机器学习导论
习题三

Boardwell, Nanjing University

2017 年 7 月 10 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): 反证法：如果不存在与训练集完全一致的决策树，则至少在某一个分类的节点处出现了冲突，即存在了无法划分的多个数据（即这些数据存在完全一致的特征却有不同的标记）。否则，构造一个这样的决策树：从根节点开始，每一层节点为同一个划分属性的判断（不进行剪枝），则每一个当前的节点都可以分出若干个分支，每一个分支对应归类到当前节点中的元素当前划分属性的一个值（如果这些元素有n个不同的离散属性值，就分出n个子树对应这些值，如果是连续的数值，则排序后选取这些值中间的$n-1$个离散的数值划分出n个区间，让这些元素的数值落入的区间并且两两不同）。如果不存在冲突的标记，那么按照以上方法构造，一定可以生成一棵决策树使其满足与训练集完全一致。
假设不存在与训练集完全一致的决策树，则至少在某一个分类的节点处出现了冲突，与题设不含冲突数据矛盾，故必存在与训练集一致的决策树。
（2）：根据上一问我们可以看出，对于决策树这样的算法，一定存在一个与训练集一致的决策树。所以如果使用“最小训练误差”来训练一棵决策树的话，一定会最终收敛于某个与训练集完全一致的决策树（因为此时训练误差已经为0了，在“最小训练误差”的意义上已经达到了最优）。但是这样就导致了模型的过拟合，与训练集完全一致的模型的泛化性能一定不高，因为该模型将有限的样本的特征当成了整个问题的解，也就和“死记硬背”式的一一对应判断没有什么区别了，这样导致的过拟合（泛化能力差）就是使用“最小训练误差”作为决策树划分的最大缺陷。

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。

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

Solution.

1.D=all.

$Ent(D) = -(\frac{1}{2}log_2\frac{1}{2} + \frac{1}{2}log_2\frac{1}{2}) = 1$

A:

$Ent(D^0) = -(\frac{1}{3}log_2\frac{1}{3} + \frac{2}{3}log_2\frac{2}{3}) = 0.9183$

$Ent(D^1) = -(\frac{1}{3}log_2\frac{1}{3} + \frac{2}{3}log_2\frac{2}{3}) = 0.9183$

$Gain(D, A) = Ent(D) - \frac{1}{2}Ent(D^0) - \frac{1}{2}Ent(D^1) = 0.0817$

B:

$Ent(D^0) = -(\frac{1}{2}log_2\frac{1}{2} + \frac{1}{2}log_2\frac{1}{2}) = 1$

$Ent(D^1) = -(\frac{1}{2}log_2\frac{1}{2} + \frac{1}{2}log_2\frac{1}{2}) = 1$

$Gain(D, B) = Ent(D) - \frac{1}{2}Ent(D^0) - \frac{1}{2}Ent(D^1) = 0$

C:

$Ent(D^0) = -(\frac{1}{3}log_2\frac{1}{3} + \frac{2}{3}log_2\frac{2}{3}) = 0.9183$

$Ent(D^1) = -(\frac{1}{3}log_2\frac{1}{3} + \frac{2}{3}log_2\frac{2}{3}) = 0.9183$

$Gain(D, C) = Ent(D) - \frac{1}{2}Ent(D^0) - \frac{1}{2}Ent(D^1) = 0.0817$

After the first step, we can choose either A or C as the first attributes to to do devision, I use A to be the first node.

2.$D = (A = 1)$

$Ent(D) = -(\frac{1}{3}log_2\frac{1}{3} + \frac{2}{3}log_2\frac{2}{3}) = 0.9183$

B:

$Ent(D^1) = -(\frac{1}{2}log_2\frac{1}{2} + \frac{1}{2}log_2\frac{1}{2}) = 1$

$Ent(D^0) = -1log_2 1 = 0$

$Gain(D, B) = 0.9183 - \frac{1}{3}Ent(D^0) - \frac{2}{3}Ent(D^1) = 0.9183 - 0.667 = 0.2513$

C:

$Ent(D^1) = -(\frac{1}{2}log_2\frac{1}{2} + \frac{1}{2}log_2\frac{1}{2}) = 1$

$Ent(D^0) = -1log_2 1 = 0$

$Gain(D, C) = 0.9183 - \frac{1}{3}Ent(D^0) - \frac{2}{3}Ent(D^1) = 0.9183 - 0.667 = 0.2513$

In this step, we can choose either B or C as the attributes to to do devision, I use B to be

the devision node.
After this step, naturally using C as the devision attributes and finish this branch.
3.$D = (A = 0)$
$Ent(D) = -(\frac{1}{3}log_2\frac{1}{3} + \frac{2}{3}log_2\frac{2}{3}) = 0.9183$
B:
$Ent(D^1) = -(\frac{1}{2}log_2\frac{1}{2} + \frac{1}{2}log_2\frac{1}{2}) = 1$
$Ent(D^0) = -1log_21 = 0$
$Gain(D, B) = 0.9183 - \frac{1}{3}Ent(D^0) - \frac{2}{3}Ent(D^1) = 0.9183 - 0.667 = 0.2513$
C:
$Ent(D^1) = -1log_21 = 0$
$Ent(D^0) = -(\frac{1}{2}log_2\frac{1}{2} + \frac{1}{2}log_2\frac{1}{2}) = 1$
$Gain(D, C) = 0.9183 - \frac{1}{3}Ent(D^1) - \frac{2}{3}Ent(D^0) = 0.9183 - 0.667 = 0.2513$
In this step, we can choose either B or C as the attributes to to do devision, I use B to be the devision node.Like the situation of $A = 1$, naturally using C as the devision attributes and finish this branch.

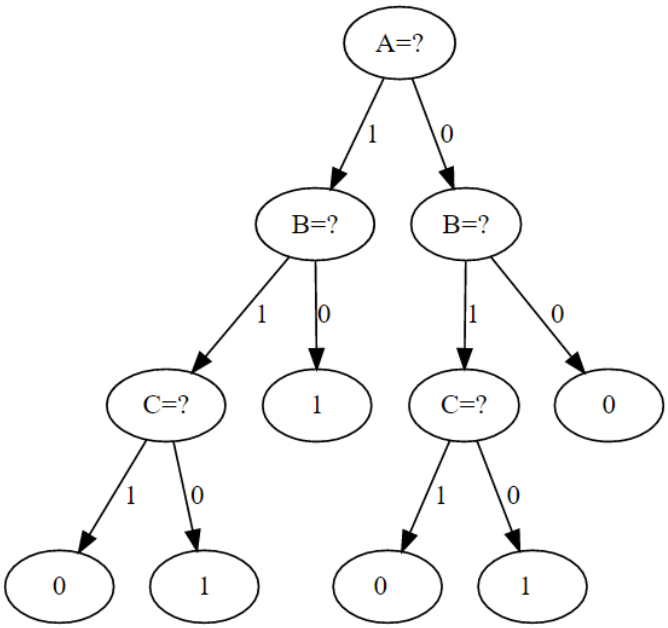

图 1: Decision tree

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.

1.The Back Propagation algorithm is very effective to recognize figures for which in my program after the first iteration I find the result's accuracy has became over 80%.

2.However, I think that the target function's phase plane many have many little potholes so that there are many local minimum to avoid gradient decent reaching a global minimum or the data set itself have many noises because that although accuracy reaching over 80% in the first step, it is hard to end in a higher accuracy(for example 95%). In my program it may reach an accuracy about 92% after 50 times of iteration.

3.Actually, the deepest impression on me is not the BP algorithm itself but how slow python program runs without vectorlization.I don't know why python is far slower than C++ program(maybe too many check to avoid making mistakes) and later I find outside environment running on more efficient language(for example, C) is very common in tensorflow and many other library functions.

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.

1.The new model using library functions show little improvement in accuracy when they are both reaching their own optimal condition(BP algorithm realized by myself being 92.1% and new model realized by keras being 92.6%).But I think new model in keras is better bacause its more hidden layers of neural network and more neurons in each layer represents more complexity which enables it to fit the problem more propriately.

2.Although the performance in these two models show little differences, the time cost still play an important role in this program. My first BP algorithm using python's "for" cycle show a very bad performance in time cost and it nearly needs 2 minutes to conduct one epoch. After an improvement by using matrix caculation in library numpy, it still need about 2 seconds to finish one epoch. But the keras program only need about 0.5 second to do an epoch although it uses a mini-accumulation BP algorithm which means using the accumulation loss funciton in a small batch consist of few data and it may decline the time cost.

One of the reasons of it may be that the keras and tensorflow package make the whole caculation in outside environment and avoid passing parameters frequently instead of numpy

only using some matrix caculation outside and passing a lot variables' values when caculating.