2.线性模型算法

# 1 实验目的

1.了解sklearn.linear\_model模块中的线性回归算法和逻辑回归算法的原理；

2.了解sklearn.linear\_model模块中正则化线性模型：岭回归、Lasso回归算法；

2.了解如何使用sklearn.linear\_model模块中常用算法完成机器学习任务。

# 2 实验要求

本次实验后，要求同学们能：

1.掌握最线性回归、逻辑回归算法的实现原理；

2.学会使用Scikit-learn实现线性模型中的常用算法

2.了解sklearn.linear\_model模块中各算法的不同特点，了解算法中各个超参数的含义。

# 3 实验内容

## 3.1 线性模型简介

线性模型就是假设样本数据符合一种“线性”的态势，用一个线性的假设函数去拟合数据，通过最小化损失函数以及正则化手段来优化假设函数，最终得到样本数据的一个较为理想的线性拟合函数，可以用于对新数据进行预测。

scikit-learn库中sklearn.linear\_model模块提供了不同的线性模型算法，下面介绍其中几个常见算法，并给出简单的使用示例。

## 3.2 普通线性模型LinearRegression

### 3.2.1原理简介

LinearRegression模型的假设函数为：

其中Y是预测结果；X=（x1，x2，…，xn）是样本数据，x1，x2，…，xn是样本在不同维度特征上的取值；ω=（ω1，ω2，…，ωn）是权值系数；b是偏差值。

一般用均方误差作为普通最小二乘法的损失函数：

普通线性回归算法就是先假定初始的权值系数ω，计算损失函数loss，然后通过各种方法，比如梯度下降法，不断优化并更新权值系数ω，让损失函数loss达到尽可能小的一个值，从而实现对样本数据的拟合。

### 3.2.2 实验

（1）导入依赖并加载数据

用sklearn.datasets中的make\_regression()方法得到一组带有噪音的线性回归数据作为测试数据，其重要参数及含义如下

**n\_samples：设置数据的样本数。**

**n\_features：设置数据中每个样本的特征数。**

**n\_targets：设置回归问题的目标数量。**

**n\_informative：设置信息特征的数量，即用于构建回归数据的线性模型的权值参数个数。**

**coef：默认为False，设置为True时，会输出线性模型的权值参数。**

**bias：设置基本线性模型中的偏差量。**

**noise：设置数据中的符合高斯分布的噪声大小。**

**random\_state：用于固定噪声的随机性，即，random\_state取相同值且其他参数也相同时，**make\_regression()产生的线性回归数据一样。

按如下参数生成一个用于展示回归算法的线性模型，并划分训练集和测试集。

# 用于可视化

import matplotlib.pyplot as plt

# 导入线性模型中的线性回归算法

from sklearn.linear\_model import LinearRegression

# 导入数据集

from sklearn.datasets import make\_regression

# 用于划分数据集

from sklearn.model\_selection import train\_test\_split

# 载入数据

data, target = make\_regression(n\_samples=100, n\_targets=1, n\_features=10, effective\_rank=5, bias=10, noise=3)

# 划分训练集和测试集

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(

data,

target,

test\_size=0.2,

random\_state=1

)

（2）创建算法模型实例

用sklearn.linear\_model中的LinearRegression类可以创建线性回归算法模型实例，其构造器重要参数及其含义如下：

**fit\_intercept：默认为True，表示是否计算截距，如果设置为False，则拟合出过原点的曲线。**

**normalize：默认为False，表示是在回归算法之前，否**通过减去均值并除以l2-范数来对回归变量X进行归一化。

**copy\_X：默认为True，表示回归算法前是否复制X，如果未False，则X可能会被覆盖**

**注：**LinearRegression(\*,fit\_intercept=True,normalize=False,copy\_X=True, n\_jobs=None)中的“\*”的作用是限制关键字参数的名字，只接受\*后面的命名关键字参数。例如LinearRegression(fit\_intercept=True) 可正确创建算法模型，而写成LinearRegression(fit=True)会报错，之后再出现时不再赘述。

本实验直接用默认值创建算法模型，代码如下：

**model1 = LinearRegression()**

**（3）训练并测试**

**该算法实例提供以下方法：**

**fit(X, y[, sample\_weight])：使用X作为训练集、y作为训练集标签来拟合线性模型。sample\_weight默认为None，表示样本的权重（可选）。函数会返回一个估计器实例；**

**get\_params()：获取此估计器的参数；**

**set\_params(\*\*params)：主动设置此估计器的参数；**

**predict(X)：使用线性模型对数据集X进行预测；**

**score(X, y[, sample\_weight])：使用X作为测试集、y作为测试集标签对模型进行评分，Scikit-learn库中的回归模型的评分计算公式为：，评分越接近1说明训练出的模型效果越好。**

**调用fit()函数进行训练，在测试集上进行预测，然后打印出评分，将预测结果可视化，代码如下**

**model1.fit(X\_train, Y\_train)**

**# 预测**

**result1 = model1.predict(X\_test)**

**# 评分**

**print("\n训练集评分", model1.score(X\_train, Y\_train))**

**print("测试集评分", model1.score(X\_test, Y\_test))**

**# 可视化**

**plt.plot(result1,"ro-",label="predict value")**

**plt.plot(Y\_test,"bo--",label="real value")**

**plt.title("LinearRegression")**

**plt.xlabel("index")**

**plt.ylabel("value")**

**plt.legend()**

**plt.show()**

**（4）检查训练出的参数**

**可以用parameter\_（参数名加下划线）的形式访问算法模型的成员变量，查看训练得到的参数。**

**coef\_：可直接访问训练出的估计权值（即X的系数w =（w1，…，wp））。**

**intercept\_：可直接访问训练出的截距。**

**打印出训练后的权值和截距。**

**print("\n估计权值为", model1.coef\_)**

**print("估计出的截距为", model1.intercept\_)**

### 3.2.3 实验结果

（1）普通线性模型评分



图1 普通线性模型评分

（2）预测结果可视化

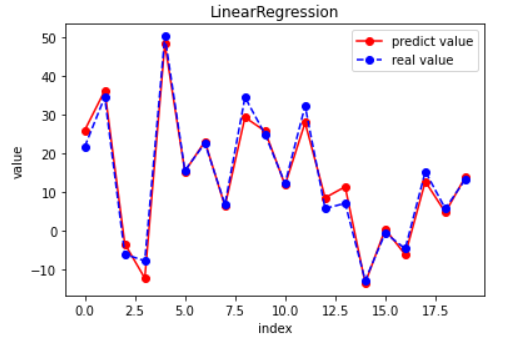


图2 预测结果可视化

（3）查看训练出的权值和截距

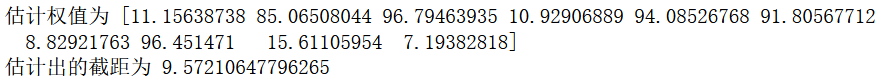


图3 训练出的权值与截距

## 3.3 正则线性模型

### 3.3.1 原理

普通线性模型在训练时，通过调整权值系数ω，让训练集上损失函数loss尽可能小，但是模型复杂时，很可能损失函数loss在测试集上仍然较大，泛化能力不强。这是因为没有对ω进行任何限制，造成了较大的结构性风险。

正则化就是对ω进行限制，符合奥卡姆剃刀原理：在所有可能选择的模型中，能够很好地解释已知数据，并且最简单的，才是最好的模型。

在线性模型中，各个权值系数都取值较小，或者特征数较少（多个权值系数为0），则认为是一个比较简单的模型。

L1正则化（又称L1范数、L1惩罚项）：

L2正则化（又称L2范数、L2惩罚项）：

岭回归Ridge就是在普通线性回归的损失函数中加上了一个对ω的L2正则化项：

岭回归的损失函数：

而套索回归Lasso是在普通线性回归的损失函数中加上了一个对ω的L1正则化项：

套索回归的损失化函数：

其中，α为正则化系数，可以控制正则化的强弱，即α越大，最终学习到的权值系数越小；α越小，最终学习到的权值系数越大。将损失函数L1或L2正则化之后，算法模型的学习过程等于是要同时让原本的损失函数和正则化项达到最小值。

值得注意的一点是，Lasso回归用了L1正则化，原本的损失函数和L1正则化项同时达到最小值很可能出现在坐标轴上（即容易出现为0的权值），从而在降低结构风险的同时，还能起到特征选择的作用。而Ridge回归用到L2正则化，模型学习到的权值一般都较小但不为0。

### 3.3.2 岭回归（Ridge）实验

用sklearn.linear\_model模块下的Ridge类可以创建Ridge回归算法模型实例，其构造器重要参数及含义如下：

**alpha：是L2正则化项前面的系数，用于设置正则化强度。**

**fit\_intercept、normalize、copy\_X：与普通线性模型含义相同。**

**tol：设置最大可忍受的误差，当误差小于tol时停止训练。**

**max\_iter：设置最大迭代次数，与tol同为算法训练停止的条件。**

**solver：选择岭回归的求解器，默认为“auto”，可以根据数据自动选择。**

**岭回归模型实例的主要成员变量和成员方法与普通线性模型类似，不再赘述。**

**下面创建了一个默认参数的岭回归模型实例，并在用**make\_regression构建的线性回归模型上进行测试。

from sklearn.linear\_model import Ridge

from sklearn.model\_selection import train\_test\_split

from sklearn.datasets import make\_regression

# 加载数据

data, target = make\_regression(n\_samples=100, n\_targets=1, n\_features=10, n\_informative=5,

bias=10, noise=3, random\_state=1)

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(

data,

target,

test\_size=0.2,

random\_state=1

)

# 创建并训练模型

model2 = Ridge(alpha=1)

model2.fit(X\_train, Y\_train)

# 验证模型

print("训练集评分", model2.score(X\_train, Y\_train))

print("测试集评分", model2.score(X\_test, Y\_test))

print("\n估计权值为", model2.coef\_)

print("估计出的截距为", model2.intercept\_)

### 3.3.3 套索回归（Lasso）实验

用sklearn.linear\_model模块的Lasso类可以创建Lasso回归算法模型实例，其构造器接受的主要参数及含义与岭回归相似，实例提供的主要成员变量与成员方法也与普通线性模型类似，所以不再赘述，

下面创建了一个默认参数的套索回归模型实例，在与上文岭回归实验中相同的数据上进行测试，比较两种正则化线性模型的差异。

from sklearn.linear\_model import Lasso

from sklearn.model\_selection import train\_test\_split

from sklearn.datasets import make\_regression

# 加载数据

data, target = make\_regression(n\_samples=100, n\_targets=1, n\_features=10, n\_informative=5,

bias=10, noise=3, random\_state=1)

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(

data,

target,

test\_size=0.2,

random\_state=1

)

# 创建并训练模型

model3 = Lasso(alpha=1)

model3.fit(X\_train, Y\_train)

# 模型验证

print("训练集评分", model3.score(X\_train, Y\_train))

print("测试集评分", model3.score(X\_test, Y\_test))

print("\n估计权值为", model3.coef\_)

print("估计出的截距为", model3.intercept\_)

### 3.3.4 实验结果

（1）岭回归实验结果

可以看到，相比普通线性回归模型训练出对的权值，加上了L1正则的岭回归模型训练得到的权值总体上更小。模型在测试集上的评分也跟搞，代表模型的泛化能力更强。

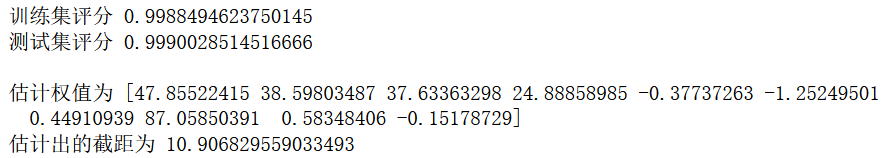


图4 岭回归实验结果

（2）套索回归实验结果

可以看到，相同的数据由套索回归训练出的模型有很多值为0的权值，这说明通过L2正则，不仅可以防止过拟合，还能起到特征筛选的作用。

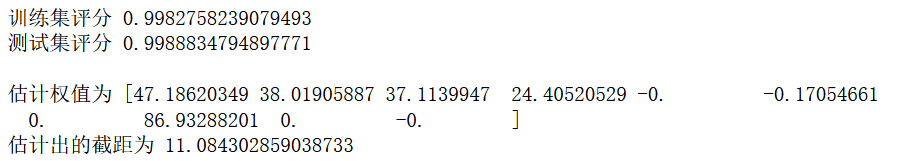


图5 套索回归实验结果

## 3.4 逻辑回归算法（LogisticRegression）

### 3.4.1原理

Logistic算法虽然叫作逻辑“回归”，但实际上用于解决分类问题。主要用于二分类问题，在多分类问题上容易欠拟合。逻辑回归实际上是在线性模型上“套”了一层sigmoid函数从而实现分类功能。它的假设函数如下：

损失函数为：

这个损失函数实际上是由预测准确的概率得来，由于概率不能用梯度下降优化，机器学习中常对概率取对数再进行优化，这就得到了这个看起来很怪的损失函数。

与其他线性模型一样，逻辑回归也是可以用L1或L2惩罚函数来正则化的，Scikit-learn中默认添加L2正则化。

### 3.4.2 实验

（1）导入依赖并加载数据

用Scikit-learn自带的鸢尾花数据集作为实验数据，它是经典且非常容易的多类分类数据集。

鸢尾花数据集一共含有150个样本（每一类各50个样本），每一行对应一朵花，列代表每朵花的四个测量数据，分别是：花瓣的长度，宽度，花萼的长度、宽度。最后一项表示鸢尾花的类别，共三类，分别是山鸢尾（0表示）、色鸢尾（1表示）、维吉尼亚鸢尾（2表示）

from sklearn.linear\_model import LogisticRegression

from sklearn.datasets import load\_iris

from sklearn.model\_selection import train\_test\_split

iris\_data = load\_iris()

data = iris\_data.data

target = iris\_data.target

# 划分数据集

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(

data,

target,

test\_size=0.1,

random\_state=1

)

（2）创建逻辑回归算法模型实例。

用构造函数sklearn.linear\_model模块下的LogisticRegression类可以创建一个逻辑回归算法模型实例，其构造器的重要参数与含义如下（与前文线性回归模型中重复的参数不再说明）：

**penalty：用于指定正则化范数，默认为“L2”，代表使用L2正则。也可以选择“L1”表示使用L1正则，或者“None”表示不进行正则化**

**C：设置正则化强度的倒数，必须为正浮点数。**

**class\_weight：给每个分类类别人为第设置不同的重要性权重，默认为“None”表示不设置权重；也可以设置为“balanced”将根据类别样本数设置权重，样本数多的，权值高；还可以通过一个字典类型来自定义。**

下面创建一个最大迭代次数为1000的逻辑回归算法模型实例进行实验。

model4 = LogisticRegression(max\_iter=1000)

（3）训练并检验模型

除了fit()、predict()、get\_params()、set\_params()、score()等方法，逻辑回归算法还提供以下方法：

**predict\_proba(X)：**返回预测X中各个样本属于各个类别的概率。

**predict\_log\_proba(X)：**返回预测X中各个样本属于各个类别的概率的对数。

调用fit()方法进行训练，用训练好的模型对测试数据进行分类，最后打印出算法模型的评分。实验代码如下：

model4.fit(X\_train, Y\_train)

print("测试数据分类结果", model4.predict(X\_test))

print("测试集实际值", Y\_test )

print("\n训练集评分", model4.score(X\_train, Y\_train))

print("测试集评分", model4.score(X\_test, Y\_test))

（3）查看逻辑回归算法模型训练出的参数

代码如下：

print("\n分类函数的决策系数为", model4.coef\_)

print("分类函数的偏差为", model4.intercept\_)

print("\n实际迭代次数为", model4.n\_iter\_)

### 3.4.3 实验结果

（1）预测与评估

如图所示，用逻辑回归解决鸢尾花分类问题可以得到比较满意的准确度。

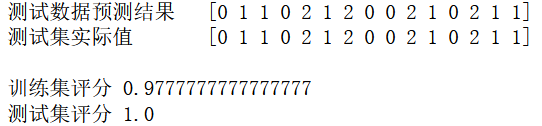


图6 模型预测与评估

（2）查看训练出的参数

由于本实验中鸢尾花数据是一个四个特征的三分类问题，所以逻辑回归算法拟合出了三条决策边界，它们的权值系数和偏差值如下图所示。

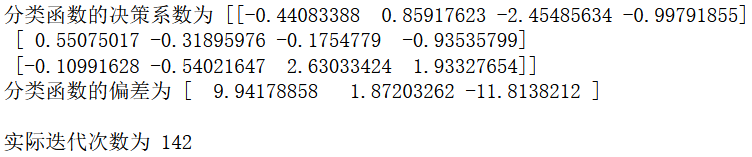


图7 模型训练出的权值系数和偏差值