基于MNist数据集对机器学习进行入门学习

|  |  |  |
| --- | --- | --- |
| 姓名 | 班级 | 学号 |
| 王红麟 | 硬件一班 | 161403106 |

实验预备内容

1. python3环境，pycharm，（jupyter notebook）
2. 需要用到的包：numpy,matplotlib,skearn
3. mnist手写数据集

实验过程

1. 学习机器学习的概念知识

(1)机器学习的基本任务分为：分类任务和回归任务。

(2)机器学习可分为：监督学习，无监督学习，半监督学习，加强学习；也可分为批量学习，在线学习，参数学习，非参数学习。

注：脱离具体问题，谈哪个算法好是没有；意义的。

1. 学习Numpy基础（由于我先用的jupyternote，所以先学习了其相关的基础操作，以及一些魔法命令）

较为重要的东西：

1. array，arange，random（可帮助我们轻松的获得一些简单的，用来操作的数据）
2. 用于在numpy中是把数据看做矩阵的，并有极高的效率，学习numpy中矩阵的操作concatnate，h/vstack，split等
3. 学习fancy indexing，很重要，knn的predict的底层编写就用的它。
4. 学习matplotlib基础

简单的懂得如何使用pyplot中的plot和scatter来实现数据可视化。

1. 学习skearn基础
2. KNN——K近邻算法

Knn算法非常简单，对于底层实现的核心我认为是：

Distance=[sqrt(np.sum((x\_train-x)\*\*2)) for x\_train in X\_train]

随着对KNN的学习引入很多东西：tran\_test\_split，predict，score，accurscore，通过imshow的数据可视化，通过网格搜索寻找超参数（K，是否考虑距离，P），数据归一化（最值归一化，均值方差归一化）

1. 线性回归

Mnist数据集由于一张图片为28\*28像素点构成，所以有784个特征值，但特征值之间并没有线性关系，所以使用线性回归算法效果极差(以至于使用R\_squared指标得出的结果为负数！——其他衡量回归算法指标还有MSE，RMS，MAE)

底层实现核心：num=(x\_i-x\_mean).dot(y\_i=y\_mean) //尽可能使用矩阵操作

d=(x\_i-x\_mean).dot(x\_i-x\_mean) //相较生成表达式还要快50倍

a=num/d

b=y\_mean-a\*x\_mean

y=a\*X\_predict+b

1. 逻辑回归

属于判别式模型，有很多正则化模型的方法（L0， L1，L2，etc）

只能处理两分类问题（在此基础上衍生出来的softmax可以用于多分类），且必须线性可分

1. 朴素贝叶斯

碰巧我最近在学习概率论，对全概率公式和贝叶斯公式理解很深（贝叶斯公式也称逆概率公式，拿着结果去反推概率，对原来的概率有修正）

1. 决策树

易于解释。它可以毫无压力地处理特征间的交互关系并且是非参数化的

1. SVM支持向量机

高准确率，为避免过拟合提供了很好的理论保证，而且就算数据在原特征空间线性不可分，只要给个合适的核函数，它就能运行得很好

1. 人工神经网络

并行分布处理能力强,分布存储及学习能力强但不能观察之间的学习过程，输出结果难以解释，会影响到结果的可信度和可接受程度；

1. K-Means聚类

算法尝试找出使平方误差函数值最小的k个划分。当簇是密集的、球状或团状的，且簇与簇之间区别明显时，聚类效果较好。

实验小结

随着学习的进行，越发感觉到python和machine learning的神奇，不过还是由于时间的问题对最后几个算法的学习有些草率，不过实验的结束不代表我对机器学习学习的结束随后会慢慢补充圆满。