链滴

机器学习基础知识

作者：kanadeblisst

原文链接：https://ld246.com/article/1565602650227

来源网站：链滴

一、数据预处理

一行一样本，一列一特征

import sklearn.preprocessing as sp

1）标准化

sp.scale(原样本) # 返回值为标准化样本

定义：将样本矩阵中各列的平均值和标准差统一为0和1

实现：假设有样本[a b c]，均值为m，标准差为s，则标准化的样本为[(a-m)/s (b-m)/s (c-m)/s]

2）范围缩放

mms = sp.MinMaxScaler(feature_range=(0, 1))

mms.fit_transform(原样本) # 返回范围缩放后的样本

定义：样本矩阵中每一列的最大值和最小值为某个给定值(通常是[0, 1])，其它元素线性缩放

实现：k * max + b = 1, k *min + b =0,解出k，b其他值则代入得到

3）归一化

sp.normalize(原样本, norm='l1') # 返回归一化样本矩阵

- l1：l1范数，样本中各元素绝对值的1次方之和
- l2：l2范数，样本中各元素绝对值的2次方之和
- lk：lk范数，样本中各元素绝对值的k次方之和

定义：用样本中的每一列的数除以这一列所有数绝对值之和，使得处理后的每一列的元素的绝对值之为1，这是norm参数等于l1的

4）二值化

bin = sp.Binarizer(threshold=阈值)

bin.transform(原样本)# 返回二值化样本

定义：根据一个事先设定的阈值，将样本矩阵中不大于阈值的元素设定为0，大于阈值的元素设定为1

5）独热编码

ohe = sp.OneHotEncoder(sparse=紧缩格式(默认True),dtype=目标类型)

ohe.fit_transform(原样本)# 返回独热编码样本

顺序随意是指1不一定要对应10，可以1对应01，7对应10。

在计算机中，因为位数相同，它实际只是存储1的位置，而不记录所有的0和1

6）标签编码

lbe = sp.LabelEncoder()

lbe.fit_transform(原样本) # 返回标签编码样本

lbe.inverse_transform(标签编码样本) # 返回原样本

定义：将一列字符串形式的特征值，按照字典排序，用每个字符在排序序列中的索引号表示该字符，般用于非数字数据

二、线性回归和梯度下降算法

1、根据机器学习的任务把不同的业务模型划分为四种基本问题：

回归问题 、分类问题

：在有监督条件下，根据已知的输入和输出，构建预测模型，对未知输出的输入给出大概率的输出 。果输出是连续域的值，则被划分为回归问题。如果输出被限定在离散域(已知的一些值或类别)内，则划分为分类问题。

聚类问题：在无监督模式下，根据输入的特征划分族群。

降维问题：在无监督模式下，对输入特征进行取舍以降低维度。

2、线性回归：通过线性方程描述输出和输入之间的关系，以解决回归问题

1）预测函数：y = kx + b

最小二乘法(上一篇博客解释了)：总样本误差E = (y1-y1')^2 + (y2-y2') ^2 + ... + (yn-yn') ^2。为

方便计算，实际的E一般取E/2

总误差E其实是关于系数k、b的函数，被称为损失函数。函数在空间表现为一个空间曲面，形状类似球体的下半部分

2）梯度下降算法

梯度(百度百科解释)：一个向量，表示某一函数在该点处的方向导数沿着该方向取得最大值，即函数该点处沿着该方向（此梯度的方向）变化最快，变化率最大（为该梯度的模）

梯度下降算法：给定一个起始点，求得该点的梯度，然后沿着梯度的反向移动一些距离(这个移动的距和学习率的大小有关)得到第二个点，重复上面的过程，直到损失值达到某个值，或者深度(重复次数)经超过某个值

原理：梯度既然是方向导数沿着该方向取得最大值，也就是说，沿梯度方向，函数值增加最快。同样知，方向导数的最小值在梯度的相反方向取得，此最小值为最大值的相反数，从而沿梯度相反方向函值的减少最快，然后我们沿这个方向行进一些距离得到的点(这个点还在损失函数上)一定比上一个点低或者说z值更小)，重复一定次数，则最后的结果就接近于最小值了。

至于为什么沿梯度方向增加最快，这就是数学上的问题，有兴趣的可以了解一下。当然大部分时候，们能使用别人已经造出的轮子，不必去刻意关心过程。

疑问：损失函数既然是一个类似于下半球的曲面，那么在数学上完全可以求得最低点的切平面，进而到最小值，为什么还要梯度下降算法？因为梯度下降算法是应用于计算机的，而计算机并不像人，它适合做一些重复不用思考的工作，而且这样做在时间上也不比人思考来的慢

计算：E = f(k, b) , 梯度为(k", b"),k"是E对k求偏微分得到的表达式，而b"是E对b求偏微分得到的表达。那么某一点的梯度只需要将点的k,b带入到两个表达式得到相应的(k", b")

过程示例代码：kb.py

调库示例代码：sklearn_kb.py

三、岭回归

原理：在最小二乘法的基础上减掉一个正则项乘以正则强度，即总误差E = f(k, b) - 正则项 * 正则强，目的是为了防止过拟合(过分匹配训练样本)。

解释：而正则项是除常数项的所有系数的平方和(因为线性回归的未知数x不一定只有一个，通用表达为y= w0 + ... + wnxn)，假设y = w0 + w1x1 + w2x2，正则项为(w1^2 + w2 ^2)，正则强度是可参数，会根据实际情况做出测试调整。当正则项为所有系数的和时，则是另一种回归，被称作lasso归。

sklearn代码：

```
import sklearn.linear_model as lm
#创建模型
model = lm.Ridge(正则强度, fit_intercept=是否训练截距,
    max_iter=最大迭代次数)
# 训练模型
model.fit(训练输入样本，训练输出样本)
# 预测
预测输出 = model.predict(预测输入样本)
```

四、多项式回归

其实和线性回归原理相同，只是表达式改成了y = w0 + w1x + w2x^2 + ...，但是它并不是直接利用项式的表达式求总误差，而是将高次项看成是一次项的扩展再用管道连接。我并不理解这样做的原理不过没关系，代码直接用就行。

代码：

```
import sklearn.linear_model as lm

# 创建模型
model = pl.make_pipeline(
    sp.PolynomialFeatures(10), # 多项式特征扩展器，10为多次项的最高项的次数，即十次多项式
    lm.LinearRegression())     # 线性回归器
# 训练模型
model.fit(x, y)
# 预测模型
model.predict(x)
```

五、决策树

核心思想：相似的输入必会产生相似的输出。

原理：首先从训练样本矩阵中选择第一个特征进行划分，使每个子表中该特征的值全部相同(比如第一特征是男女，则可以划分出两个子表，男表和女表)，然后再在每个子表中选择下一个特征按照同样的则继续划分更小的子表(比如第二个特征是年龄，我可以划分成三个子表(当然根据情况的不同而不同)小于18，大于18小于60，大于60，则在男女表中分别又有三个子表，每个子表下的特征值都相同)，断重复直到所有的特征全部使用完为止，此时便得到叶级子表，其中所有样本的特征值全部相同。

解释：决策树是一种分类方法，用于对样本的特征分类。而分类完成之后，得到的结果是同一类(或者为表)的所有特征基本相同，然后根据某一类的所有样本通过平均(回归)或者投票(分类)得到一个输出那么，当有新的待预测样本需要预测输出时，我只需知道样本属于哪个类(表)。

工程优化(剪枝)：不必用尽所有的特征，叶级子表中允许混杂不同的特征值，以此降低决策树的层数在精度牺牲可接受的前提下，提高模型的性能。通常情况下，可以优先选择使信息熵减少量最大的特作为划分子表的依据。(通俗的讲就是有些特征值并不区分，比如第一个特征是男女，我并不分成两个，而是放在一个表里，这种情况一般是男女这个特征对输出的影响不大，至于怎么知道这个特征值对出结果的影响程度，我暂时还没明白)

集合算法(组合树)

1） 自助聚合：每次从总样本矩阵中以有放回抽样的方式随机抽取部分样本构建决策树，这样形成多包含不同训练样本的决策树，以削弱某些强势样本对模型预测结果的影响，提高模型的泛化特性。

2） 随机森林：在自助聚合的基础上，每次构建决策树模型时，不仅随机选择部分样本，而且还随机择部分特征，这样的集合算法，不仅规避了强势样本对预测结果的影响，而且也削弱了强势特征的影，是模型的预测能力更加泛化。

3） 正向激励：首先为样本矩阵中的样本随机分配初始权重，由此构建一棵带有权重的决策树，在由决策树提供预测输出时，通过加权平均或者加权投票的方式产生预测值。将训练样本代入模型，预测输出，对那些预测值与实际值不同的样本，提高其权重，由此形成第二棵决策树。重复以上过程，构出不同权重的若干棵决策树。