机器学习导论
习题三

151242041, 王昊庭, hatsuyukiw@gmail.com

2017 年 4 月 25 日

1 [30pts] Decision Tree Analysis

决策树是一类常见的机器学习方法，但是在训练过程中会遇到一些问题。

(1) **[15pts]** 试证明对于不含冲突数据 (即特征向量完全相同但标记不同) 的训练集，必存在与训练集一致 (即训练误差为 0) 的决策树;

(2) **[15pts]** 试分析使用“最小训练误差”作为决策树划分选择的缺陷。

Solution. (1) 使用反证法. 假设不存在一个完全符合训练集的决策树, 那么任取一个决策树, 必然存在一个叶子结点, 其中包含属于不同分类的多个数据, 且无法继续分割. (如果所有叶子结点都只包含一个数据点, 那么就可以使得分类完全符合训练集.) 但是根据假设, 不存在冲突的数据, 那么自然可以将这个叶子结点继续分割. 矛盾. 因此必存在与训练集一致 (即训练误差为 0) 的决策树.

(2) 若以最小训练误差作为决策树划分的依据, 那么总可以得到一棵与训练集 (几乎) 完全一致的决策树. (这里几乎的意思是不考虑冲突数据.) 但是由于真实世界中的真实分布总是有噪声的, 这样完全与训练集一致的决策树必然存在过拟合情况, 对于样本外数据的泛化能力很差.

2 [30pts] Training a Decision Tree

考虑下面的训练集：共计 6 个训练样本，每个训练样本有三个维度的特征属性和标记信息。详细信息如表1所示。

请通过训练集中的数据训练一棵决策树，要求通过“信息增益”(information gain) 为准则来选择划分属性。请参考书中图 4.4，给出详细的计算过程并画出最终的决策树。

Solution. 初始时信息熵为 $-2(\frac{1}{2}\log_2\frac{1}{2}) = 1$, 若以 **A** 为划分属性, 则数据集被划分为 {0, 0, 1} 和 {0, 1, 1}, 信息增益为 $1 - 2\frac{1}{2}\{-\left(\frac{1}{3}\log_2\frac{1}{3} + \frac{2}{3}\log_2\frac{2}{3}\right)\} = 0.0817$. 若以 **B** 为划分属性, 则数据集被划分为 {0, 0, 1, 1} 和 {0, 1}, 信息增益为 $1 - 2\frac{1}{2}\{-\left(\frac{1}{2}\log_2\frac{1}{2} + \frac{1}{2}\log_2\frac{1}{2}\right)\} = 0$. 若以 **C** 为划分属性, 则数据集被划分为 {0, 0, 1} 和 {0, 1, 1}, 信息增益为 $1 - 2\frac{1}{2}\{-\left(\frac{1}{3}\log_2\frac{1}{3} + \frac{2}{3}\log_2\frac{2}{3}\right)\} = 0.0817$. **A**, **C** 是同样好的划分属性, 由于 **A** 字典序更小, 所以这个作者选择 **C**.

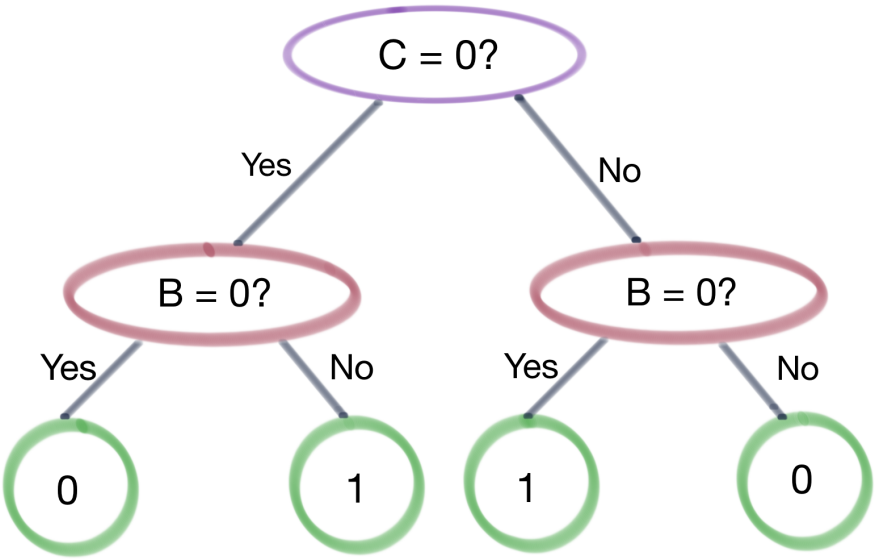
表 1: 训练集信息

序号	特征 **A**	特征 **B**	特征 **C**	标记
1	0	1	1	0
2	1	1	1	0
3	0	0	0	0
4	1	1	0	1
5	0	1	0	1
6	1	0	1	1

对于 **C**= 0 的标记为 {0, 1, 1} 的数据集, 选择 **B** 可以使数据集正好被正确分类, 因此信息增益最大, 所以选择 **B** 作为划分属性.

对于 **C**= 1 的标记为 {0, 0, 1} 的数据集, 选择 **B** 可以使数据集正好被正确分类, 因此信息增益最大, 所以选择 **B** 作为划分属性.

最终的决策树如下,



这个作者的评论: 事实上如果一开始选择 **A** 作为划分属性, 那么会得到一颗高度为 3 的满二叉树 (过程略). 由于选择 **C** 得到的决策树更为简洁, 并且存在诠释: 标记 $= B$ xor C, 根据奥卡姆剃刀原则, 选择简洁且有合理诠释的决策树. 当然这只是这个作者的归纳偏好和假设. 如果数据更大, 很难出现一个无关变量使得信息增益最大的情况.

3 [40pts] Back Propagation

单隐层前馈神经网络的误差逆传播 (error BackPropagation, 简称 BP) 算法是实际工程实践中非常重要的基础, 也是理解神经网络的关键。

请编程实现 BP 算法, 算法流程如课本图 5.8 所示。详细编程题指南请参见链接 : http:

//lamda.nju.edu.cn/ml2017/PS3/ML3_programming.html

在实现之后，你对 BP 算法有什么新的认识吗？请简要谈谈。

Solution. `main.py` 中的代码是这个作者自己完成的 Stanford CS231n 的作业. 除了两个函数的声明和注释的知识产权属于作业的设计者, 其余代码的知识产权都属于这个作者.

以下简单谈谈朴素 BP 算法训练单隐含层神经网络.

对于朴素 BP 算法来说, 在固定模型本身的结构的情况下, 唯三的超参数 (hyper parameter) 是学习率 (learning rate), 正则方法 (regularization) 和训练轮数 (number of epochs). 这个作者在日常调参过程中最喜欢使用的方法选择一个最大的不会使模型发散的学习率, 之后经过若干个 Epoch, 令学习率乘以一个系数 γ. 为了确定正则项, 在参数空间中搜索即可.

经过尝试发现调整超参数对于模型性能影响很小, 可能是因为问题本身比较简单.

最初这个作者的实现的性能比较差 (大概本机 1.2s per epoch), 经过助教赵鹏先生的提醒, 这个作者将 BP 算法全部用向量化方法实现, 性能大概提升了两个数量级. 在此表示感谢. 实现之后对 BP 算法的新认识是向量化很重要.

附加题 [30pts] Neural Network in Practice

在实际工程实现中，通常我们会使用已有的开源库，这样会减少搭建原有模块的时间。因此，请使用现有神经网络库，编程实现更复杂的神经网络。详细编程题指南请参见链接：http://lamda.nju.edu.cn/ml2017/PS3/ML3_programming.html

和上一题相比，模型性能有变化吗？如果有，你认为可能是什么原因。同时，在实践过程中你遇到了什么问题，是如何解决的？

Solution. 模型的性能有变化. 主要原因是模型的结构和激活函数发生了变化. 另外这个作者使用了更先进的优化方法 (Nestorov 的 SGD). 遇到的问题是虽然 Keras 和 Tensorflow 的官方文档声称不支持 Python 3.6, 但是这个作者成功地强行安装了, 希望不要出什么问题.