100-Days-Of-ML-Code

<Excerpt in index | 首页摘要>
这是一个由Youtuber Siraj Raval发起的机器学习挑战活动
LeN旨在号召大家每天至少花1个小时的时间在Machine Learning的学习上，内容涵盖了机器学习，深度学习等很多方面。
Website：100DaysOfMLCode
<The rest of contents | 余下全文>

Data PreProcessing | Day 1

第1步：导入库

```
import numpy as np
import pandas as pd
```

numpy：数学计算函数
pandas：用于导入和管理数据集

第4步：解析分类数据

```
from sklearn.preprocessing import LabelEncoder,OneHotEncoder
labelencoder_X = LabelEncoder()
X[ : , 0] = labelencoder_X.fit_transform(X[ : , 0])
```

LabelEncoder：对不连续的数字或文本编号。
One-Hot Encoding：采用位状态寄存器来对个状态进行编码

第五步： 拆分数据集分训练集和测试集：

```
from sklearn.model_selection import train_test_split
X_train,X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.2,
 random_state = 0)
```

train_test_split：随机划分训练集和测试集
`X_train,X_test, y_train, y_test =train_test_split(train_data,train_target,test_size=0.4, random_state=0)`

- **train_data**：所要划分的样本特征集
- **train_target**：所要划分的样本结果
- **test_size**：样本占比，如果是整数的话就是样本的数量
- **random_state**：是随机数的种子。

第6步：特征量化：

```
from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.fit_transform(X_test)
```

StandardScaler：数据在前处理的时候，经常会涉及到数据标准化。将现有的数据通过某种关系，映射到某一空间内。常用的标准化方式是,减去平均值，然后通过标准差映射到均至为0的空间内。

Simple Linear Regression | Day 2

线性回归

给定数据集 $D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)\}$，其中 $x_i = (x_{i1}; x_{i2}; \ldots; x_{id})$，$y_i \in \mathbb{R}$. “线性回归”(linear regression)试图学得一个线性模型以尽可能准确地预测实值输出标记.

公式化：

$$f(x_i) = wx_i + b, \text{ 使得 } f(x_i) \simeq y_i .$$

训练模型并预测：

```
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor = regressor.fit(X_train,Y_train)
Y_pred = regressor.predict(X_test)
```

训练集可视化：

```
import matplotlib.pyplot as plt
plt.scatter(X_train,Y_train,color = "red")
```

scatter:画散点图

```
import matplotlib.pyplot as plt
plt.plot(X_train, Y_train, color = 'blue')
```

plot:画连线图

结果:

Multiple Linear Regression | Day 3

当y值的影响因素不唯一时,采用多元线性回归模型。

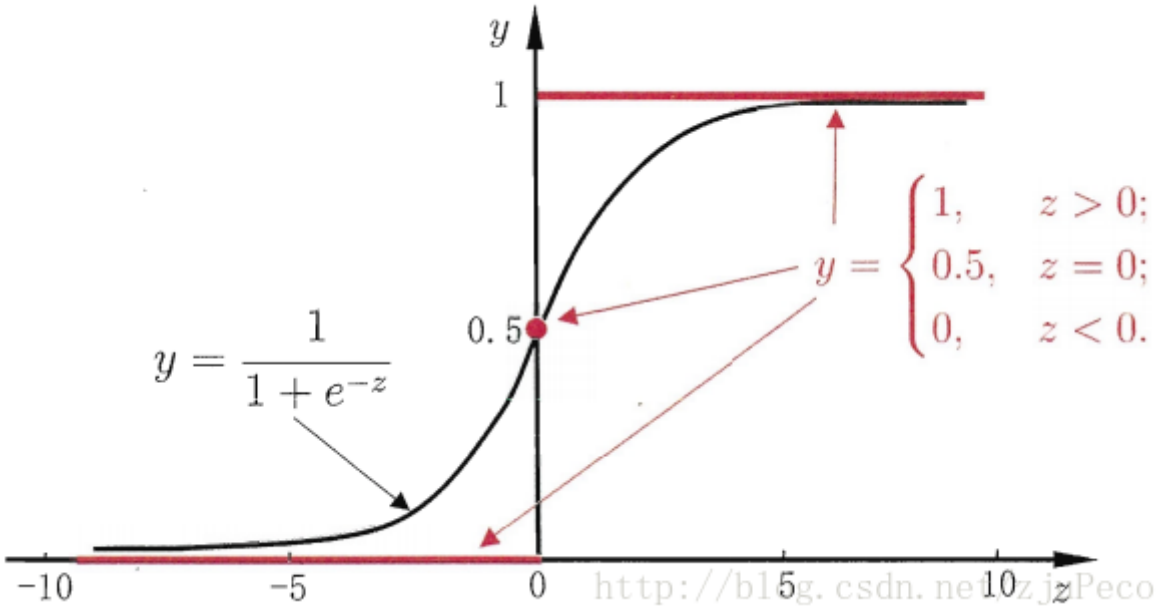
y =y=β0+β1x1+β2x2+…+βnxn

具体操作参考Siraj Raval

Logistic Regression | Day 4

逻辑回归（Logistic Regression）是用于处理因变量为分类变量的回归问题，常见的是二分类或二项分布问题，也可以处理多分类问题，它实际上是属于一种分类方法。

Sigmoid function



建立模型

```
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression()
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)
```

模型可视化

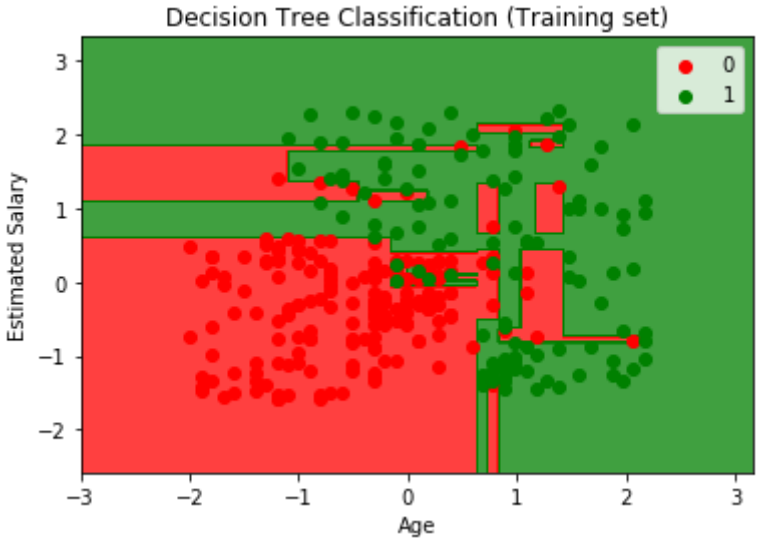
```
from matplotlib.colors import ListedColormap
X_set,y_set=X_train,y_train
X1,X2=np. meshgrid(np. arange(start=X_set[:,0].min()-1, stop=X_set[:,
0].max()+1, step=0.01),
                   np. arange(start=X_set[:,1].min()-1, stop=X_set[:,1].m
ax()+1, step=0.01))
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(),X2.ravel
()]).T).reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('red', 'green')))
plt.xlim(X1.min(),X1.max())
plt.ylim(X2.min(),X2.max())
for i,j in enumerate(np. unique(y_set)):
    plt.scatter(X_set[y_set==j,0],X_set[y_set==j,1],
                c = ListedColormap(('red', 'green'))(i), label=j)
```

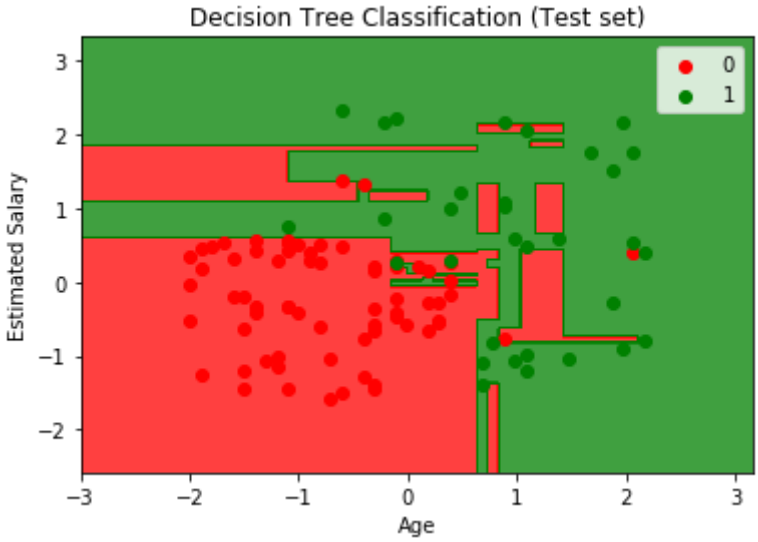
```
plt. title(' LOGISTIC(Training set)')
plt. xlabel(' Age')
plt. ylabel(' Estimated Salary')
plt. legend()
plt. show()

X_set,y_set=X_test,y_test
X1,X2=np. meshgrid(np. arange(start=X_set[:,0].min()-1, stop=X_set[:,
0].max()+1, step=0.01),
                   np. arange(start=X_set[:,1].min()-1, stop=X_set[:,1].m
ax()+1, step=0.01))

plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(),X2.ravel
()]).T).reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('red', 'green')))
plt.xlim(X1.min(),X1.max())
plt.ylim(X2.min(),X2.max())
for i,j in enumerate(np. unique(y_set)):
    plt.scatter(X_set[y_set==j,0],X_set[y_set==j,1],
                c = ListedColormap(('red', 'green'))(i), label=j)

plt. title(' LOGISTIC(Test set)')
plt. xlabel(' Age')
plt. ylabel(' Estimated Salary')
plt. legend()
plt. show()
```





K Nearest Neighbours | Day 5

KNN是通过测量不同特征值之间的距离进行分类。

这里计算距离一般使用欧氏距离或曼哈顿距离：

$$\text{欧式距离：}\ d(x,y)=\sqrt{\sum_{k=1}^{n}(x_k-y_k)^2}\ ,\qquad \text{曼哈顿距离：}\ d(x,y)=\sqrt{\sum_{k=1}^{n}|x_k-y_k|}$$

算法的描述为：

- 1）计算测试数据与各个训练数据之间的距离；
- 2）按照距离的递增关系进行排序；
- 3）选取距离最小的K个点；
- 4）确定前K个点所在类别的出现频率；
- 5）返回前K个点中出现频率最高的类别作为测试数据的预测分类。

使用K-NN对训练集进行训练并预测

```
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors=5,metric = 'minkowski', p
=2)
classifier.fit(X_train , Y_train)
y_pred = classifier.predict(X_test)
```

Support Vector Machines | Day 6

SVM是一个有监督学习的机器学习算法，可用于分类和回归，主要用于分类问题。寻找一个**超平面**，将数据分成两类。

对SVM来说，**最佳超平面**是指距离两类数据最远的一个超平面，即此超平面到最近的元素的距离最远。

特征量化：

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.fit_transform(X_test)
```

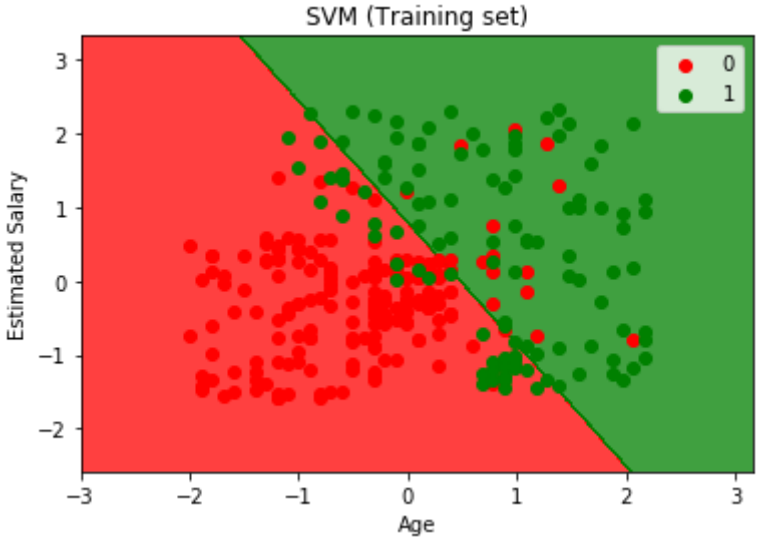
fit_transform()函数：先拟合数据，然后转化它将其转化为标准形式

transform()函数：通过找中心和缩放等实现标准化

两者的差别在于前者多了一个fit数据的步骤

使用SVM对训练集进行训练并预测

```
from sklearn.svm import SVC
classifier = SVC(kernel= 'linear' , random_state= 0 )
classifier.fit(X_train , y_train)
y_pred = classifier.predict(X_test)
```



Naive Bayes Classifier and Black Box Machine Learning | Day 7

1、朴素贝叶斯分类器

贝叶斯公式：

$$P(B|A) = \frac{P(A|B)P(B)}{P(A)}$$

也即是：

$$p(\text{类别}|\text{特征}) = \frac{p(\text{特征}|\text{类别})p(\text{类别})}{p(\text{特征})}$$

优点：

（1） 算法逻辑简单,易于实现（算法思路很简单，只要使用贝叶斯公式转化即可！）

（2）分类过程中时空开销小（假设特征相互独立，只会涉及到二维存储）

缺点：

理论上，朴素贝叶斯模型与其他分类方法相比具有最小的误差率。但是实际上并非总是如此，这是因为朴素贝叶斯模型假设属性之间相互独立，这个假设在实际应用中往往是不成立的，在属性个数比较多或者属性之间相关性较大时，分类效果不好。

而在属性相关性较小时，朴素贝叶斯性能最为良好。对于这一点，有半朴素贝叶斯之类的算法通过考虑部分关联性适度改进。

Decision Trees | Day 8

决策树（decision tree）：在分类问题中，表示基于特征对实例进行分类的过程，可以认为是if-then的集合，也可以认为是定义在特征空间与类空间上的条件概率分布。

决策树学习的算法通常是一个递归地选择最优特征，并根据该特征对训练数据进行分割，使得各个子数据集有一个最好的分类的过程。

决策树的特点：

- 优点：计算复杂度不高，输出结果易于理解，对中间值的缺失不敏感，可以处理不相关特征数据。
- 缺点：可能会产生过度匹配的问题
- 适用数据类型：数值型和标称型

信息增益

在划分数据集之前之后信息发生的变化成为信息增益，知道如何计算信息增益，我们就可以计算每个特征值划分数据集获得的信息增益，获得信息增益最高的特征就是最好的选择。

信息熵：

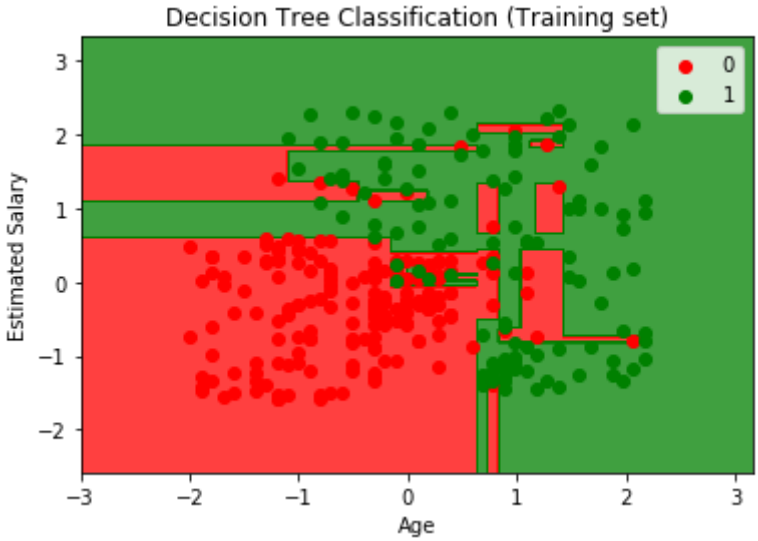
$$H = -\Sigma_{i=1}^{n} p(x_i) log_2 p(x_i)$$

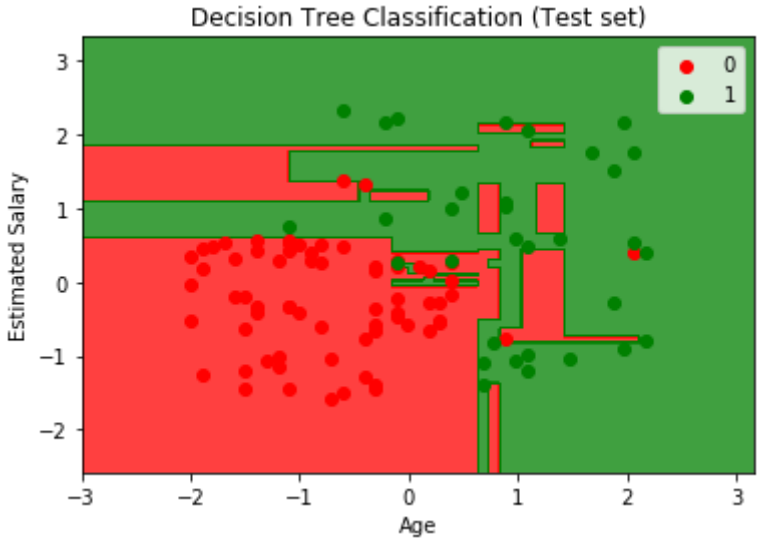
其中，$p(x_i)$是选择该分类的概率。n为分类数目，熵越大，随机变量的不确定性就越大。

使用decision tree对训练集进行训练并预测

```
from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier(criterion = 'entropy', random_state =
 0)
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)
```

结果：





Linear Algebra | Day 9

向量的本质：

1. 线性代数的本质：**向量是什么**
2. 线性代数的本质：**线性组合、张量的空间与基**
3. 线性代数的本质：**矩阵与线性变换**
4. 线性代数的本质：**矩阵乘法与线性变换复合**
5. 线性代数的本质：**三维空间中的线性变换**
6. 线性代数的本质：**行列式**
7. 线性代数的本质：**逆矩阵、列空间和零空间**
8. 线性代数的本质：**点积和对偶性**
9. 线性代数的本质：**叉积的标准介绍**
10. 线性代数的本质：**以线性变换的眼光看叉积**
11. 线性代数的本质：**基变换**
12. 线性代数的本质：**特征向量与特征值**
13. 线性代数的本质：**抽象的向量空间**

Essence of calculus | Day 10

微积分的本质：

1. 微积分的本质：**导数的悖论**
2. 微积分的本质：**用几何来求导**
3. 微积分的本质：**直观理解链式法则和乘积法则**
4. 微积分的本质：**指数函数求导**
5. 微积分的本质：**隐函数求导**

6. 微积分的本质：**极限**
7. 微积分的本质：**积分与微积分基本定理**
8. 微积分的本质：**面积与斜率有什么联系**
9. 微积分的本质：**高阶导数**
10. 微积分的本质：**泰勒级数**

Random Forests | Day 11

Bagging和Boosting的概念与区别

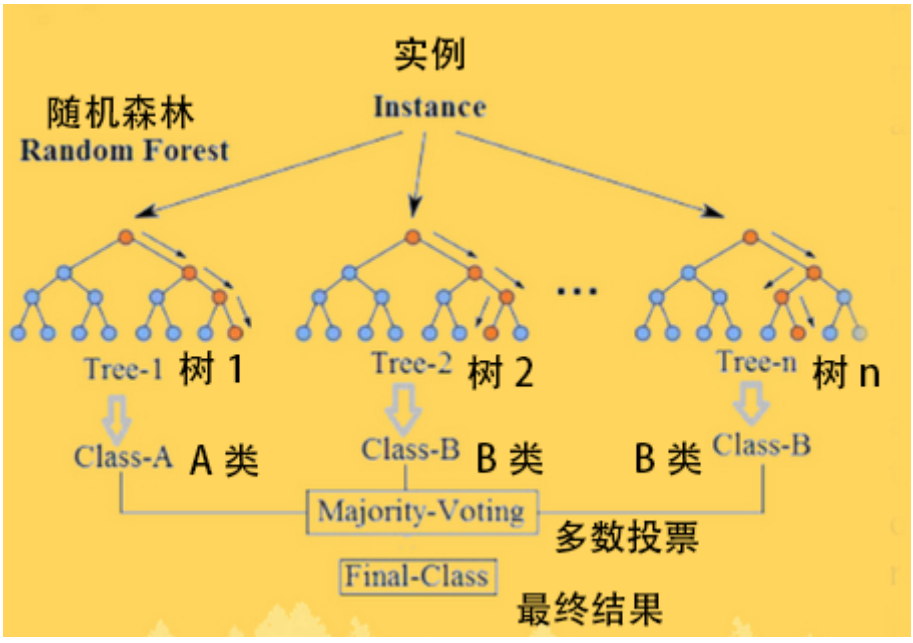
随机森林属于集成学习（Ensemble Learning）中的**bagging**算法。在集成学习中，主要分为bagging算法和boosting算法。

Bagging算法：

1. 从原始样本集中使用Bootstraping方法随机抽取n个训练样本，**共进行k轮抽取，得到k个训练集。**（k个训练集之间相互独立，元素可以有重复）.
2. 对于k个训练集，我们**训练k个模型**（这k个模型可以根据具体问题而定，比如决策树，knn等）.
3. 对于分类问题：由**投票表决产生分类结果**；对于回归问题：由k个模型预测结果的**均值**作为最后预测结果。（所有模型的重要性相同.

Boosting算法：

1. 对于训练集中的每个样本建立权值wi，表示对每个样本的关注度。**当某个样本被误分类的概率很高时，需要加大对该样本的权值。**
2. 进行迭代的过程中，每一步迭代都是一个弱分类器。我们需要用某种策略将其组合，作为最终模型。（例如AdaBoost给每个弱分类器一个权值，将其线性组合最为最终分类器。**误差越小的弱分类器，权值越大**）

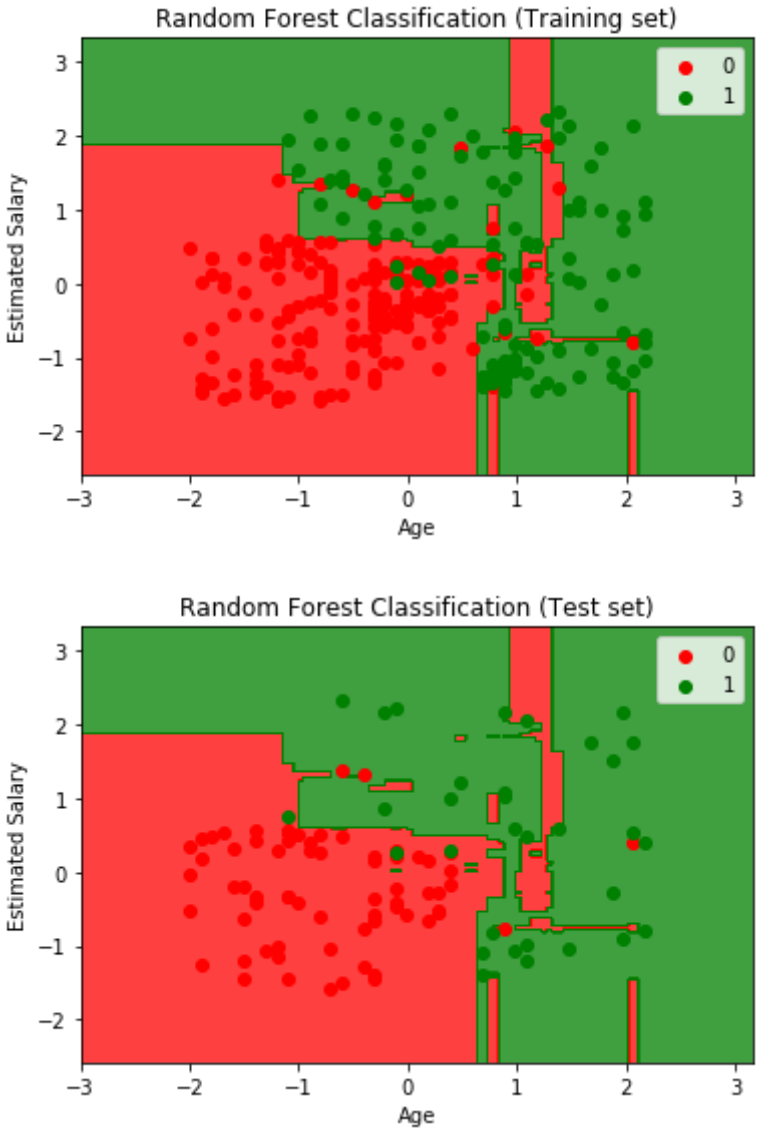


集成学习模型复合了多个机器学习模型，使得整体性能更好。在随机森林中，单个决策树做出的决策都是**“弱”因素**，而大量决策树在一起使用使他们的结果**整合在一起，就会产生“强”合奏。**

使用random forests对训练集进行训练并预测

```
from sklearn.ensemble import RandomForestClassifier
classifier = RandomForestClassifier(n_estimators = 10, criterion = 'entro
py', random_state = 0)
classifier.fit(X_train, y_train)
```

结果：

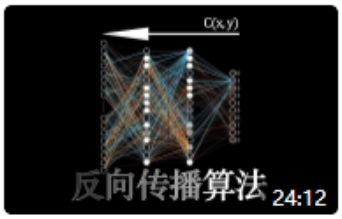


What is a Neural Network | Day 11

深度学习 Deep Learning

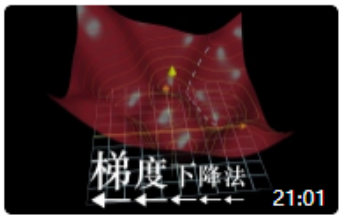
TA的频道 > 深度学习 Deep Learning

3个视频 | 2017-11-8更新



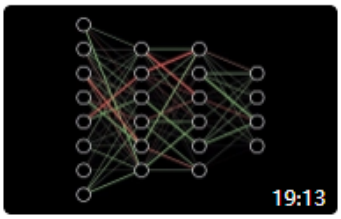
【官方双语】深度学习之反向传播算法 上/下 Part 3 ver 0.9 beta

5.6万 2017-11-22



【官方双语】深度学习之梯度下降法 Part 2 ver 0.9 beta

5.5万 2017-11-8



【官方双语】深度学习之神经网络的结构 Part 1 ver 2.0

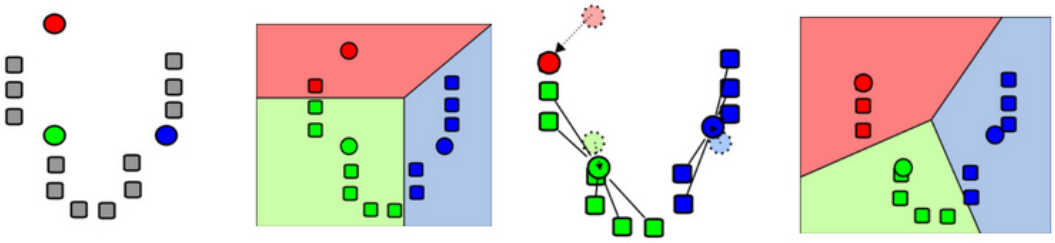
11.5万 2017-10-19

K Means Clustering | Day 12

Relevant blog:

种聚类算法（原理+代码+对比分析）最全总结

4) K-均值聚类如何工作



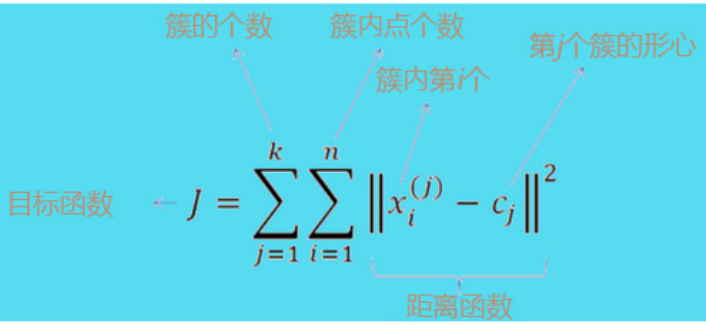
1. 在数据域中随机生成k个初始“均值”（本例中k=3）。

2. 通过关联每个观测值到最近的均值，创建k个簇。

3. 每个簇的形心变成新的均值。

4. 重复步骤2和步骤3，直到收敛。

K-均值聚类的目标是使总体群内方差最小，或平方误差函数：



Digging Deeper | NUMPY | Day 13

2 NumPy入门
2.1 理解Python中的数据类型
2.2 NumPy数组基础
2.3 NumPy数组的计算：通用函数

Digging Deeper | NUMPY | Day 14

2.4 聚合：最小值、最大值和其他值
2.5 数组的计算：广播
2.6 比较、掩码和布尔运算

Digging Deeper | NUMPY | Day 15

19

2.4 聚合：最小值、最大值和其他值
2.5 数组的计算：广播
2.6 比较、掩码和布尔运算

Digging Deeper | NUMPY | Day 16

2.7 花哨的索引
2.8 数组的排序
2.9 结构化数据：NumPy的结构化数组

Digging Deeper | PANDAS| Day 17

3 Pandas数据处理
3.1 Pandas对象简介
3.2 数据取值与选择
3.3 Pandas数值运算方法
3.4 处理缺失值
3.5 层级索引
3.6 合并数据集：ConCat和Append方法

Digging Deeper | PANDAS | Day 18

3.7 合并数据集：合并与连接
3.8 累计与分组
3.9 数据透视表

Digging Deeper | MATPLOTLIB | Day 19

4 Matplotlib数据可视化
4.1 简易线形图
4.2 简易散点图
4.3 可视化异常处理
4.4 密度图与等高线图

Digging Deeper | MATPLOTLIB | Day 20

4.5 直方图
4.6 配置图例
4.7 配置颜色条
4.8 多子图
4.9 文字与注释

反馈与建议

- 微博：@柏林designer
- 邮箱：wwj123@zju.edu.cn