X, \hat{\mathbf{w}}, \sigma)$$

等价的

$$\hat{\mathbf{w}}^* = \arg\max_{\hat{\mathbf{w}}} \log p(\mathbf{y}|X, \hat{\mathbf{w}}, \sigma)$$

等价的

$$\hat{\mathbf{w}}^* = \arg\min_{\hat{\mathbf{w}}} -\log p(\mathbf{y}|X, \hat{\mathbf{w}}, \sigma)$$

令$\frac{\partial \log p(\mathbf{y}|X,\hat{\mathbf{w}},\sigma)}{\partial \hat{\mathbf{w}}} = 0$，得

$$\mathbf{w}^{ML} = \left(\mathbf{X}^{\top}\mathbf{X}\right)^{-1}\mathbf{X}^{\top}\mathbf{y}$$

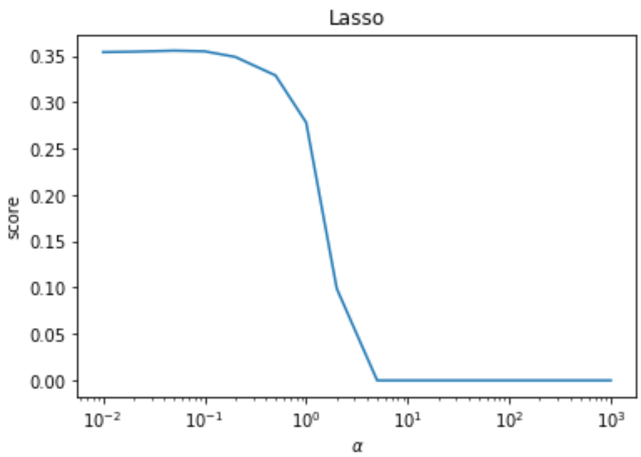
2_1_4_多元线性回归模型应用

In [3]:
```
import matplotlib.pyplot as plt
import numpy as np
from sklearn import datasets, linear_model,model_selection

def load_data():
    diabetes = datasets.load_diabetes()
    return model_selection.train_test_split(diabetes.data,diabetes.target,test_
size=0.25,random_state=0)

def test_LinearRegression(*data):
    X_train,X_test,y_train,y_test=data
    regr = linear_model.LinearRegression()
    regr.fit(X_train, y_train)
    print('Coefficients:%s, intercept %.2f'%(regr.coef_,regr.intercept_))
    print("Residual sum of squares: %.2f"% np.mean((regr.predict(X_test) - y_te
st) ** 2))
    print('Score: %.2f' % regr.score(X_test, y_test))

if __name__=='__main__':
    X_train,X_test,y_train,y_test=load_data()
    test_LinearRegression(X_train,X_test,y_train,y_test)
```

```
Coefficients:[ -43.26774487 -208.67053951  593.39797213  302.89814903
-560.27689824
  261.47657106   -8.83343952  135.93715156  703.22658427   28.34844354], inter
cept 153.07
Residual sum of squares: 3180.20
Score: 0.36
```

In [12]:
```
import matplotlib.pyplot as plt
import numpy as np
from sklearn import datasets, linear_model,model_selection

def load_data():
    diabetes = datasets.load_diabetes()
    return model_selection.train_test_split(diabetes.data,diabetes.target,
        test_size=0.25,random_state=0)
def test_Lasso(*data):
    X_train,X_test,y_train,y_test=data
    regr = linear_model.Lasso()
    regr.fit(X_train, y_train)
    print('Coefficients:%s, intercept %.2f'%(regr.coef_,regr.intercept_))
    print("Residual sum of squares: %.2f"% np.mean((regr.predict(X_test) - y_te
st) ** 2))
    print('Score: %.2f' % regr.score(X_test, y_test))

def test_Lasso_alpha(*data):
    X_train,X_test,y_train,y_test=data
    alphas=[0.01,0.02,0.05,0.1,0.2,0.5,1,2,5,10,20,50,100,200,500,1000]
    scores=[]
    for i,alpha in enumerate(alphas):
        regr = linear_model.Lasso(alpha=alpha)
        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(alphas,scores)
    ax.set_xlabel(r"$\alpha$")
    ax.set_ylabel(r"score")
    ax.set_xscale('log')
    ax.set_title("Lasso")
    plt.show()

if __name__=='__main__':
    X_train,X_test,y_train,y_test=load_data()
    test_Lasso(X_train,X_test,y_train,y_test)
    test_Lasso_alpha(X_train,X_test,y_train,y_test)
```

```
Coefficients:[   0.           -0.          442.67992538    0.             0.
   0.          -0.            0.          330.76014648    0.          ], intercept
152.52
Residual sum of squares: 3583.42
Score: 0.28
```



In [11]:
```
import matplotlib.pyplot as plt
import numpy as np
from sklearn import datasets, linear_model,model_selection

def load_data():
    diabetes = datasets.load_diabetes()
    return model_selection.train_test_split(diabetes.data,diabetes.target,
        test_size=0.25,random_state=0)

def test_Ridge(*data):
    X_train,X_test,y_train,y_test=data
    regr = linear_model.Ridge()
    regr.fit(X_train, y_train)
    print('Coefficients:%s, intercept %.2f'%(regr.coef_,regr.intercept_))
    print("Residual sum of squares: %.2f"% np.mean((regr.predict(X_test) - y_te
st) ** 2))
    print('Score: %.2f' % regr.score(X_test, y_test))

def test_Ridge_alpha(*data):
    X_train,X_test,y_train,y_test=data
    alphas=[0.01,0.02,0.05,0.1,0.2,0.5,1,2,5,10,20,50,100,200,500,1000]
    scores=[]
    for i,alpha in enumerate(alphas):
        regr = linear_model.Ridge(alpha=alpha)
        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(alphas,scores)
    ax.set_xlabel(r"$\alpha$")
    ax.set_ylabel(r"score")
    ax.set_xscale('log')
    ax.set_title("Ridge")
    plt.show()

if __name__=='__main__':
    X_train,X_test,y_train,y_test=load_data()
    test_Ridge(X_train,X_test,y_train,y_test)
    test_Ridge_alpha(X_train,X_test,y_train,y_test)
```

```
Coefficients:[  21.19927911  -60.47711393  302.87575204  179.41206395
8.90911449
  -28.8080548  -149.30722541  112.67185758  250.53760873   99.57749017], inter
cept 152.45
Residual sum of squares: 3192.33
Score: 0.36
```