1. 引言

也许你和这个叫『机器学习』的家伙一点也不熟，但是你举起iphone手机拍照的时候，早已习惯它帮你框出人脸；也自然而然点开今日头条推给你的新闻；也习惯逛淘宝点了找相似之后货比三家；亦或喜闻乐见微软的年识别网站结果刷爆朋友圈。恩，这些功能的核心算法就是机器学习领域的内容。

套用一下大神们对机器学习的定义，机器学习**研究的是计算机怎样模拟人类的学习行为，以获取新的知识或技能，并重新组织已有的知识结构使之不断改善自身**。简单一点说，就是计算机从数据中学习出规律和模式，以用在新数据上做预测的任务。近年来互联网数据大爆炸，数据的丰富度和覆盖面远远超出人工可以观察和总结范畴，而机器学习的算法能指引计算机在海量数据中，挖掘出有用的价值，也使得无数学习者为之着迷。

但是越说越觉得机器学习有距离感，云里雾里高深莫测，我们不是专家，但说起算有一些从业经验，做过一些目在实际数据上应用机器学习。这一篇就我们的经验和各位同仁的分享，总结一些对于初学者入门有帮助的方和对进阶有用的资料。

人工智能

2. 机器学习关注问题

并非所有的问题都适合用机器学习解决(很多逻辑清晰的问题用规则能很高效和准确地处理)，也没有一个机器习算法可以通用于所有问题。咱们先来了解了解，机器学习，到底关心和解决什么样的问题。

从功能的角度分类，机器学习在一定量级的数据上，可以解决下列问题：

1.分类问题

- 根据数据样本上抽取出的特征，判定其属于有限个类别中的哪一个。比如：
    - 垃圾邮件识别(结果类别：1、垃圾邮件 2、正常邮件)
    - 文本情感褒贬分析(结果类别：1、褒 2、贬)
    - 图像内容识别识别(结果类别：1、喵星人 2、汪星人 3、人类 4、草泥马 5、都不是)。

2.回归问题

- 根据数据样本上抽取出的特征，预测一个连续值的结果。比如：
    - 星爷《美人鱼》票房
    - 大帝都2个月后的房价
    - 隔壁熊孩子一天来你家几次，宠幸你多少玩具

3.聚类等问题

- 根据数据样本上抽取出的特征，让样本抱抱团(相近/相关的样本在一团内)。比如：
    - google的新闻分类
    - 用户群体划分

我们再把上述常见问题划到机器学习最典型的2个分类上。

- 分类与回归问题需要用已知结果的数据做训练，属于**“监督学习”**
- 聚类的问题不需要已知标签，属于**“非监督学习”**。

如果在IT行业(尤其是互联网)里溜达一圈，你会发现机器学习在以下热点问题中有广泛应用：

1.计算机视觉

- 典型的应用包括：人脸识别、车牌识别、扫描文字识别、图片内容识别、图片搜索等等。

2.自然语言处理

- 典型的应用包括：搜索引擎智能匹配、文本内容理解、文本情绪判断，语音识别、输入法、机器翻译等等。

3.社会网络分析

- 典型的应用包括：用户画像、网络关联分析、欺诈作弊发现、热点发现等等。

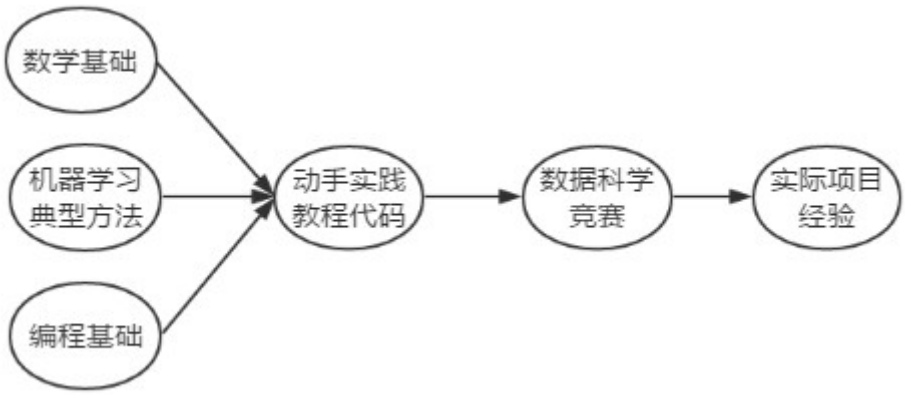
4.推荐

- 典型的应用包括：虾米音乐的“歌曲推荐”，某宝的“猜你喜欢”等等。

3. 入门方法与学习路径

OK，不废话，直接切重点丢干货了。看似学习难度大，曲线陡的机器学习，对大多数入门者也有一个比较通的学习路径，也有一些优秀的入门资料可以降低大家的学习门槛，同时激发我们的学习乐趣。

简单说来，大概的一个学习路径如下：



简单说一点，之所以最左边写了『数学基础』『典型机器学习算法』『编程基础』三个并行的部分，是因为机学习是一个将数学/算法理论和工程实践紧密结合的领域，需要扎实的理论基础帮助引导数据分析与模型调优时也需要精湛的工程开发能力去高效化地训练和部署模型和服务。

需要多说一句的是，在互联网领域从事机器学习的人，有2类背景的人比较多，其中一部分(很大一部分)是程序出身，这类同学工程经验相对会多一些，另一部分是学数学统计领域的同学，这部分同学理论基础相对扎实一些。因此对比上图，2类同学入门机器学习，所欠缺和需要加强的部分是不一样的。

下面就上述图中的部分，展开来分别扯几句：

3.1 数学基础

有无数激情满满大步向前，誓要在机器学习领域有一番作为的同学，在看到公式的一刻突然就觉得自己狗带了
是啊，机器学习之所以相对于其他开发工作，更有门槛的根本原因就是数学。每一个算法，要在训练集上最大
度拟合同时又保证泛化能力，需要不断分析结果和数据，调优参数，这需要我们对数据分布和模型底层的数学
理有一定的理解。所幸的是如果只是想合理应用机器学习，而不是做相关方向高精尖的research，需要的数学
啃一啃还是基本能理解下来的。至于更高深的部分，恩，博主非常愿意承认自己是『数学渣』。

基本所有常见机器学习算法需要的数学基础，都集中在微积分、线性代数和概率与统计当中。下面我们先过一
知识重点，文章的后部分会介绍一些帮助学习和巩固这些知识的资料。

3.1.1 微积分

- **微分的计算及其几何、物理含义**，是机器学习中大多数算法的求解过程的核心。比如算法中运用到梯度
  降法、牛顿法等。如果对其几何意义有充分的理解，就能理解“梯度下降是用平面来逼近局部，牛顿法是
  曲面逼近局部”，能够更好地理解运用这样的方法。
- **凸优化和条件最优化** 的相关知识在算法中的应用随处可见，如果能有系统的学习将使得你对算法的认识
  到一个新高度。

3.1.2 线性代数

- 大多数机器学习的算法要应用起来，依赖于高效的计算，这种场景下，程序员GG们习惯的**多层for循环**
  就行不通了，而大多数的循环操作可转化成矩阵之间的乘法运算，这就和线性代数有莫大的关系了
- 向量的**内积运算**更是随处可见。
- **矩阵乘法**与分解在机器学习的**主成分分析（PCA）和奇异值分解（SVD）** 等部分呈现刷屏状地出现。

3.1.3 概率与统计

从广义来说，机器学习在做的很多事情，和统计层面数据分析和发掘隐藏的模式，是非常类似的。

- **极大似然思想、贝叶斯模型** 是理论基础，朴素贝叶斯(Naïve Bayes)、语言模型(N-gram)、隐马尔科夫
  （HMM）、隐变量混合概率模型是他们的高级形态。
- 常见分布如高斯分布是混合高斯模型(GMM)等的基础。

3.2 典型算法

绝大多数问题用典型机器学习的算法都能解决，粗略地列举一下这些方法如下：

1. 处理分类问题的常用算法包括：**逻辑回归(工业界最常用)，支持向量机，随机森林，朴素贝叶斯(NLP中**
   用)，深度神经网络(视频、图片、语音等多媒体数据中使用)。
2. 处理回归问题的常用算法包括：**线性回归**，普通最小二乘回归（Ordinary Least Squares Regression），逐
   回归（Stepwise Regression），多元自适应回归样条（Multivariate Adaptive Regression Splines）
3. 处理聚类问题的常用算法包括：**K均值（K-means）**，基于密度聚类，LDA等等。

4. 降维的常用算法包括：**主成分分析（PCA）,奇异值分解（SVD）** 等。
5. 推荐系统的常用算法：**协同过滤算法**
6. 模型融合(model ensemble)和提升(boosting)的算法包括：**bagging，adaboost，GBDT，GBRT**
7. 其他很重要的算法包括：EM算法等等。

我们多插一句，机器学习里所说的**“算法”**与程序员所说的“数据结构与算法分析”里的“算法”略有区别。**前者更注结果数据的召回率、精确度、准确性等方面，后者更关注执行过程的时间复杂度、空间复杂度等方面。** 。然，实际机器学习问题中，对效率和资源占用的考量是不可或缺的。

3.3 编程语言、工具和环境

看了无数的理论与知识，总归要落到实际动手实现和解决问题上。而没有工具所有的材料和框架、逻辑、思路给你，也寸步难行。因此我们还是得需要合适的编程语言、工具和环境帮助自己在数据集上应用机器学习算法或者实现自己的想法。对初学者而言，Python和R语言是很好的入门语言，很容易上手，同时又活跃的社区支持，丰富的工具包帮助我们完成想法。相对而言，似乎计算机相关的同学用Python多一些，而数学统计出身的学更喜欢R一些。我们对编程语言、工具和环境稍加介绍：

3.3.1 python

python有着全品类的数据科学工具，从数据获取、数据清洗到整合各种算法都做得非常全面。

- 网页爬虫: scrapy
- 数据挖掘:
  - pandas：模拟R，进行数据浏览与预处理。
  - numpy：数组运算。
  - scipy：高效的科学计算。
  - matplotlib：非常方便的数据可视化工具。
- 机器学习:
  - scikit-learn：远近闻名的机器学习package。未必是最高效的，但是接口真心封装得好，几乎所有的器学习算法输入输出部分格式都一致。而它的支持文档甚至可以直接当做教程来学习，非常用心。于不是非常高纬度、高量级的数据，scikit-learn胜任得非常好(有兴趣可以看看sklearn的源码，也很意思)。
  - libsvm：高效率的svm模型实现(了解一下很有好处，libsvm的系数数据输入格式，在各处都非常常
  - keras/TensorFlow：对深度学习感兴趣的同学，也能很方便地搭建自己的神经网络了。
- 自然语言处理:
  - nltk：自然语言处理的相关功能做得非常全面，有典型语料库，而且上手也非常容易。
- 交互式环境:
  - ipython notebook：能直接打通数据到结果的通道，方便至极。强力推荐。

3.3.2 R

R最大的优势是开源社区，聚集了非常多功能强大可直接使用的包，绝大多数的机器学习算法在R中都有完善的可直接使用，同时文档也非常齐全。常见的package包括：RGtk2, pmml, colorspace, ada, amap, arules, biclust, cb descr, doBy, e1071, ellipse等等。另外，值得一提的是R的可视化效果做得非常不错，而这对于机器学习是非常助的。

3.3.3 其他语言

相应资深程序员GG的要求，再补充一下java和C++相关机器学习package。

- Java系列
  - WEKA Machine Learning Workbench 相当于java中的scikit-learn
  - 其他的工具如Massive Online Analysis（MOA）、MEKA 、 Mallet 等也非常有名。
  - 更多详细的应用请参考这篇文章《25个Java机器学习工具&库》
- C++系列
  - mlpack，高效同时可扩充性非常好的机器学习库。
  - Shark：文档齐全的老牌C++机器学习库。

3.3.4 大数据相关

- Hadoop：基本上是工业界的标配了。一般用来做特征清洗、特征处理的相关工作。
- spark：提供了MLlib这样的大数据机器学习平台，实现了很多常用算法。但可靠性、稳定性上有待提高

3.3.5 操作系统

- mac和linux会方便一些，而windows在开发中略显力不从心。所谓方便，主要是指的mac和linux在下载安
  软件、配置环境更快捷。
- 对于只习惯windows的同学，推荐anaconda，一步到位安装完python的全品类数据科学工具包。

3.4 基本工作流程

以上我们基本具备了机器学习的必要条件，剩下的就是怎么运用它们去做一个完整的机器学习项目。其工作流
如下:

3.4.1 抽象成数学问题

- **明确问题是进行机器学习的第一步**。机器学习的训练过程通常都是一件非常耗时的事情，胡乱尝试时间
  本是非常高的。
- 这里的抽象成数学问题，指的我们明确我们可以获得什么样的数据，目标是一个分类还是回归或者是聚
  的问题，如果都不是的话，如果划归为其中的某类问题。

3.4.2 获取数据

- **数据决定了机器学习结果的上限，而算法只是尽可能逼近这个上限**。
- 数据要有**代表性**，否则必然会过拟合。
- 而且对于分类问题，**数据偏斜不能过于严重**，不同类别的数据数量不要有数个数量级的差距。
- 而且还要对**数据的量级有一个评估**，多少个样本，多少个特征，可以估算出其**对内存的消耗程度**，判断
  练过程中内存是否能够放得下。如果放不下就得考虑改进算法或者使用一些降维的技巧了。如果数据量

在太大，那就要考虑分布式了。

3.4.3 特征预处理与特征选择

- 良好的数据要能够提取出良好的特征才能真正发挥效力。
- 特征预处理、数据清洗是很关键的步骤，往往能够使得算法的效果和性能得到显著提高。归一化、离散化、因子化、缺失值处理、去除共线性等，数据挖掘过程中很多时间就花在它们上面。这些工作简单可制，收益稳定可预期，是机器学习的基础必备步骤。
- 筛选出显著特征、摒弃非显著特征，需要机器学习工程师反复理解业务。这对很多结果有决定性的影响特征选择好了，非常简单的算法也能得出良好、稳定的结果。这需要运用特征有效性分析的相关技术，相关系数、卡方检验、平均互信息、条件熵、后验概率、逻辑回归权重等方法。

3.4.4 训练模型与调优

- 直到这一步才用到我们上面说的算法进行训练。现在很多算法都能够封装成黑盒供人使用。但是真正考水平的是调整这些算法的（超）参数，使得结果变得更加优良。这需要我们对算法的原理有深入的理解理解越深入，就越能发现问题的症结，提出良好的调优方案。

3.4.5 模型诊断

如何确定模型调优的方向与思路呢？这就需要对模型进行诊断的技术。

- 过拟合、欠拟合 判断是模型诊断中至关重要的一步。常见的方法如交叉验证，绘制学习曲线等。过拟合基本调优思路是增加数据量，降低模型复杂度。欠拟合的基本调优思路是提高特征数量和质量，增加模复杂度。
- 误差分析 也是机器学习至关重要的步骤。通过观察误差样本，全面分析误差产生误差的原因:是参数的还是算法选择的问题，是特征的问题还是数据本身的问题……
- 诊断后的模型需要进行调优，调优后的新模型需要重新进行诊断，这是一个反复迭代不断逼近的过程，要不断地尝试， 进而达到最优状态。

3.4.6 模型融合

- 一般来说，模型融合后都能使得效果有一定提升。而且效果很好。
- 工程上，主要提升算法准确度的方法是分别在模型的前端（特征清洗和预处理，不同的采样模式）与后（模型融合）上下功夫。因为他们比较标准可复制，效果比较稳定。而直接调参的工作不会很多，毕竟量数据训练起来太慢了，而且效果难以保证。

3.4.7 上线运行

- 这一部分内容主要跟工程实现的相关性比较大。工程上是结果导向，模型在线上运行的效果直接决定模的成败。 不单纯包括其准确程度、误差等情况，还包括其运行的速度(时间复杂度)、资源消耗程度（空复杂度）、稳定性是否可接受。