[2022.9.11]

机器学习笔记

摘要

[人工智能]

[日期及邮编地址]

[ID名称]

[机器学习]

[机器学习准备知识 3](#_Toc115867472)

[线性回归模型 3](#_Toc115867473)

[算法库：scikit-learn 3](#_Toc115867474)

[源数据准备 3](#_Toc115867475)

[读取数据： 3](#_Toc115867476)

[处理数据 3](#_Toc115867477)

[求解线性回归问题 3](#_Toc115867478)

[评估模型表现 3](#_Toc115867479)

[图形展示 4](#_Toc115867480)

[散点图 4](#_Toc115867481)

[多张图同时展示 4](#_Toc115867482)

[逻辑回归 4](#_Toc115867483)

[简单二分类问题公式 4](#_Toc115867484)

[更复杂分类问题时的公式 5](#_Toc115867485)

[逻辑回归求解 5](#_Toc115867486)

[逻辑回归模型评估：准确率 5](#_Toc115867487)

[聚类 5](#_Toc115867488)

[K均值聚类Kmeans（无监督学习） 5](#_Toc115867489)

[K近邻分类KNN（监督学习） 6](#_Toc115867490)

[均值漂移聚类Mean-shift（无监督学习） 6](#_Toc115867491)

[Kmeans实现聚类 6](#_Toc115867492)

[决策树 7](#_Toc115867493)

[决策树求解方法： 7](#_Toc115867494)

[异常检测 7](#_Toc115867495)

[主成分分析 8](#_Toc115867496)

机器学习准备知识

特征值：对象的属性，也可称为变量

维度：有几个特征值就是几维

可视化数据工具：matplotlib

数据处理工具：pandas

机器学习流程：

1. 数据预处理
2. 建立模型
3. 模型训练
4. 模型预测
5. 模型评估

线性回归模型

y=ax+b

对于一个线性模型，有三步要做：1.准备源数据。2.求解线性回归问题a、b。3.评估模型表现

算法库：scikit-learn

python语言中专门针对机器学习应用而发展起来耳朵一款开原框架（算法库），可以实现数据预处理、分类、回归、降维、模型选择等常用的机器学习算法

源数据准备

读取数据：

import pandas as pd

data = pd.read\_csv(‘xxx.csv’)

x = data.loc[:,’x’]

y = data.loc[:,’y’]

处理数据

sklearn模型接收的参数必须是二维数组，此时需要将数据处理成二维数组。

import numpy as np

x = np.array(x).reshape(-1,1)

y = np.array(y).reshape(-1,1)

求解线性回归问题

寻找a、b（y=ax+b）将数组（x = x1，x2…xn y = y1，y2…yn）输入fit函数

from sklearn.linear\_model import LinearRegression

lr\_model = LinearRegression()

lr\_model.fit(x,y)

展示a、b

a = lr\_model.coef\_

b = lr\_model.intercept\_

评估模型表现

计算y与y’的均方误差（MSE）、R方值（R2\_score）

from sklearn.metrics import mean\_squared\_error,r2\_score

MSE = mean\_squared\_error(y,y\_predict)

R2 = r2\_score(y,y\_predict)

画图对比y与y‘，可视化模型表现：

from matplotlib import pyplot as plt

plt.scatter(y,y’)

MSE越小越好，R方越接近1越好

y’ vs y集中度越高越好（越接近直线分布）

图形展示

散点图

import matplotlib.pyplot as plt

plt.scatter(x,y)

plt.show()

多张图同时展示

fig1 = plt.subplot(211)

plt.scatter(x1,y1)

fig2 = plt.subplot(212)

plt.scatter(x2,y2)

plt.show()

注：subplot后面的参数如211意思为2行1列第1个

单因子线性回归

一个变量的线性回归，预测结果可能会存在不准确的现象

多因子线性回归

多个变量的线性回归，处理复杂问题的时候性能比较好，预测结果会更准确

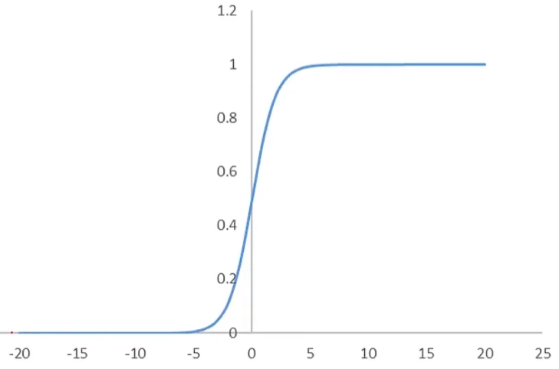
逻辑回归

对于分类问题，线性回归存在局限性，当样本量变大以后，当x距离远点变远，预测开始不准确。此时使用逻辑回归处理分类问题比较适合。

逻辑回归用于解决分类问题的一种模型。根据数据特征或属性，计算其归属于某一类别的概率，根据概率数值y判断其所属类别。主要应用场景：二分类问题。

简单二分类问题公式

为概率分布函数，y为类别结果，x为特征值



更复杂分类问题时的公式

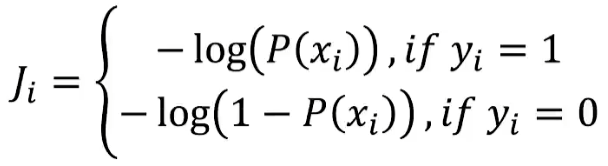
g(x)为决策边界，g(x)将待分类对象最理想的区分开

分类问题最核心的点在于将决策边界找出，逻辑回归结合多项式边界函数可解决复杂的分类问题

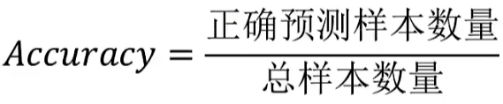
逻辑回归求解

根据训练样本，寻找类别边界g(x)内a1、a2、a3.。。。

逻辑回归求解：最小化损失函数



逻辑回归模型评估：准确率



越接近1越好

聚类

K均值聚类Kmeans（无监督学习）

K-均值算法：以空间中k个点为中心进行聚类，对组靠近他们的对象归类，是聚类算法中最为基础但也是最为重要的算法

算法流程：

1. 选择聚类的个数k
2. 确定聚类中心
3. 根据点到聚类中心确定各个点所属类别
4. 根据各个类别数据更新聚类中心
5. 重复以上步骤直到收敛（中心点不再变化）

优点：

1. 原理简单，实现容易，收敛速度快
2. 参数少，方便使用

缺点：

1. 必须设置簇的数量（分类数量，分成几类）
2. 随机选择初始聚类中心，结果可能缺乏一致性

K近邻分类KNN（监督学习）

给定一个训练数据集，对新的输入实例，在训练数据集中找到与该实例最邻近的K个实例（也就是K个邻居），这K个实例的多数属于某个类，就把该输入实例分类的到这个类中。最简单的机器学习算法之一

均值漂移聚类Mean-shift（无监督学习）

均值漂移算法：一种基于密度梯度上升的聚类算法（沿着密度上升方向寻找聚类中心点）

算法流程：

1. 随机选择未分类点作为中心点
2. 找出离中心点距离在带宽之内的点，记做集合S
3. 计算从中心点到集合S中每个元素的偏移向量M
4. 中心点以向量M移动
5. 重复步骤2-4，直到收敛
6. 重复1-5知道所有的点都被分类
7. 分类：根据每个类，对每个点的访问频率，取访问频率最大的那个类，作为当前点集的所属类

Kmeans实现聚类

模型训练

from sklearn.cluster import Kmeans

km = Kmeans(n\_cluster = 3, random\_state = 0)

km.fit(x)

获取模型确定的中心点

centers = km.cluster\_centers\_

准确率计算

from sklearn.metrics import accuracy\_score

accuracy = accuracy\_score(y, y\_predict)

预测结果矫正

结果矫正

y\_cal = []

for i in y\_predict:

if i == 0:

y\_cal.append(2)

elif i == 1:

y\_cal.append(1)

else:

y\_cal.append(0)

print(y\_predict, y\_cal)

Meanshift实现聚类

自动计算带宽（区域半径）

from sklearn.cluster import MeanShift, estimate\_bandwidth

bandwidth = estimate\_bandwidth(x, n\_samples = 500)

模型建立与训练

ms = MeanShift(bandwidth = bandwidth)

ms.fit(x)

KNN实现分类

模型训练

from sklearn.neighbors import KneighborsClassifier

knn = KneighborsClassifier(n\_neighbors = 3)

knn.fit(x,y)

决策树

一种对实例进行分类的树形结构，通过多层判断区分目标所属类别

本质：通过多层判断，从训练数据集中归纳出一组分类规则

优点：

计算量小，运算速度快

易于理解，可清晰查看各属性的重要性

缺点：

忽略属性间的相关性

样本类别分布不均匀时，容易影响模型表现

决策树分类核心：特征选择，每一个节点应该选用哪个特征

决策树求解方法：

ID3、C4.5、CART

ID3：

利用信息熵原理选择信息增益最大的属性作为分类属性，递归地拓展决策树的分支，完成决策树的构造

信息熵是度量随机变量不确定性的指标，熵越大，变量的不确定性越大

利用ID3算法求出各节点的信息增益，根据信息增益画出决策树

目标：划分后样本分布不确定性尽可能小，即信息熵越小、信息增益越大越好

异常检测

通过高斯方程（正态分布）找到异常点（概率分布小于一个值的点）

主成分分析

数据降维：在某些限定条件下，降低随机变量个数，得到一组“不相关”主变量的过程

作用：

减少模型分析数据量，提升处理效率，降低计算难度

实现数据可视化

数据降维的实现：主成分分析（PCA）

主成分分析是数据降维技术中，应用最多的方法

目标：寻找k（k<n）维新数据，使他们反映事物的主要特征

核心：在信息损失尽可能少的情况下，降低数据维度

如何保留主要信息：投影后的不同特征数据尽可能分的开（不相关）

如何实现：使投影后数据的方差最大，因为方差越大数据也越分散

计算过程：

1. 原始数据预处理（标准化：u=0，标准差=1）
2. 计算协方差矩阵特征向量、及数据在各特征向量投影后的方差
3. 根据需求（任务指定或方差比例）确定降维维度k
4. 选取k维特征向量，计算数据在其形成空间的投影

决策树模型训练

from sklearn import tree

dc\_tree = tree.DecisionTreeClassifier(criterion = ‘entropy’, min\_samples\_leaf = 5)

dc\_tree.fit(x,y)

可视化决策树：

tree.plot\_tree(dc\_tree, filled = ‘True’, feature\_names = [‘xxx1’, ‘xxx2’], class\_names = [‘nn1’, ‘nn2’])

模型评价与优化

过拟合和欠拟合

模型不合适，导致其无法对数据实现有效预测

过拟合原因：

1. 模型结构过于复杂（纬度过高）
2. 使用了过多属性，模型训练时包含了干扰项信息

解决办法：

1. 简化模型结构（使用低阶模型，比如线性模型）
2. 数据预处理，保留主要成分信息（数据PCA处理）
3. 在模型训练时，增加正则化项

混淆矩阵

分类任务中，相比单一的预测准确率，混淆矩阵提供了更全面的模型评估信息

通过混淆矩阵，我们可以计算出多样的模型表现衡量指标

模型优化

目标：在确定模型类别后，如何让模型表现的更好

尝试以下方法：

1.遍历核心参数组合，评估对应模型表现（比如：逻辑回归边界函数考虑多项式，KNN尝试不同的n\_neighbors值）

2.扩大数据样本

3.增加或减少数据属性

4.数据降维

5.正则化处理，调整正则项的数值

深度学习之多层感知器