信息论基础

2019年5月18日 11:05

1.熵：度量的是消息中所含的信息量，其中去除了由消息固有结构所决定的部分，比如语言结构的冗余性以及语言中字母、词的使用品读等统计特性。

$$H(X) = -\sum_i p_i \log p_i$$

$H(X)$被称为随机变量的x熵，表示随机变量不确定的度量，是对所有可能发生的事件产生的信息量的期望

这一定义可以用来推算传递经二进制编码后的信息所需要的带宽

2.联合熵

$$H(X,Y) = -\sum_x \sum_y p(x,y) \log p(x,y)$$

3.条件熵

$$H(Y|X) = \sum_x p(x) H(Y|X=x) = -\sum_x p(x) \sum_y p(y|x) \log p(y|x) = -\sum_x \sum_y p(x,y) \log p(y|x)$$

$$= \sum_{x,y} (x,y) \log p(y|x)$$

$$H(Y|X) = H(X,Y) - H(X)$$

4.信息增益基尼不纯度

将来自集合中的某种结果随记应用于集合中某一数据项的预期差率

决策树的不同分类算法的原理及应用场景

2019年5月18日 11:10

期望信息越小，信息增益越大，从而纯度越高。

决策树算法作为一种分类算法，目标就是将具有p维特征的n个样本分到c个类别中去。相当于做一个投影，c=f(n)，将样本经过一种变换赋予一种类别标签。决策树为了达到这一目的，可以把分类的过程表示成一棵树，每次通过选择一个特征pi来进行分叉。

ID3用信息增益，C4.5用信息增益率，CART用gini系数。

1.ID3算法
ID3的算法的核心思想就是以信息增益度量属性选择，选择分裂后信息增益最大的属性进行分裂
ID3的算法思想：
1）自顶向下的贪婪搜索遍历可能的决策树空间构造决策树(此方法是ID3算法和C4.5算法的基础)；
2）从“哪一个属性将在树的根节点被测试”开始；
3）使用统计测试来确定每一个实例属性单独分类训练样例的能力，分类能力最好的属性作为树的根结点测试
4）然后为根结点属性的每个可能值产生一个分支，并把训练样例排列到适当的分支（也就是说，样例的该属性值对应的分支）之下。
5）重复这个过程，用每个分支结点关联的训练样例来选取在该点被测试的最佳属性。

这形成了对合格决策树的贪婪搜索，也就是算法从不回溯重新考虑以前的选择。
优点：
理论清晰、方法简单、学习能力较强
缺点：
（1）只能处理分类属性的数据，不能处理连续的数据；
（2）划分过程会由于子集规模过小而造成统计特征不充分而停止；
（3）ID3算法在选择根节点和各内部节点中的分支属性时，采用信息增益作为评价标准。信息增益的缺点是 倾向于选择取值较多的属性，在有些情况下这类属性可能不会提供太多有价值的信息。

2.C4.5算法
它是决策树(决策树也就是做决策的节点间的组织方式像一棵树，其实是一个倒树)核心算法，ID3的改进算法，所以基本上了解了一半决策树构造方法就能构造它。
3.CART分类树
CART使用基尼系数选择最佳划分变量和划分值(Breiman et al., 1984)
https://blog.csdn.net/zeo_m/article/details/80076721

回归树原理

2019年5月18日 11:11

1.决策树实际上是将空间用超平面进行划分的一种方法，每次分割的时候，都将当前的空间一分为二， 这样使得每一个叶子节点都是在空间中的一个不相交的区域，在进行决策的时候，会根据输入样本每一维feature的值，一步一步往下，最后使得样本落入N个区域中的一个（假设有N个叶子节点）

决策树防止过拟合手段

2019年5月18日 11:12

1.定义

过度拟合(overfitting):给定一个假设空间H,一个假设h属于H,如果存在其他的假设h'属于H,使得在训练样例上h的错误率比h'小,但在整个实例分布上h'比h的错误率小,那么就说假设h过度拟合训练数据.

overfittingt是这样一种现象:一个假设在训练数据上能够获得比其他假设更好的拟合,但是在训练数据外的数据集上却不能很好的拟合数据.此时我们就叫这个假设出现了overfitting的现象.

2.产生原因

原因1：样本问题

（1）样本里的噪音数据干扰过大，大到模型过分记住了噪音特征，反而忽略了真实的输入输出间的关系；（什么是噪音数据？）

（2）样本抽取错误，包括（但不限于）样本数量太少，抽样方法错误，抽样时没有足够正确考虑业务场景或业务特点，等等导致抽出的样本数据不能有效足够代表业务逻辑或业务场景；

（3）建模时使用了样本中太多无关的输入变量。

原因2：构建决策树的方法问题

在决策树模型搭建中，我们使用的算法对于决策树的生长没有合理的限制和修剪的话，决策树的自由生长有可能每片叶子里只包含单纯的事件数据或非事件数据，可以想象，这种决策树当然可以完美匹配（拟合）训练数据，但是一旦应用到新的业务真实数据时，效果是一塌糊涂。

上面的原因都是现象，但是其本质只有一个，那就是“业务逻辑理解错误造成的”，无论是抽样，还是噪音，还是决策树等等，如果我们对于业务背景和业务知识非常了解，非常透彻的话，一定是可以避免绝大多数过拟合现象产生的。因为在模型从确定需求，到思路讨论，到搭建，到业务应用验证，各个环节都是可以用业务敏感来防止过拟合于未然的。

https://blog.csdn.net/sinat_32043495/article/details/78729610

针对原因1的解决方法：

合理、有效地抽样，用相对能够反映业务逻辑的训练集去产生决策树；

针对原因2的解决方法（主要）：

剪枝：提前停止树的增长或者对已经生成的树按照一定的规则进行后剪枝。

剪枝的方法

先剪枝、后剪枝

模型评估

2019年5月18日 11:12

1.

分类模型误差分为：训练误差（training error）、泛化误差（generalization error）。

一个好的模型需要有较低的泛化误差和训练误差。

自助法（bootstrap）：

训练集是对于原数据集的有放回抽样，如果原始数据集N，可以证明，大小为N的自助样本大约包含原数据63.2%的记录。当N充分大的时候，1-（1-1/N）^(N) 概率逼近 1-e^(-1)=0.632。抽样 b 次，产生 b 个bootstrap样本，则，总准确率为（accs为包含所有样本计算的准确率）：

$$acc_{boot} = \frac{1}{b}\sum_{i=1}^{b}(0.632 \times \varepsilon_i + 0.368 \times acc_s)$$

准确度的区间估计：

将分类问题看做二项分布，则有：
令 X 为模型正确分类，p 为准确率，X 服从均值 Np、方差 Np（1-p）的二项分布。acc=X/N为均值 p，方差 p（1-p）/N 的二项分布。acc 的置信区间：

$$P\left(-Z_{\frac{\alpha}{2}} \leq \frac{acc-p}{\sqrt{p(1-p)/N}} \leq Z_{1-\frac{\alpha}{2}}\right) = 1-\alpha$$

$$P \in \frac{2 \times N \times acc + Z_{\frac{\alpha}{2}}^2 \pm Z_{\frac{\alpha}{2}}\sqrt{Z_{\frac{\alpha}{2}}^2 + 4 \times N \times acc - 4 \times N \times acc^2}}{2(N + Z_{\frac{\alpha}{2}}^2)}$$

sklearn参数详解

2019年5月12日 10:00

1.LinearSVC

```
class sklearn.svm.LinearSVC(penalty='l2', loss='squared_hinge', dual=True, tol=
0.0001, C=1.0, multi_class='ovr', fit_intercept=True, intercept_scaling=1,
class_weight=None, verbose=0, random_state=None, max_iter=1000)
```

penalty:正则化参数，L1和L2两种参数可选，仅LinearSVC有。
loss:损失函数，有‘hinge’和‘squared_hinge’两种可选，前者又称L1损失，后者称为L2损失，默认是是’squared_hinge’，其中hinge是SVM的标准损失，squared_hinge是hinge的平方。
dual:是否转化为对偶问题求解，默认是True。
tol:残差收敛条件，默认是0.0001，与LR中的一致。
C:惩罚系数，用来控制损失函数的惩罚系数，类似于LR中的正则化系数。
multi_class:负责多分类问题中分类策略制定，有‘ovr’和‘crammer_singer’ 两种参数值可选，默认值是’ovr’，'ovr'的分类原则是将待分类中的某一类当作正类，其他全部归为负类，通过这样求取得到每个类别作为正类时的正确率，取正确率最高的那个类别为正类；‘crammer_singer’ 是直接针对目标函数设置多个参数值，最后进行优化，得到不同类别的参数值大小。
fit_intercept:是否计算截距，与LR模型中的意思一致。
class_weight:与其他模型中参数含义一样，也是用来处理不平衡样本数据的，可以直接以字典的形式指定不同类别的权重，也可以使用balanced参数值。
verbose:是否冗余，默认是False.
random_state:随机种子的大小。
max_iter:最大迭代次数，默认是1000。

对象
coef_:各特征的系数（重要性）。
intercept_:截距的大小（常数值）。

2.NuSVC

```
class sklearn.svm.NuSVC(nu=0.5, kernel='rbf', degree=3, gamma='auto', coef0=0.0,
shrinking=True, probability=False, tol=0.001, cache_size=200, class_weight=None,
verbose=False, max_iter=-1, decision_function_shape='ovr', random_state=None))
```

nu:训练误差部分的上限和支持向量部分的下限，取值在（0，1）之间，默认是0.5
kernel:核函数，核函数是用来将非线性问题转化为线性问题的一种方法，默认是“rbf”核函数，常用的核函数有以下几种：
· Linear 线性核函数

· Poly 多项式核函数

· Rbf 高斯核函数

· Sigmod sigmod核函数

· Precomputed 自定义核函数

degree:当核函数是多项式核函数的时候，用来控制函数的最高次数。（多项式核函数是将低维的输入空间映射到高维的特征空间）
gamma:核函数系数，默认是“auto”，即特征维度的倒数。
coef0:核函数常数值(y=kx+b中的b值)，只有‘poly’和‘sigmoid’核函数有，默认值是0。
max_iter:最大迭代次数，默认值是-1，即没有限制。
probability:是否使用概率估计，默认是False。
decision_function_shape:与'multi_class'参数含义类似。
cache_size:缓冲大小，用来限制计算量大小，默认是200M。

对象
support_:以数组的形式返回支持向量的索引。
support_vectors_:返回支持向量。
n_support_:每个类别支持向量的个数。
dual_coef_:支持向量系数。
coef_:每个特征系数（重要性），只有核函数是LinearSVC的时候可用。
intercept_:截距值（常数值）。

3.SVC

```
class sklearn.svm.SVC(C=1.0, kernel='rbf', degree=3, gamma='auto', coef0=0.0,
shrinking=True, probability=False, tol=0.001, cache_size=200, class_weight=None,
verbose=False, max_iter=-1, decision_function_shape='ovr', random_state=None)
```

C:惩罚系数。
SVC和NuSVC方法基本一致，唯一区别就是损失函数的度量方式不同（NuSVC中的nu参数和SVC中的C参数）。

方法
这三种分类方法的方法基本一致:
decision_function(X):获取数据集X到分离超平面的距离。
fit(X, y):在数据集(X,y)上使用SVM模型。
get_params([deep]):获取模型的参数。
predict(X):预测数据值X的标签。
score(X,y):返回给定测试集和对应标签的平均准确率。

Python绘制决策树

2019年5月18日 11:13