概念

2019年5月11日 9:05

1.机器学习：通过一个叫神经网络的算法使得机器学习取得进步
原理：模拟人类的大脑运作过程
发源：AI

2.包括：
1）数据挖掘
2）有些机器应用不能通过手工编程来实现（无人机自驾、手写识别、自然语言处理，即NLP、计算机视觉）
3）用户自定制化程序

3.定义（Tom Mitchell，1998）：
一个程序被认为能从经验E中学习，解决任务T达到性能度量值P，当且仅当有了经验E后，经过P评判，程序在处理T时的性能有所提升。

4.监督学习（Supervised Learing）：
（教计算机如何去完成任务）监督学习又称回归问题（Regression Problem），意指要预测一个连续值的输出，要预测这类连续值属性的类别

5.无监督学习（Unsupervised Learing）：
让计算机自己进行学习，其中一种是聚类算法，；另一种是
给算法输入大量的数据，要求它找出数据中蕴含的类型结构

6.支持向量机算法：
存在一个简洁的数学方法能让电脑处理无限多的特征

7.编程软件：
Octave，开源免费

线性回归

2019年5月11日 23:43

1.线性回归原理：

最小化问题，是的假设函数的系数最小，同时使得预测值与真实值误差最小，即它们的差的平方和最小

$h_\theta(x) = \theta_0 + \theta_1 x$

2.代价函数（costfunction）：

称为平方误差函数，亦为平方误差代价函数

$$J(\theta_0, \theta_1) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$$

$$\min_{\theta_0\ \theta_1} J(\theta_0, \theta_1)$$

3.参数评估指标

后续补充P10 2-5

梯度下降

2019年5月12日 9:21

1.梯度下降（Gradient descent）

此算法可将代价函数J最小化

有函数J(θ_0,θ_1)，求

$\min_{\theta_0\ \theta_1} \mathrm{J}(\theta_0,\theta_1)$

2.Outline

·对θ_0,θ_1初步假设（通常均假设为0）

·不断地改变θ_0,θ_1，去减小$J(\theta_0,\theta_1)$

·直到J最小（有可能是局部最小）

3.定义

:=赋值运算符

重复直至收敛{

$\theta_j := \theta_j - \alpha\frac{\partial}{\partial\theta_j}\mathrm{J}(\theta_0,\theta_1)\ \ \mathrm{for}(\mathrm{j}=0\ \mathrm{and}\ \mathrm{j}=1)$

}

learning rate: α 控制参数更新的步长（决定是大步下山还是小步下山）

更新θ_j

temp0 :=$\theta_0 - \alpha\frac{\partial}{\partial 0}\mathrm{J}(\theta_0,\theta_1)$

temp1 :=$\theta_1 - \alpha\frac{\partial}{\partial\theta_1}\mathrm{J}(\theta_0,\theta_1)$

θ_0 :=temp0

θ_1 :=temp1

同步更新，非同步更新并非指上述梯度下降算法

牛顿法

2019年5月12日 9:59

1.牛顿法也称为切线法，该方法使用函数f(x)的泰勒级数的前几项来寻找方程的根，其最大的优点在于收敛速度很快。

2.迭代公式：

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

3.优缺点

优点：二阶收敛，收敛速度快；

缺点：牛顿法是一种迭代算法，每一步都需要求解目标函数的Hessian矩阵的逆矩阵，计算比较复杂。

拟牛顿法

2019年5月12日 10:00

1.求解非线性优化问题最有效方法之一

拟牛顿法的本质思想是改善牛顿法每次需要求解复杂的Hessian矩阵的逆矩阵的缺陷，它使用正定矩阵来近似Hessian矩阵的逆，从而简化了运算的复杂度。

2.基本思想

·构造目标函数在当前迭代x_k的二次模型：

$$m_k(p) = f(x_k) + \nabla f(x_k)^T p + \frac{p^T B_k p}{2}$$

B_k是一个正定矩阵，于是取这个模型的最优解作为搜索方向，并且得到新的迭代点：

$$x_{k+1} = x_k + \alpha_k p_k$$

α_k满足Wolfe条件，这样的迭代与牛顿法类似，区别就在于用近似的Hesse矩阵B_k 代替真实的Hesse矩阵。所以拟牛顿法最关键的地方就是每一步迭代中矩阵Bk 的更新。现在假设得到一个新的迭代x_{k+1}，并得到一个新的二次模型：

$$m_{k+1}(p) = f(x_{k+1}) + \nabla f(x_{k+1})^T p + \frac{p^T B_{k+1} p}{2}$$

尽可能利用上一部的信息来选取B_k。具体地，我们要求：

$$\nabla f(x_{k+1}) - \nabla f(x_k) = \alpha_k B_{k+1} p_k$$

从而得到

$$B_{k+1}(x_{k+1} - x_k) = \nabla f(x_{k+1}) - \nabla f(x_k)$$

此公式被称为割线方程，常用的拟牛顿法有DFP算法和BFGS算法。

https://www.cnblogs.com/shixiangwan/p/7532830.html

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。