day7

一、今日内容

a. 线性回归算法
b. 过拟合与欠拟合
c. 岭回归算法
d. 特征降维
e. Kmeans算法
f. 购物篮分析案例

二、复习

a. 特征预处理：处理数字类型的特征
    i. 包含内容： 归一化 标准化
b. 归一化：
    i. 概念： 通过转换函数，将数字类型的特征，转换为适合机器学习的特征
    ii. 转换公式：
        1. x‘ = （x - min） / (max - min)
        2. x" = x’ * (mx - mi) + mi
    iii. api：from sklearn.preprocessing import MinMaxScaler
        1. fit: 制定转换数据的标准
        2. transform： 使用制定的转换标准，对未知数据进行转换
c. 特征抽取
    i. 概念： 将任意类型的数据转换为适合机器学习的数字类型的特征
    ii. 原理：将文本中出现的特征 词的个数作为数字类的数据统计
    iii. 中文分词工具： jieba.cut
        1. pip install jieba
        2. jieba.cut
    iv. api：from sklearn.feature_extraction.text import CountVectorizer
        1. fit: 制定转换数据的标准
        2. transform:对数据进行转换处理 返回值 sparse矩阵
        3. get_feature_names():
        4. toarray()
d. 朴素贝叶斯算法
    i. 文章分类原理： 分别计算文章属于每一个类别的概率
    ii. 概率
        1. 条件概率：P(A|B)
        2. 联合概率：P(A,B) = P(A) *P(B)
    iii. 贝叶斯公式

iv. 拉普拉斯平滑系数

v. api:from sklearn.naive_bayes import MultinomialNB

1. fit
2. predict
3. score

三、线性回归算法

3.1 概念

线性回归： 特征值 （x） 与 目标值（y） 之间建立的某种函数关系。

demo: y = kx + b： 单因子线性回归。

3.2 线性回归通式

通用公式：$h(w) = w_1x_1 + w_2x_2 + w_3x_3 \ldots + b = w^Tx + b$

目标值　特征值　特征值　特征值

3.3 回归系数和损失的计算

真实关系：真实房子价格 = 0.02×中心区域的距离 + 0.04×城市一氧化氮浓度 + (-0.12×自住房平均房价) + 0.254×城镇犯罪率

x1 x2 x3 x4

线性回归关系： h(w) = w1x1 + w2x2 + w3x3 +w4x4

question：如何计算线性回归的回归系数。

正规方程：

概念：通过公式一步计算出模型损失最低的回归系数。

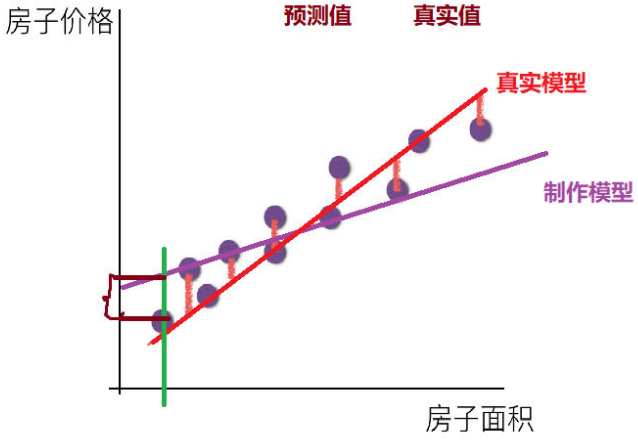
公式：

$$w = (X^T X)^{-1} X^T y$$

缺点： 当模型过于复杂，有可能出现无法计算出来损失的情况。

error $\sum_{i=1}^{m}(h_w(x_i) - y_i)^2$

预测值 真实值



梯度下降学习法

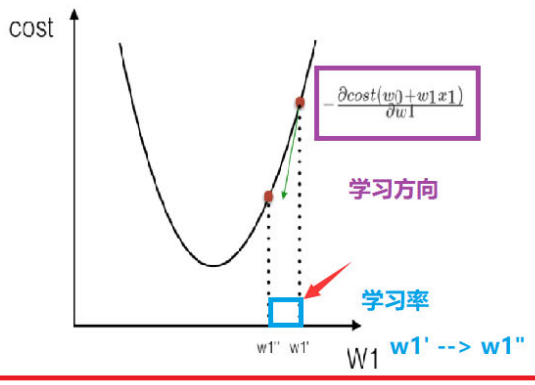
概念： 根据给出的学习率和学习方向，一步一步的找到损失最低点的回归系数

公式：

$$w1 := w1 - \alpha \frac{\partial cost(w0+w1x1)}{\partial w1}$$

学习率 学习方向

target: 找到模型损失最低的回归系数。



3.4 算法理解

波士顿房价预测：

真实关系：真实房子价格 = 0.02×中心区域的距离 + 0.04×城市一氧化氮浓度 + (-0.12×自住房平均房价) + 0.254×城镇犯罪率

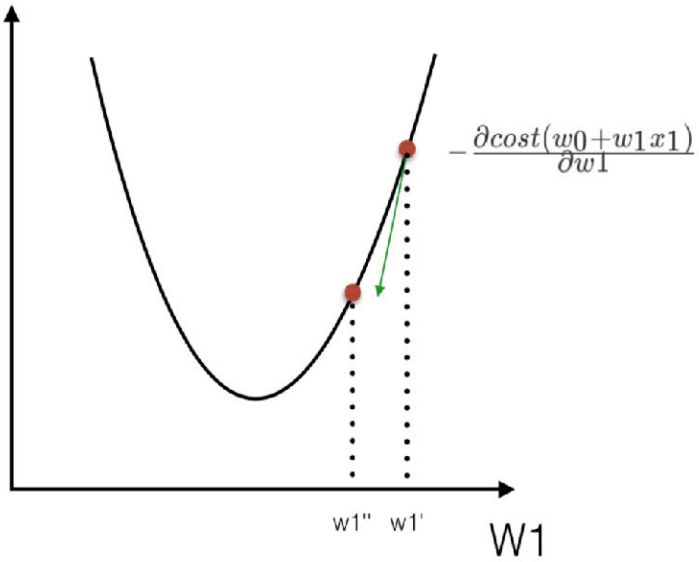
随机指定关系：预测房子价格 = 0.25×中心区域的距离 + 0.14×城市一氧化氮浓度 + 0.42×自住房平均房价 + 0.34×城镇犯罪率

随机值　随机值　随机值　随机值

通过指定的关系制作出模型 A ,与实际情况的模型 一定有一个损失

预测的模型的损失：

$$\sum_{i=1}^{m}(h_w(x_i)-y_i)^2$$



通过调整 w的值， 使得模型的损失越来越小。

调整w的值，是需要 （1）正规方程 （2) 梯度下降

3.5 梯度下降 api接口

```
# 导入梯度下降的api
from sklearn.linear_model import SGDRegressor
# 导入损失计算公式
from sklearn.metrics import mean_squared_error
"""
SGDRegressor 梯度下降学习法

__init__成员函数

函数原型:
__init__(self, fit_intercept=True learning_rate="invscaling", eta0=0.01)

函数参数:
    fit_intercept: bool类型,标识是否计算模型的偏置 y = kx + b
    learning_rate: str类型: 标识学习率的计算方式。一般情况下是 constant标识固定学习率
    eta0:  学习率

fit: 拟合训练集的数据,制作模型
predict: 根据数据，进行预测结果。

```

```
mean_squared_error函数

函数原型:
    def mean_squared_error(y_true, y_pred);

函数参数:
    y_true: 真实值结果 y_test
    y_Pred: 预测值的结果 测试集特征值经过模型预测出来的结果.

"""
```

3.6 波士顿房价预测

实例数量:	506
属性数量:	13 数值型或类别型，帮助预测的属性

:中位数（第14个属性）经常是学习目标

属性信息 (按顺序):	• CRIM 城镇人均犯罪率 • ZN 占地面积超过2.5万平方英尺的住宅用地比例 • INDUS 城镇非零售业务地区的比例 • CHAS 查尔斯河虚拟变量 (= 1 如果土地在河边；否则是0) • NOX 一氧化氮浓度（每1000万份） • RM 平均每居民房数 **特征值** • AGE 在1940年之前建成的所有者占用单位的比例 • DIS 与五个波士顿就业中心的加权距离 • RAD 辐射状公路的可达性指数 • TAX 每10,000美元的全额物业税率 • PTRATIO 城镇师生比例 • B 1000(Bk - 0.63)^2 其中 Bk 是城镇的黑人比例 • LSTAT 人口中地位较低人群的百分数 • MEDV 以1000美元计算的自有住房的中位数 **目标值**
缺失属性值:	无
创建者:	Harrison, D. and Rubinfeld, D.L.

这是UCI ML（欧文加利福尼亚大学 机器学习库）房价数据集的副本。 http://archive.ics.uci.edu/ml/datasets/Housing

该数据集是从位于卡内基梅隆大学维护的StatLib图书馆取得的。

```python
# 导入波士顿房价的数据
from sklearn.datasets import load_boston
# 导入归一化
from sklearn.preprocessing import MinMaxScaler
# 导入线性回归
from sklearn.linear_model import SGDRegressor
# 导入分割数据集
from sklearn.model_selection import train_test_split
# 导入模型的损失计算
```

```python
from sklearn.metrics import mean_squared_error

def main():
    """ 预测波士顿房价 """

    # 1.获取原始数据集
    lb = load_boston()

    # 2.确定特征值与目标值
    x = lb.data
    y = lb.target

    # 3. 分割数据集
    x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3)

    # 4. 归一化处理数据
    concert = MinMaxScaler()
    # 制定转换数据的标准
    concert.fit(x_train)
    x_train = concert.transform(x_train)
    x_test = concert.transform(x_test)

    # 实例化 梯度下降学习法
    estimate = SGDRegressor(fit_intercept=True, learning_rate="constant", eta0=0.01)

    # 拟合数据制作模型
    estimate.fit(x_train, y_train)

    # 线性回归的模型 y = kx + b   k,b变量
    coef = estimate.coef_  # coef_ 回归系数
    intercept = estimate.intercept_ # 偏置

    print("模型的回归系数: ", coef)
    print("模型的偏置: ", intercept)

    # 计算模型的损失
    y_predict = estimate.predict(x_test)
    error = mean_squared_error(y_test, y_predict)

    print("模型的损失: ",error)
    return 0

main()
```

四、模型的保存与加载

```
# 保存模型
"""
joblib.dump函数

函数原型:
    def dump(value, filename);

函数功能:
    将模型以文件的形式保存下来.

函数参数:
    value: 需要保存的模型
    filename: 模型保存的位置以及名称   模型是以 PKL结尾的
"""
# joblib.dump(estimate, "./estimate.pkl")

# 导入模型
"""
joblib.load函数

函数原型:
    load(filename);

函数功能:
    将保存的pkl文件加载到程序中

函数参数:
    filename: 需要加载的文件名

函数返回值:
    加载后的模型

"""
estimate = joblib.load("./estimate.pkl")

```

五、过拟合与欠拟合

5.1 欠拟合

现象：

测试集的数据进行测试的时候，模型的准确度很低，损失很高。

训练集的数据进行测试的时候，模型的准确度很低，损失很高。

这种情况就是发生了 欠拟合

原因：

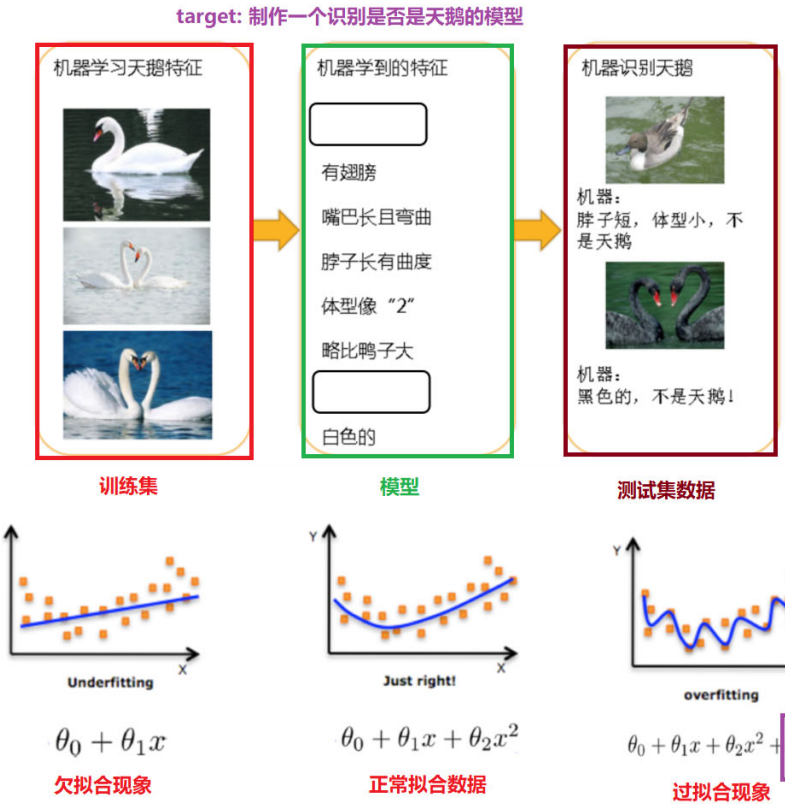
a. 特征选取过少，模型过于简单， 泛化能力太强了。

解决方式：

a. 增加数据的多样性。

b. 手动提取特征，增加多个特征值。

5.2 过拟合



现象：

测试集数据进行测试： 模型的准确度很低，模型的损失也很高

训练集数据进行测试： 模型的准确度很高，模型的损失也很低。

这种情况： 发生了过拟合现象

原因：

a.模型特征值过多，导致模型过于复杂，泛化能力很弱。

解决方式：

a. 手动去除掉一些特征 --- 特征降维

b. 去除数据中的一些异常值。 -- 噪点。

c. 正则化

L1正则化： 将高此项系数直接置为 0

L2正则化： 将高此项系数趋近与 0

$\theta_0 + \theta_1 x$

欠拟合现象

$\theta_0 + \theta_1 x + \theta_2 x^2$

正常拟合数据

$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

过拟合现象

六、岭回归算法

6.1概念

自带L2正则化的线性回归算法。同时可以调节正则化的力度： 正则化力度越高，越趋近于0.

6.2 api

```
# 岭回归 api接口
from sklearn.linear_model import Ridge
```

```
"""
Ridge成员函数

__init__成员函数:

函数原型:
    def __init__(self, alpha=1.0, *, fit_intercept=True);

函数参数:
    alpha: 正则化力度:  [1.0, 10.0] alpha正则化力度越高
    fit_intercept: 是否计算模型的偏置
"""
```

6.3 使用岭回归制作波士顿房价模型

```
# 岭回归 api接口
from sklearn.linear_model import Ridge
# 导入数据
from sklearn.datasets import load_boston
# 划分数据集
from sklearn.model_selection import train_test_split
# 归一化处理数据集
from sklearn.preprocessing import MinMaxScaler



def main():

    # 1.获取原始数据集
    lb = load_boston()

    # 2.确定特征值与目标值
    x = lb.data
    y = lb.target

    # 3.划分数据集，将数据集划分为训练集和测试集
    x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3)

    # 4.归一化处理数据
    concert = MinMaxScaler()

```

```
    # 制定转换数据的标准
    concert.fit(x_train)

    # 开始转换数据
    x_train = concert.transform(x_train)
    x_test = concert.transform(x_test)

    # 5.实例化算法对象
    estimate = Ridge(alpha=2.0)

    # 6.拟合数据,制作模型
    estimate.fit(x_train, y_train)

    # 7.打印模型的回归系数和偏置
    coef = estimate.coef_
    intercept = estimate.intercept_

    # 8.打印结果
    print("岭回归算法的回归系数: ", coef)
    print("岭回归算法的偏置: ", intercept)

    return 0

main()
```

七、特征降维

7.1 概念

降维是指在某些限定条件下，降低随机变量(特征)个数，得到一组“不相关”主变量的过程。

降低随机变量的个数:

.

加载中...

不相关的主变量的过程

两个变量之间没有任何的联系

7.2 特征降维包含内容

i. 特征选择

ii. 主成分分析

7.3 特征选择

7.3.1 概念

数据中包含冗余或无关变量（或称特征、属性、指标等），旨在从原有特征中找出主要特征

加载中...

7.3.2 方差过滤法

方差： 数据的混乱程度。

特征方差小：某个特征大多样本的值比较相近

特征方差大：某个特征很多样本的值都有差别

7.3.3 方差过滤法API接口

```
# 位置
from sklearn.feature_selection import VarianceThreshold
"""
VarianceThreshold 特征选择类

__init__成员函数

函数功能:
    初始化一个低方差过滤法的对象

函数原型:
    __init__(self, threshold=0.);

函数参数:
    threshold:  方差: 过滤掉小于这个值的方差

```

```
fit: 制定转换数据的标准
transform:对数据进行转换处理

"""
```

7.3.4 课堂练习

```
# 位置
from sklearn.feature_selection import VarianceThreshold
"""
VarianceThreshold 特征选择类

__init__成员函数

函数功能:
    初始化一个低方差过滤法的对象

函数原型:
    __init__(self, threshold=0.);

函数参数:
    threshold:  方差: 过滤掉小于这个值的方差

fit: 制定转换数据的标准
transform:对数据进行转换处理

"""

# 原始数据
data = [[90, 2, 10, 40],
        [60, 4, 15, 45],
        [75, 3, 13, 46]]

# 目标去除冗余的数据
concert = VarianceThreshold(threshold=2)

# 制定转换数据的标准
concert.fit(data)

# 开始转换数据
result = concert.transform(data)

```

```
print("result : ", result )
```

7.4 主成分分析

7.4.1 概念

高维数据转化为低维数据的过程，在此过程中可能会舍弃原有数据、创造新的变量.

高维数据转换为低维数据

加载中...

创造新的变量

加载中...

7.4.2 API接口

```
# API接口位置
from sklearn.decomposition import PCA
"""
PCA 成员函数

__init__成员函数

函数原型
    __init__(self, n_components=None);

函数参数:
    n_components: 保留原来 x% 的信息, 一般是float类型的数据. float

fit: 制定转换数据的标准
transform:对数据进行转换.
"""
```

7.4.3 购物篮分析案例

加载中...

i. 数据介绍

order_products__prior.csv：订单与商品信息

i. order_id, product_id, add_to_cart_order, reordered

products.csv：商品信息

i. product_id, product_name, aisle_id, department_id

orders.csv: 用户的订单信息

i. order_id,user_id,eval_set,order_number,….

aisles.csv:商品所属具体物品类别

i. aisle_id, aisle

i. 思路

1. 读取四张表的内容
2. 合并四张表的内容 -- 通过指定的字段
3. 将合并好的表,纵坐标 user_id 横坐标 aisle_id制作一张新的表

```
# 导入pandas模块读取四张表格
import pandas as pd

# 导入降维的api接口
from sklearn.decomposition import PCA

def main():

    # 1.读取四张表的内容  pd.read_csv()
    aisles = pd.read_csv("./aisles.csv")
    order_products__prior = pd.read_csv("./order_products__prior.csv")
    orders = pd.read_csv("./orders.csv")
    products = pd.read_csv("./products.csv")

    # 2.合并四张表里面的内容 pd.merge()
    mt1 = pd.merge(order_products__prior, products, on=["product_id", "product_id"])
    mt2 = pd.merge(mt1, orders, on=["order_id", "order_id"])

    mt = pd.merge(mt2, aisles, on=["aisle_id", "aisle_id"])

    # 3. 通过横纵坐标转换为一个新的表格 pd.crosstab()
    data = pd.crosstab(mt["user_id"], mt["aisle"])

    print("data.shape :", data.shape)
    print(data[:5])

```

```
    # 4. 实例化转换数据的对象
    concert = PCA(n_components=0.9)

    # 5.开始转换数据
    concert.fit(data)
    result = concert.transform(data)
    print(result)
    print(result.shape)




    return 0


main()
```

八、聚类算法 kmeans

8.1 概念

kmeans算法是一个无监督学习算法(输入的数据，只有特征值，没有目标值)

加载中...

8.2 聚类的过程

加载中...

8.3 Kmeans算法api接口

```
# k-means算法的api接口
from sklearn.cluster import KMeans
"""
KMeans类

__init__成员函数

```

```
函数原型:
    __init__(self, n_clusters=8);

函数参数:
    n_clusters: 具体聚成几个类别。

fit 成员函数

函数原型:
    def fit(self, X);

predict成员函数

函数原型:
    predict(self, X);


"""
```

8.4 购物篮聚类

```
# 导入pandas模块读取四张表格
import pandas as pd

# 导入降维的api接口
from sklearn.decomposition import PCA

# 导入Kmeans算法
from sklearn.cluster import KMeans


def main():

    # 1.读取四张表的内容  pd.read_csv()
    aisles = pd.read_csv("./aisles.csv")
    order_products__prior = pd.read_csv("./order_products__prior.csv")
    orders = pd.read_csv("./orders.csv")
    products = pd.read_csv("./products.csv")

    # 2.合并四张表里面的内容 pd.merge()
    mt1 = pd.merge(order_products__prior, products, on=["product_id", "product_id"])
```

```
    mt2 = pd.merge(mt1, orders, on=["order_id", "order_id"])

    mt = pd.merge(mt2, aisles, on=["aisle_id", "aisle_id"])

    # 3. 通过横纵坐标转换为一个新的表格 pd.crosstab()
    data = pd.crosstab(mt["user_id"], mt["aisle"])

    print("data.shape :", data.shape)
    print(data[:5])

    # 4. 实例化转换数据的对象
    concert = PCA(n_components=0.9)

    # 5.开始转换数据
    concert.fit(data)
    result = concert.transform(data)

    # 6. 实例化一个算法
    estimate = KMeans(n_clusters=4)

    estimate.fit(result[:500])

    # 查看前500个数据的类别
    temp = estimate.predict(result[:500])
    print(temp)









    return 0


main()
```