

<<机器学习实战>>

图书基本信息

书名：<<机器学习实战>>

13位ISBN编号：9787115317957

10位ISBN编号：711531795X

出版时间：2013-6

出版时间：人民邮电出版社

作者：Peter Harrington

译者：李锐,李鹏,曲亚东,王斌

版权说明：本站所提供下载的PDF图书仅提供预览和简介，请支持正版图书。

更多资源请访问：<http://www.tushu007.com>

<<机器学习实战>>

内容概要

机器学习是人工智能研究领域中的一个极其重要的研究方向，在现今的大数据时代背景下，捕获数据并从中萃取有价值的信息或模式，成为各行业求生存、谋发展的决定性手段，这使得这一过去为分析师和数学家所专属的研究领域越来越为人们所瞩目。

本书第一部分主要介绍机器学习基础，以及如何利用算法进行分类，并逐步介绍了多种经典的监督学习算法，如k近邻算法、朴素贝叶斯算法、Logistic回归算法、支持向量机、AdaBoost集成方法、基于树的回归算法和分类回归树（CART）算法等。

第三部分则重点介绍无监督学习及其一些主要算法：k均值聚类算法、Apriori算法、FP-Growth算法。

第四部分介绍了机器学习算法的一些附属工具。

全书通过精心编排的实例，切入日常工作任务，摒弃学术化语言，利用高效的可复用Python代码来阐释如何处理统计数据，进行数据分析及可视化。

通过各种实例，读者可从中学会机器学习的核心算法，并能将其运用于一些策略性任务中，如分类、预测、推荐。

另外，还可用它们来实现一些更高级的功能，如汇总和简化等。

<<机器学习实战>>

作者简介

Peter Harrington 拥有电气工程学士和硕士学位，他曾经在美国加州和中国的英特尔公司工作7年。Peter拥有5项美国专利，在三种学术期刊上发表过文章。他现在是Zillabyte公司的首席科学家，在加入该公司之前，他曾担任2年的机器学习软件顾问。Peter在业余时间还参加编程竞赛和建造3D打印机。

<<机器学习实战>>

书籍目录

第一部分 分类 第1章 机器学习基础 2 1.1 何谓机器学习 3 1.1.1 传感器和海量数据 4 1.1.2 机器学习非常重要 5 1.2 关键术语 5 1.3 机器学习的主要任务 7 1.4 如何选择合适的算法 8 1.5 开发机器学习应用程序的步骤 9 1.6 Python语言的优势 10 1.6.1 可执行伪代码 10 1.6.2 Python比较流行 10 1.6.3 Python语言的特色 11 1.6.4 Python语言的缺点 11 1.7 NumPy函数库基础 12 1.8 本章小结 13 第2章 k—近邻算法 15 2.1 k—近邻算法概述 15 2.1.1 准备：使用Python导入数据 17 2.1.2 从文本文件中解析数据 19 2.1.3 如何测试分类器 20 2.2 示例：使用k—近邻算法改进约会网站的配对效果 20 2.2.1 准备数据：从文本文件中解析数据 21 2.2.2 分析数据：使用Matplotlib创建散点图 23 2.2.3 准备数据：归一化数值 25 2.2.4 测试算法：作为完整程序验证分类器 26 2.2.5 使用算法：构建完整可用系统 27 2.3 示例：手写识别系统 28 2.3.1 准备数据：将图像转换为测试向量 29 2.3.2 测试算法：使用k—近邻算法识别手写数字 30 2.4 本章小结 31 第3章 决策树 32 3.1 决策树的构造 33 3.1.1 信息增益 35 3.1.2 划分数据集 37 3.1.3 递归构建决策树 39 3.2 在Python中使用Matplotlib注解绘制树形图 42 3.2.1 Matplotlib注解 43 3.2.2 构造注解树 44 3.3 测试和存储分类器 48 3.3.1 测试算法：使用决策树执行分类 49 3.3.2 使用算法：决策树的存储 50 3.4 示例：使用决策树预测隐形眼镜类型 50 3.5 本章小结 52 第4章 基于概率论的分类方法：朴素贝叶斯 53 4.1 基于贝叶斯决策理论的分类方法 53 4.2 条件概率 55 4.3 使用条件概率来分类 56 4.4 使用朴素贝叶斯进行文档分类 57 4.5 使用Python进行文本分类 58 4.5.1 准备数据：从文本中构建词向量 58 4.5.2 训练算法：从词向量计算概率 60 4.5.3 测试算法：根据现实情况修改分类器 62 4.5.4 准备数据：文档词袋模型 64 4.6 示例：使用朴素贝叶斯过滤垃圾邮件 64 4.6.1 准备数据：切分文本 65 4.6.2 测试算法：使用朴素贝叶斯进行交叉验证 66 4.7 示例：使用朴素贝叶斯分类器从个人广告中获取区域倾向 68 4.7.1 收集数据：导入RSS源 68 4.7.2 分析数据：显示地域相关的用词 71 4.8 本章小结 72 第5章 Logistic回归 73 5.1 基于Logistic回归和Sigmoid函数的分类 74 5.2 基于最优化方法的最佳回归系数确定 75 5.2.1 梯度上升法 75 5.2.2 训练算法：使用梯度上升找到最佳参数 77 5.2.3 分析数据：画出决策边界 79 5.2.4 训练算法：随机梯度上升 80 5.3 示例：从疝气病症预测病马的死亡率 85 5.3.1 准备数据：处理数据中的缺失值 85 5.3.2 测试算法：用Logistic回归进行分类 86 5.4 本章小结 88 第6章 支持向量机 89 6.1 基于最大间隔分隔数据 89 6.2 寻找最大间隔 91 6.2.1 分类器求解的优化问题 92 6.2.2 SVM应用的一般框架 93 6.3 SMO高效优化算法 94 6.3.1 Platt的SMO算法 94 6.3.2 应用简化版SMO算法处理小规模数据集 94 6.4 利用完整PlattSMO算法加速优化 99 6.5 在复杂数据上应用核函数 105 6.5.1 利用核函数将数据映射到高维空间 106 6.5.2 径向基核函数 106 6.5.3 在测试中使用核函数 108 6.6 示例：手写识别问题回顾 111 6.7 本章小结 113 第7章 利用AdaBoost元算法提高分类性能 115 7.1 基于数据集多重抽样的分类器 115 7.1.1 bagging：基于数据随机重抽样的分类器构建方法 116 7.1.2 boosting 116 7.2 训练算法：基于错误提升分类器的性能 117 7.3 基于单层决策树构建弱分类器 118 7.4 完整AdaBoost算法的实现 122 7.5 测试算法：基于AdaBoost的分类 124 7.6 示例：在一个难数据集上应用AdaBoost 125 7.7 非均衡分类问题 127 7.7.1 其他分类性能度量指标：正确率、召回率及ROC曲线 128 7.7.2 基于代价函数的分类器决策控制 131 7.7.3 处理非均衡数据的数据抽样方法 132 7.8 本章小结 132 第二部分 利用回归预测数值型数据 第8章 预测数值型数据：回归 136 8.1 用线性回归找到最佳拟合直线 136 8.2 局部加权线性回归 141 8.3 示例：预测鲍鱼的年龄 145 8.4 缩减系数来“理解”数据 146 8.4.1 岭回归 146 8.4.2 lasso 148 8.4.3 前向逐步回归 149 8.5 权衡偏差与方差 152 8.6 示例：预测乐高玩具套装的价格 153 8.6.1 收集数据：使用Google购物的API 153 8.6.2 训练算法：建立模型 155 8.7 本章小结 158 第9章 树回归 159 9.1 复杂数据的局部性建模 159 9.2 连续和离散型特征的树的构建 160 9.3 将CART算法用于回归 163 9.3.1 构建树 163 9.3.2 运行代码 165 9.4 树剪枝 167 9.4.1 预剪枝 167 9.4.2 后剪枝 168 9.5 模型树 170 9.6 示例：树回归与标准回归的比较 173 9.7 使用Python的Tkinter库创建GUI 176 9.7.1 用Tkinter创建GUI 177 9.7.2 集成Matplotlib和Tkinter 179 9.8 本章小结 182 第三部分 无监督学习 第10章 利用K—均值聚类算法对未标注数据分组 184 10.1 K—均值聚类算法 185 10.2 使用后处理来提高聚类性能 189 10.3 二分K—均值算法 190 10.4 示例：对地图上的点进行聚类 193 10.4.1 Yahoo! PlaceFinderAPI 194 10.4.2 对地理坐标进行聚类 196 10.5 本章小结 198 第11章 使用Apriori算法进行关联分析 200 11.1 关联分析 201 11.2 Apriori原理 202 11.3 使用Apriori算法来发现频繁集 204 11.3.1 生成候选项集 204 11.3.2 组织完整的Apriori算法 207 11.4 从频繁项集中挖掘关联规则 209 11.5 示例：发现国会投票中的

<<机器学习实战>>

模式 212 11.5.1 收集数据：构建美国国会投票记录的事务数据集 213 11.5.2 测试算法：基于美国国会投票记录挖掘关联规则 219 11.6 示例：发现毒蘑菇的相似特征 220 11.7 本章小结 221 第12章 使用FP—growth算法来高效发现频繁项集 223 12.1 FP树：用于编码数据集的有效方式 224 12.2 构建FP树 225 12.2.1 创建FP树的数据结构 226 12.2.2 构建FP树 227 12.3 从一棵FP树中挖掘频繁项集 231 12.3.1 抽取条件模式基 231 12.3.2 创建条件FP树 232 12.4 示例：在Twitter源中发现一些共现词 235 12.5 示例：从新闻网站点击流中挖掘 238 12.6 本章小结 239 第四部分 其他工具 第13章 利用PCA来简化数据 242 13.1 降维技术 242 13.2 PCA 243 13.2.1 移动坐标轴 243 13.2.2 在NumPy中实现PCA 246 13.3 示例：利用PCA对半导体制造数据降维 248 13.4 本章小结 251 第14章 利用SVD简化数据 252 14.1 SVD的应用 252 14.1.1 隐性语义索引 253 14.1.2 推荐系统 253 14.2 矩阵分解 254 14.3 利用Python实现SVD 255 14.4 基于协同过滤的推荐引擎 257 14.4.1 相似度计算 257 14.4.2 基于物品的相似度还是基于用户的相似度？ 260 14.4.3 推荐引擎的评价 260 14.5 示例：餐馆菜肴推荐引擎 260 14.5.1 推荐未尝过的菜肴 261 14.5.2 利用SVD提高推荐的效果 263 14.5.3 构建推荐引擎面临的挑战 265 14.6 基于SVD的图像压缩 266 14.7 本章小结 268 第15章 大数据与MapReduce 270 15.1 MapReduce：分布式计算的框架 271 15.2 Hadoop流 273 15.2.1 分布式计算均值和方差的mapper 273 15.2.2 分布式计算均值和方差的reducer 274 15.3 在Amazon网络服务上运行Hadoop程序 275 15.3.1 AWS上的可用服务 276 15.3.2 开启Amazon网络服务之旅 276 15.3.3 在EMR上运行Hadoop作业 278 15.4 MapReduce上的机器学习 282 15.5 在Python中使用mrjob来自动化MapReduce 283 15.5.1 mrjob与EMR的无缝集成 283 15.5.2 mrjob的一个MapReduce脚本剖析 284 15.6 示例：分布式SVM的Pegasos算法 286 15.6.1 Pegasos算法 287 15.6.2 训练算法：用mrjob实现MapReduce版本的SVM 288 15.7 你真的需要MapReduce吗？ 292 15.8 本章小结 292 附录A Python入门 294 附录B 线性代数 303 附录C 概率论复习 309 附录D 资源 312 索引 313 版权声明 316

<<机器学习实战>>

章节摘录

版权页： 插图： 7.1.1 bagging：基于数据随机重抽样的分类器构建方法 自举汇聚法（bootstrap aggregating），也称为bagging方法，是在从原始数据集选择 S 次后得到 S 个新数据集的一种技术。

新数据集和原数据集的大小相等。

每个数据集都是通过在原始数据集中随机选择一个样本来进行替换而得到的。

这里的替换就意味着可以多次地选择同一样本。

这一性质就允许新数据集中可以有重复的值，而原始数据集的某些值在新集合中则不再出现。

在 S 个数据集建好之后，将某个学习算法分别作用于每个数据集就得到了 S 个分类器。

当我们要对新数据进行分类时，就可以应用这 S 个分类器进行分类。

与此同时，选择分类器投票结果中最多的类别作为最后的分类结果。

当然，还有一些更先进的bagging方法，比如随机森林（random forest）。

有关这些方法的一个很好的讨论材料参见网页接下来我们将注意力转向一个与bagging类似的集成分类器方法boosting。

7.1.2 boosting boosting是一种与bagging很类似的技术。

不论是在boosting还是bagging当中，所使用的多个分类器的类型都是一致的。

但是在前者当中，不同的分类器是通过串行训练而获得的，每个新分类器都根据已训练出的分类器的性能来进行训练。

boosting是通过集中关注被已有分类器错分的那些数据来获得新的分类器。

由于boosting分类的结果是基于所有分类器的加权求和结果的，因此boosting与bagging不太一样。

bagging中的分类器权重是相等的，而boosting中的分类器权重并不相等，每个权重代表的是其对应分类器在上一轮迭代中的成功度。

boosting方法拥有多个版本，本章将只关注其中一个最流行的版本AdaBoost。

下面我们将要讨论AdaBoost背后的一些理论，并揭示其效果不错的原因。

7.2训练算法：基于错误提升分类器的性能 能否使用弱分类器和多个实例来构建一个强分类器？

这是一个非常有趣的理论问题。

这里的“弱”意味着分类器的性能比随机猜测要略好，但是也不会好太多。

这就是说，在二分类情况下弱分类器的错误率会高于50%，而“强”分类器的错误率将会低很多。

AdaBoost算法即脱胎于上述理论问题。

AdaBoost是adaptive boosting（自适应boosting）的缩写，其运行过程如下：训练数据中的每个样本，并赋予其一个权重，这些权重构成了向量 D 。

一开始，这些权重都初始化成相等值。

首先在训练数据上训练出一个弱分类器并计算该分类器的错误率，然后在同一数据集上再次训练弱分类器。

在分类器的第二次训练当中，将会重新调整每个样本的权重，其中第一次分对的样本的权重将会降低，而第一次分错的样本的权重将会提高。

为了从所有弱分类器中得到最终的分类结果，AdaBoost为每个分类器都分配了一个权重值 α ，这些 α 值是基于每个弱分类器的错误率进行计算的。

其中，错误率 ϵ 的定义为：而 α 的计算公式如下：AdaBoost算法的流程如图7—1所示。

<<机器学习实战>>

编辑推荐

《机器学习实战》面向日常任务的高效实战内容，介绍并实现机器学习的主流算法。

《机器学习实战》没有从理论角度来揭示机器学习算法背后的数学原理，而是通过“原理简述+问题实例+实际代码+运行效果”来介绍每一个算法。

学习计算机的人都知道，计算机是一门实践学科，没有真正实现运行，很难真正理解算法的精髓。

这本书的最大好处就是边学边用，非常适合于急需迈进机器学习领域的人员学习。

实际上，即使对于那些对机器学习有所了解的人来说，通过代码实现也能进一步加深对机器学习算法的理解。

《机器学习实战》的代码采用Python语言编写。

Python代码简单优雅、易于上手，科学计算软件包众多，已经成为不少大学和研究机构进行计算机教学和科学计算的语言。

相信Python编写的机器学习代码也能让读者尽快领略到这门学科的精妙之处。

版权说明

本站所提供下载的PDF图书仅提供预览和简介，请支持正版图书。

更多资源请访问:<http://www.tushu007.com>