机器学习综述

主讲人：王云申 技术部

总结人：刘梓艺 宣传部

1 什么是机器学习

1.1 对机器学习的一些解释

1、机器学习是这样的一个研究领域，它能让计算机不依赖确定的编码指令来自主的学习工作。 ——Arthur Samuel（亚瑟·塞缪尔），1959

2、对于某类任务T和性能度量P，如果一个计算机程序在T上以P衡量的性能随着经验E而自我完善，那么我们称这个计算机程序在从经验E学习。——Tom M. Mitchell，1998

（很多情况下，我们用准确度、正确率来作为P，衡量性能）

3、机器学习可以说是计算机使用输入给它的数据，利用人类赋予它的算法得到某种模型的过程，其最终的目的则是使用该模型，预测未来未知数据的信息。——《python与机器学习实战》

1.2 人工智能、机器学习、深度学习的关系

1、人工智能是计算机科学的一个分支，它企图了解智能的实质，并生产出一种新的能以人类智能相似的方式做出反应的智能机器。

2、机器学习研究如何使计算机系统利用经验改善性能。它是人工智能领域的分支，也是实现人工智能的一种手段。在机器学习的众多研究方向中，表征学习关注如何自动找出表示数据的合适方式，以便更好地将输入变换为正确的输出。

3、而深度学习是具有多级表示的表征学习方法（实现方式为改良的神经网络例如：卷积神经网络（Convolutional neural networks，CNN）、深度置信网络（Deep Belief Nets，DBN）等）。在每一级（从原始数据开始），深度学习通过简单的函数将该级的表示变换为更高级的表示。因此，深度学习模型也可以看作是由许多简单函数复合而成的函数。当这些复合的函数足够多时，深度学习模型就可以表达非常复杂的变换，即深度学习可以逐级表示越来越抽象的概念或模式。

1.3 机器学习术语

1、“数据集”(Data Set）：其中，每一条单独的数据被称为“样本”（ Sample ）。数据集”也常被分为以下三类：训练集（ Training Set）、测试集（ Test Set）、验证集（ Validation set）：用来调整模型具体参数

2、每个样本，它通常具有一些“属性”（ Attribute）或者说“特征” （ Feature ）,特征所具体取的值就被称为“特征值” (Feature Value ）

3、特征和样本所张成的空间被称为“特征空间” （Feature Space）和“样本空间”（ Sample Space ），可以把它们简单地理解为特征和样本“可能存在的空间”

4、“标签空间”（ Label Space），它描述了模型的输出“可能存在的空间”：当模型是分类器时，我们通常会称之为“类别空间” 。

2 机器学习的常用模型

2.1 模型分类

1、监督学习：训练样本数据有对应的目标值，通过对数据样本因子和已知的结果建立联系，提取特征值和映射关系，通过已知的结果，已知数据样本不断的学习和训练，对新的数据进行结果的预测。

常见应用：K近邻算法、线性回归、logistic回归、支持向量机（SVM）、决策树和随机森林、神经网络等

2、无监督学习：利用无标签的数据学习数据的分布或者数据之间的关系

常见应用：聚类算法、主成分分析、关联规则学习

（对于聚类的解释：聚类就是将一堆零散的数据根据某些标准分为几个类别，一般来说最常使用的标准是距离，距离也分为好几类，比如欧式距离（空间中两点的直线距离）、曼哈顿距离（城市街区距离）、马氏距离（数据的协方差距离）和夹角余弦）

3、半监督学习：监督学习和无监督学习相互结合的一种学习方法

常见应用：照片识别（少量标志与大量无标志训练集）

2.2 常见模型

1、贝叶斯分类器

贝叶斯分类器是一类分类算法的总称，这类算法均以**贝叶斯定理**为基础，故统称为贝叶斯分类器。在许多场合，朴素贝叶斯(NaïveBayes，NB)分类算法可以与决策树和神经网络分类算法相媲美，该算法能运用到大型数据中，而且方法简单、分类准确率高、速度快。但由于其假设**每个属性独立地对分类结果发生影响**，当影响因素之间存在强关系时可能结果不好。

基本原理：

（1）贝叶斯定理：

$p(\theta \mid X)=\frac{p(\theta)p(X|\theta)}{p(X)}$

（2）极大似然概率：已知属性$x_1,x_2,\ldots x_n$，从先验概率的角度，去分析最有可能的类型θ

利用每个属性独立地对分类结果发生影响：

$p(X;\theta)\equiv p\left(x_1,x_2,\ldots,x_N;\theta\right)=\prod_{k=1}^{N}p\left(x_k;\theta\right)$

求θ变化时，概率最大时的θ：

$\hat{\theta}_{ML}=\arg\max_{\theta}\prod_{k=1}^{N}p\left(x_k;\theta\right)$

（3）最大后验估计：

利用贝叶斯定理：

$p(\theta \mid D)=\frac{p(D|\theta)p(\theta)}{p(D)}=\frac{p(D|\theta)p(\theta)}{\int p(D|\theta)p(\theta)d\theta}$

$p(D \mid \theta)=\prod_{k=1}^{N}p\left(x_k \mid \theta\right)$

求θ变化时，概率最大时的θ：

$\hat{\theta}_{MAP}=\arg\max_{\theta}p(\theta \mid X)=\arg\max_{\theta}p(\theta)p(X \mid \theta)$

2、贝叶斯线性回归

线性回归中由于极大似然估计总是会使得模型过于的复杂以至于产生过拟合的现象，所以单纯的适用极大似然估计并不是特别的有效。贝叶斯线性回归不仅可以解决极大似然估计中存在的过拟合的问题，而且，它对数据样本的利用率是100%，仅仅使用训练样本就可以有效而准确的确定模型的复杂度。

贝叶斯线性回归：

$$\ln p(\theta \mid D) = \ln p(w \mid T) = -\frac{\beta}{2}\sum_{n=1}^{N}\{y(x_n, w) - t_n\}^2 + \frac{\alpha}{2}w^T w + const$$

β、α分别为样本集合和参数w的高斯分布方差。

3、决策树

决策树是一种树形结构，其中每个内部节点表示一个属性上的判断，每个分支代表一个判断结果的输出，最后每个叶节点代表一种分类结果。

目前常用的决策树生成算法有：

1. ID3 算法可说是“最朴素”的决策树算法，它给出了 对离散型数据分类的解决方案。
2. C4.5 算法在其基础上进一步发展，给出了对混合型数据分类的解决方案。
3. CART 算法则更进一步，给出了对数据回归的解决方案。

其核心思想一致：通过不断划分数据集来生成决策树，其中每一步的划分能够使当前的信息增益达到最大，即划分后系统信息熵降最大。

4、支持向量机

支持向量机（support vector machines, SVM）是一种二分类模型，它的基本模型是定义在特征空间上的间隔最大的线性分类器，SVM还包括核技巧，这使它成为实质上的非线性分类器。

SVM学习的基本想法是求解能够正确划分训练数据集并且几何间隔最大的分离超平面。如下图所示， w*x+b=0 即为分离了超平面，对于线性可分的数据集来说，这样的超平面有无穷多个（即感知机），但是几何间隔最大的分离超平面却是唯一的。

3 实例应用

现有去年工厂生产的一批白葡萄酒和红葡萄酒的十二项相关数据。而今年的白葡萄酒和红葡萄酒被实习工搞混在了一起，需要你根据每瓶酒的参数判断它是白葡萄酒还是红葡萄酒。

H2

```
# 葡萄酒问题
import pandas as pd
from sklearn import model_selection
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree
```

```
import graphviz

# 读入数据
t = pd.read_csv("./winequality.csv")
t.describe()
# 选择自变量、因变量
xs =
t[['x1','x2','x3','x4','x5','x6','x7','x8','x9','x10','x11','y
']]
y = t['class']

# 划分数据集
xs_train, xs_test, y_train, y_test =
model_selection.train_test_split(xs, y, random_state=0,
test_size=0.1,shuffle=True)
print(xs_train.shape)
print(xs_test.shape)
print(y_train.shape)
print(y_test.shape)
# train_data：待划分的样本数据
# train_target：待划分的对应样本数据的样本标签
# test_size：1）浮点数，在0 ~ 1之间，表示样本占比（test_size =
0.3，则样本数据中有30%的数据作为测试数据，记入X_test，其余70%数据记
入X_train，同时适用于样本标签）；2）整数，表示样本数据中有多少数据记
入X_test中，其余数据记入X_train
# random_state：随机数种子，种子不同，每次采的样本不一样；种子相
同，采的样本不变（random_state不取，采样数据不同，但random_state等
于某个值，采样数据相同，取0的时候也相同，这可以自己编程尝试下，不过想
改变数值也可以设置random_state = int(time.time())）
# shuffle：洗牌模式，1）shuffle = False，不打乱样本数据顺序；2）
shuffle = True，打乱样本数据顺序

#构建决策树
est = DecisionTreeClassifier(max_depth = 5)
est.fit(xs_train, y_train)
yhat = est.predict(xs_test)
# 预测类型
print(yhat)
# 模型正确率得分
print(est.score(xs_test, y_test))
#特征重要性
print(est.feature_importances_)
# DecisionTreeClassifier参数解释
# https://blog.csdn.net/qq_16000815/article/details/80954039
```

```
(5847, 12)
```

```
(650, 12)
(5847,)
(650,)
[0 0 1 1 1 1 0 0 1 0 1 1 1 1 1 1 1 1 0 1 0 0 1 1 1 1 1 1 1 1 1 1 1 0 1 0 1
 0 0 0 1 1 1 1 0 1 1 1 1 1 1 1 0 1 1 1 1 0 1 1 1 0 1 1 0 1 1 1 0 1 1 1 1 1
 1 1 1 0 0 1 1 1 1 1 1 1 1 0 1 1 1 1 0 0 1 1 1 1 1 1 1 0 1 1 0 1 1 0 1 1 0
 1 1 1 1 0 1 1 0 1 1 1 0 1 1 1 1 1 1 0 1 1 1 1 1 0 0 0 0 1 1 1 1 1 0 0 0 0
 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1 0 0 1 1 1 1 0 0 1 0 1
 1 1 0 1 1 1 1 1 1 1 1 0 1 1 1 1 0 1 1 1 0 1 1 1 1 1 0 1 1 1 1 1 1 1 1 0 1 0
 1 0 1 0 1 1 0 1 1 1 1 1 1 1 1 0 1 1 0 1 1 1 1 1 0 0 1 0 1 1 1 1 1 1 0 1 0 1
 1 1 1 0 1 1 1 0 1 1 1 1 0 1 0 1 0 0 1 0 0 1 1 0 1 1 1 1 0 1 1 1 1 1 1 1 1 1
 1 0 1 1 1 0 1 1 1 0 0 1 1 0 0 1 1 0 0 1 1 1 1 0 1 1 0 1 0 1 0 1 1 1 1 0 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 0 0 1 0 1 1 0 1 1 1 1 1 1 1 1 0
 1 0 1 1 1 0 0 1 1 0 1 1 1 1 1 1 0 0 1 1 0 0 1 1 1 1 0 1 1 1 1 1 1 1 1 0 0 1
 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 0 1 1 1 1 1 1 0 0 0 1 1 0 1 1 1 0
 1 1 1 1 0 1 1 1 1 0 1 1 1 0 1 1 0 1 0 1 1 1 1 1 1 0 1 1 0 1 1 1 1 1 1 1 1 0
 0 0 0 1 1 1 0 1 1 1 1 0 1 1 1 1 1 0 1 1 1 1 1 0 0 1 1 0 1 1 0 0 1 0 1 1 1 0
 1 0 0 1 1 1 0 0 1 1 0 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 0 1 1 1 0 1
 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1
 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 0 0 0 1 0 1 1 1 1 0 0 0 1 1 1 1 1
 1 0 1 1 0 0 1 0 1 0 1 1 1 0 0 1 1 1 0 0 0]
0.9830769230769231
[0.00378593 0.05131628 0.         0.00096482 0.20885934 0.
 0.70503636 0.01618079 0.00292311 0.01093336 0.         0.
   ]
```