

Python 与数据挖掘
之 DKT (Deep Knowledge Tracing)



软件工程
吴雄
2019102971

2020 年 6 月

介绍

这学期跟着王老师又把 Python 温习了一遍，这次课程学到了之前好多没有触及的知识，比如装饰器等等，这给我后期的学习提供了非常大的帮助。最近我正在学习 DKT 即深度知识追踪相关的内容，然而大多数的论文没有代码。只是给了模型和结果。为了在这方面做点研究，我打算先复现一篇论文的实验。

工作量

- 特征工程-数据清洗
- 数据分布可视化
- 完成决策树
- 尝试复现主体部分代码（还没调通）

DKT 的简介


知识追踪 (Knowledge Tracing) 是根据学生过去的答题情况对学生的知识掌握情况进行建模，从而得到学生当前知识状态表示的一种技术。便于我们能准确地预测学生对于各个知识概念的掌握程度，以及学生在未来学习行为的表现。准确可靠的知识追踪意味着我们可以根据学生的自身的知识状态，给他们推荐合适的练习题目，比如，推荐给学生薄弱知识概念关联的题目，而过于困难或者过于简单的题目不应该被推荐，从而可以给学生进行高效的个性化教学。

早期的知识追踪模型都是依赖于一阶马尔科夫模型，例如贝叶斯

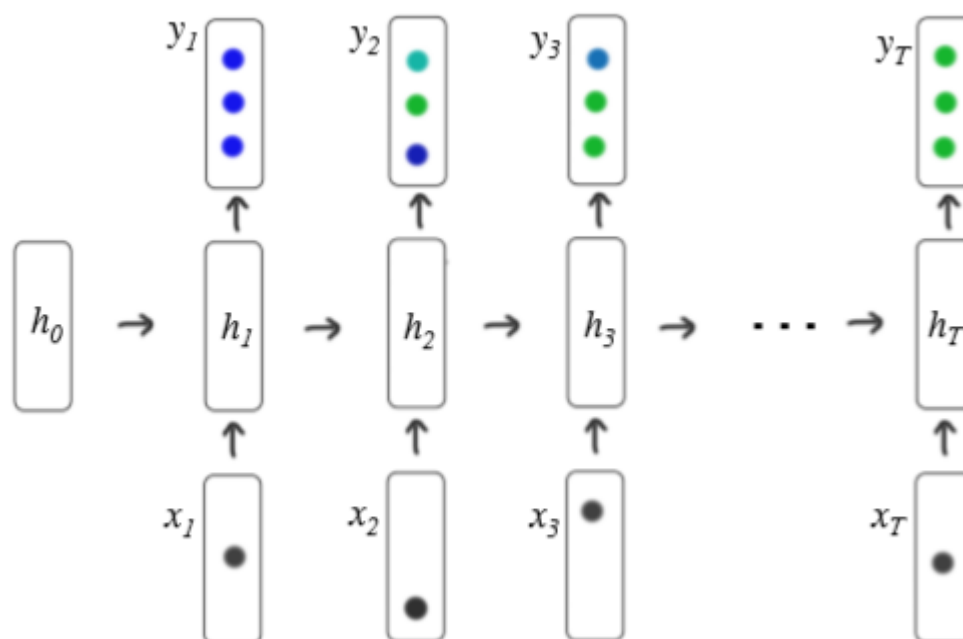
知识追踪 (Bayesian Knowledge Tracing)。将深度学习的方法引入知识追踪最早出现于发表在 NeurIPS 2015 上的一篇论文《Deep Knowledge Tracing》，作者来自斯坦福大学。在这篇论文中，作者提出了使用深度知识追踪 (Deep Knowledge Tracing) 的概念，利用 RNN 对学生的学习情况进行建模。

论文简介

Implicit Heterogeneous Features Embedding in Deep Knowledge Tracing

Haiqin Yang¹  · Lap Pong Cheung²

2015 年斯坦福大学提出 DKT 模型如下。



其实就是讲 RNN 或者 LSTM 应用到知识追踪上。

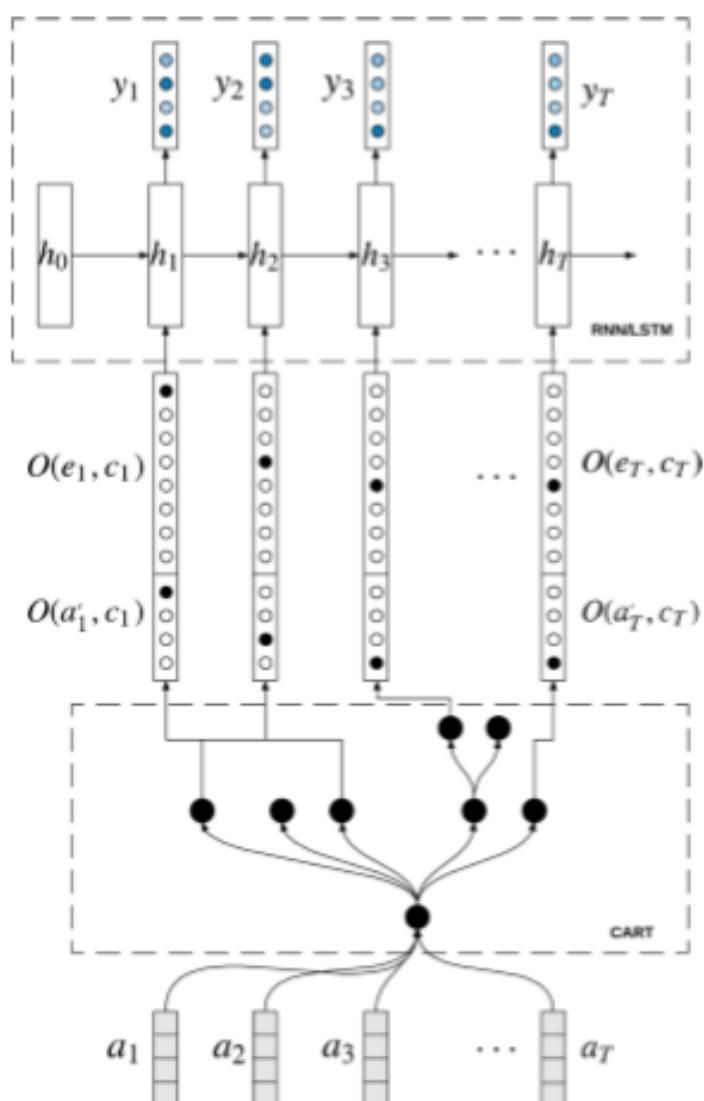
$$\mathbf{h}_t = \tanh(\mathbf{W}_{hx}\mathbf{x}_t + \mathbf{W}_{hh}\mathbf{h}_{t-1} + \mathbf{b}_h),$$

$$\mathbf{y}_t = \sigma(\mathbf{W}_{yh}\mathbf{h}_t + \mathbf{b}_y),$$

\mathbf{h}_t 是隐藏层， \mathbf{h}_t 的值跟 \mathbf{h}_{t-1} 和当前的输入 \mathbf{x}_t 有关。

这篇文章作者对斯坦福大学提出的 DKT 模型做了改变，也就是在输入到模型的地方加了一个决策树，通过学生的答题次数或者点击提示的次数通过决策树去判断这个学生是否掌握该知识点，然后输入到 DKT 模型中用 RNN/LSTM 进行训练得到了比较好的效果。

模型如下：



数据集

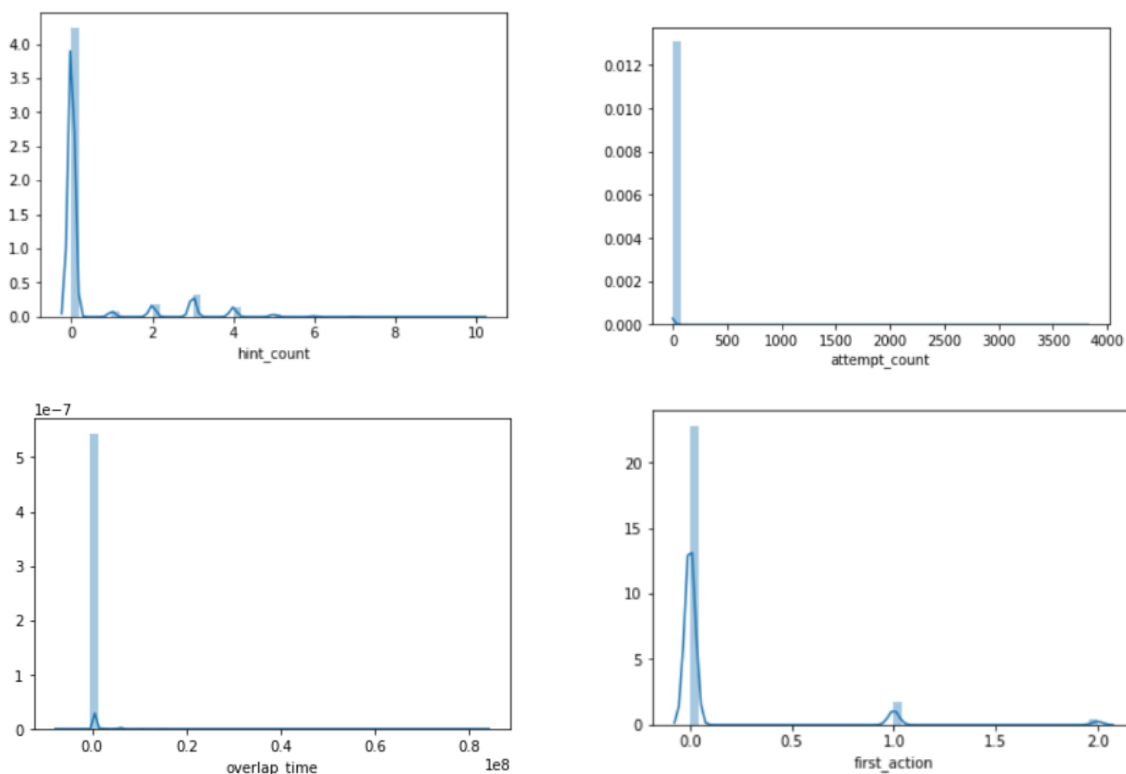
ASSISTments 2009-201。下载地址

<https://sites.google.com/site/assistmentsdata/home/assistment-20092010-data/skill-builder-data-2009-2010>

特征工程-数据清洗

我的思路这样的，通过可视化，找出边缘值，删除异常数据，然后进行决策树进行训练，这样可以一定程度上防止过拟合。

使用到的工具 seaborn。下面是四个特征的分布直方图。



就这四个数据的分布可以看出，attempt_count 主要分布在 1-10 次，但是有的数据达到了几百甚至上千，这些数据我们可以把他当成是异常的数据，可以清洗掉。

画图代码如下

```
import pandas as pd
import numpy as np
import seaborn as sb
import matplotlib as plt

'''
author:wuxiong
date:2020/6/8
'''
# 读取指定列索引字段的数据
#
#                                     usedclos=['original',
'attempt_count','ms_first_response','answer_text','assistent_id','type','hint_count','hint_t
otal',
#                                     'overlap_time','first_action','opportunity','tutor_mode','correct']
usedclos=['original',
'attempt_count','ms_first_response','assistent_id','hint_count','hint_total',
'overlap_time','first_action','opportunity','correct']
csv_data = pd.read_csv("./skill_builder_data.csv", usecols=usedclos,encoding='utf-8')
print(csv_data.head)
sb.distplot(csv_data['attempt_count'],kde=True)
# sb.distplot(csv_data['ms_first_response'],kde=True)
# sb.distplot(csv_data['hint_count'],kde=True)
# sb.distplot(csv_data['overlap_time'],kde=True)
```

数据清洗

```
import pandas as pd
import numpy as np
import seaborn as sb
import matplotlib as plt

'''
author:wuxiong
date:2020/6/8
'''
# 读取指定列索引字段的数据
#
#                                     usedclos=['original',
'attempt_count','ms_first_response','answer_text','assistent_id','type','hint_count','hint_t
otal',
#                                     'overlap_time','first_action','opportunity','tutor_mode','correct']
usedclos=['original',
'attempt_count','ms_first_response','assistent_id','hint_count','hint_total',
'overlap_time','first_action','opportunity','correct']
csv_data = pd.read_csv("./skill_builder_data.csv", usecols=usedclos,encoding='utf-8')
```

```

print(csv_data.info())
len1 = len(csv_data)
#删除任何包含 NA 值
csv_data = csv_data.dropna()
#删除包含任何空值的行
csv_data = csv_data.dropna(how='any')
#删除重复值
csv_data.duplicated('assistent_id')
print(csv_data['assistent_id'])
#删除异常点击的行
count = 0
#定义删除行的索引列表
dellist = []
for index,data in enumerate(csv_data['attempt_count']):
    if data > 10 and index < len(csv_data):
#         print('-----在删除中-----')
        count = count+1
        dellist.append(index)
print(count)
drop_data = csv_data.drop(dellist)
print('清洗完成~')
print('删除了{}行'.format(len1 - len(drop_data)))
print(drop_data.info())
drop_data.to_csv("data.csv",index=False,sep=',')

```

通过上述方法，我清洗掉了 3855 条无用的数据。

```

#         ...
525529    57830
525530    57843
525531    34577
525532    34577
525533    34577
Name: assistent_id, Length: 525534, dtype: int64
3855
清洗完成~
清洗了3855条数据
<class 'pandas.core.frame.DataFrame'>
Int64Index: 521679 entries, 0 to 525532
Data columns (total 10 columns):
assistent_id      521679 non-null int64
original           521679 non-null int64
correct            521679 non-null int64

```

开始画决策树:

```

import pandas as pd
import numpy as np
from sklearn.tree import DecisionTreeClassifier

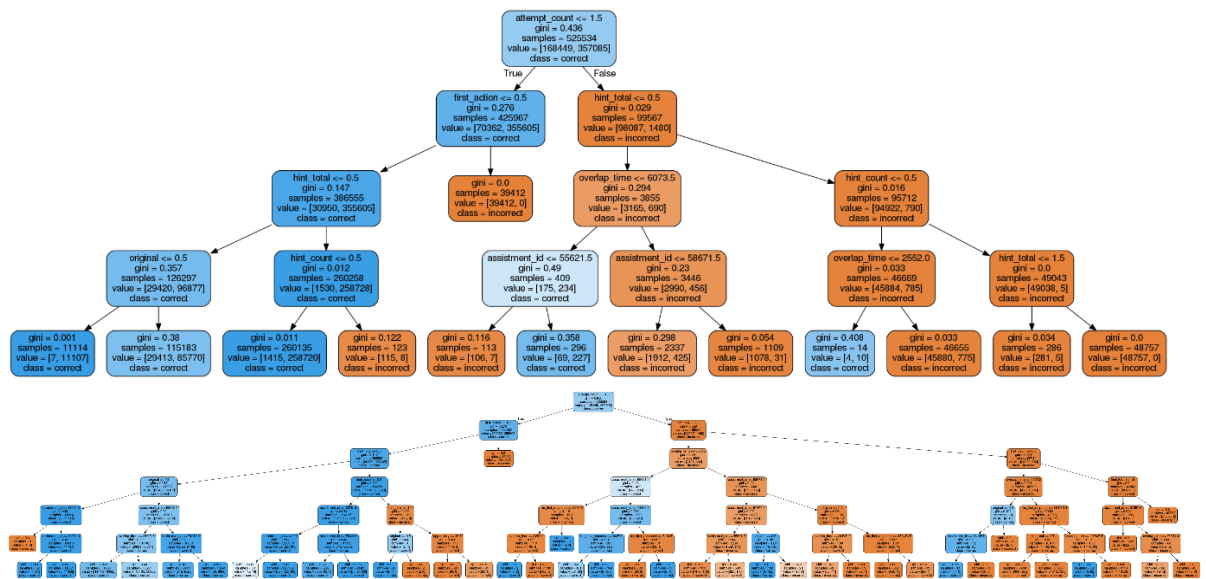
```

```

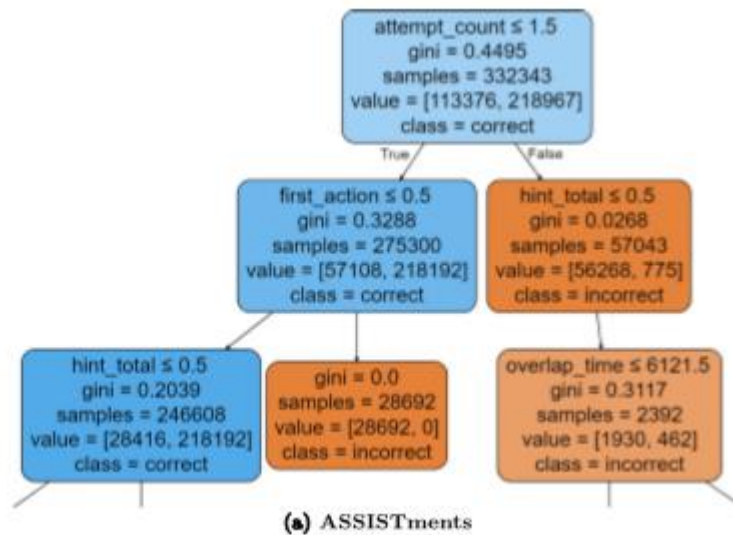
from sklearn.tree import export_graphviz
'''
author:wuxiong
date:2020/6/8
'''
# 读取指定列索引字段的数据
#
usedcols=['original',
'attempt_count','ms_first_response','answer_text','assistent_id','type','hint_count','hint_t
otal',
#
'overlap_time','first_action','opportunity','tutor_mode','correct']
usedcols=['original',
'attempt_count','ms_first_response','assistent_id','hint_count','hint_total',
'overlap_time','first_action','opportunity','correct']
csv_data = pd.read_csv("./skill_builder_data.csv", usecols=usedcols,encoding='utf-8')
print(csv_data.columns)
Y=csv_data['correct']
#删除 correct 列
del csv_data['correct']
print(csv_data.columns)
X=csv_data
feature =csv_data.columns
classname =['incorrect','correct']
#max_depth 是最大层数
tree_clf = DecisionTreeClassifier(max_depth=4)
tree_clf.fit(X, Y)
export_graphviz(
    tree_clf,
    out_file=("DecisionTreeClassifier.dot"),
    feature_names=feature,
    class_names=classname,
    rounded=True,
    filled=True,
)
print('成功! ')
'''
apt-get install graphviz
生成决策树的命令
dot -Tpng DecisionTreeClassifier.dot -o 123.png
'''

```

生成的决策树如下



论文中的决策树:



可以看出我的决策树和论文中的决策树基本上一致，即我在作者没有提供代码的情况下，通过作者的思路动手完成了他实验的第一个部分。

训练部分代码还没调试完成。还有些问题。这里就没把代码放上来。等我写完我会上传到我的 github (<https://github.com/realwuxiong>)。

总结

时光啊时光，总是飞快的流逝，这学期也过去了，这一年我从开

始接触机器学习深度学习，到现在可以慢慢上手写一些代码，期间也得益于老师的课程上的启发。做实验一般用的都是 Python，正好这学期开了这个 python 课，强化了我在 python 方面的基础。非常感谢王老师这学期的教导，今年因为疫情一直素未谋面，开学之后还希望向您讨教讨教。