吴贤铭智能工程学院

SHIEN-MING WU SCHOOL OF
INTELLIGENT ENGINEERING

The Difference between Using Python to implement Decision Tree and ANN Classification

—Use League of Legends Data Sets

Intelligent manufacturing

Yu Manchan Oct.2020

Content

The Difference between Using Python to implement Decision Tree and ANN Classification

—Use League of Legends Data Sets

Intelligent manufacturing
Yu Manchan   Oct.2020

I.Introduction

1. The introduction of LOL

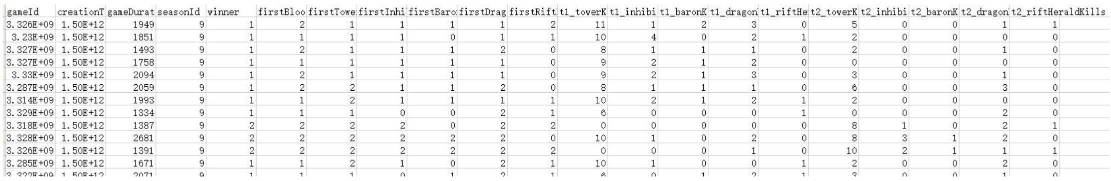
League of Legends (LoL) is developed by the Riot Games， which is a hero against MOBA competitive online Game,. With hundreds of individual heroes in the game, players will play a hero, cooperation, and with each other in the game of the team competition, the goal of the game is usually destroyed the other team's main base "crystal hub", each time is about 25 to 55 minutes of the game.

At the start of each game, the hero's level is 1 and increases as you gain experience points, eventually reaching level 18. At higher levels, players can unlock the hero's abilities and thus expand the way each hero uses them. Players can also use the money in the game to buy items to make the hero stronger, kill the other hero, or help the hero of the friend. Players earn gold by killing non-player characters known as troll and monster, by killing heroes of enemy teams, by destroying enemy buildings, and as the game goes on.

Now we're going to design a program that uses some special data to determine which side wins.

2. The introduction of Data Sets

Here are some key data types to use to determine whether a team has won or lost--First Baron, dragon, tower, blood, inhibitor, Rift Herald, the number of tower, inhibitor, Baron, dragon and Rift Herald kills.



gameId	creationT	gameDurat	seasonId	winner	firstBloo	firstTowe	firstInhi	firstBaro	firstDrag	firstRift	t1_towerK	t1_inhibi	t1_baronK	t1_dragon	t1_riftHe	t2_towerK	t2_inhibi	t2_baronK	t2_dragon	t2_riftHeraldKills
3.326E+09	1.50E+12	1949	9	1	2	1	1	1	1	2	11	1	2	3	0	5	0	0	1	1
3.23E+09	1.50E+12	1851	9	1	1	1	1	0	1	1	10	4	0	2	1	2	0	0	0	0
3.327E+09	1.50E+12	1493	9	1	2	1	1	1	2	0	8	1	1	1	0	2	0	0	1	0
3.327E+09	1.50E+12	1758	9	1	1	1	1	1	1	0	9	2	1	2	0	0	0	0	0	0
3.33E+09	1.50E+12	2094	9	1	2	1	1	1	1	0	9	2	1	3	0	3	0	0	1	0
3.287E+09	1.50E+12	2059	9	1	2	2	1	1	2	0	8	1	1	1	0	6	0	0	3	0
3.314E+09	1.50E+12	1993	9	1	1	2	1	1	1	1	10	2	1	2	1	2	0	0	0	0
3.329E+09	1.50E+12	1334	9	1	1	1	0	0	2	1	6	0	0	0	1	0	0	0	2	0
3.318E+09	1.50E+12	1387	9	2	2	2	2	0	2	2	0	0	0	0	0	8	1	0	2	1
3.328E+09	1.50E+12	2681	9	2	2	2	2	2	2	0	10	1	0	2	0	8	3	1	2	0
3.326E+09	1.50E+12	1391	9	2	2	2	2	2	2	2	0	0	0	1	0	10	2	1	1	1
3.285E+09	1.50E+12	1671	9	1	1	2	1	0	2	1	10	1	0	0	1	2	0	0	2	0
3.322E+09	1.50E+12	2071	9	1	1	1	0	1	2	1	6	0	1	2	1	3	0	0	1	0

Figure1. total 21 features

The names of each column is gameId, creationTime, gameDuration, seasonId, winner, firstBlood, firstTower, firstInhibitor, firstBaron, firstDragon, firstRiftHerald, t1_towerKills, t1_inhibitorKills, t1_baronKills, t1_dragonKills, t1_riftHeraldKills, t2_towerKills, t2_inhibitorKills, t2_baronKills, t2_dragonKill, t2_riftHeraldKills.

3. The introduction of DT

Decision tree algorithm includes two stages--training stage and test stage. In the training stage, it is necessary to use certain standards and rules to split the training sample set into several subsets, and then to split each subset with the same rules. This process recursively stops until each subset contains only samples belonging to the same class. During the training process, each segmentation node needs to save the attribute number of the classification. In the test phase, the test sample is identified from the root node to see which child node the sample belongs to, and the same recursively executes until the sample is divided into leaf nodes, at which point the sample belongs to the category of the current leaf node.

4. The introduction of ANN

Artificial neural network (ANN) is a nonlinear and adaptive information processing system composed of a large number of interconnected processing units. It is proposed on the basis of modern neuroscience research results, trying to simulate the way of brain neural network processing, memory information processing.

II. Algorithms

1. Decision Tree

1.1 Packages that need to be installed

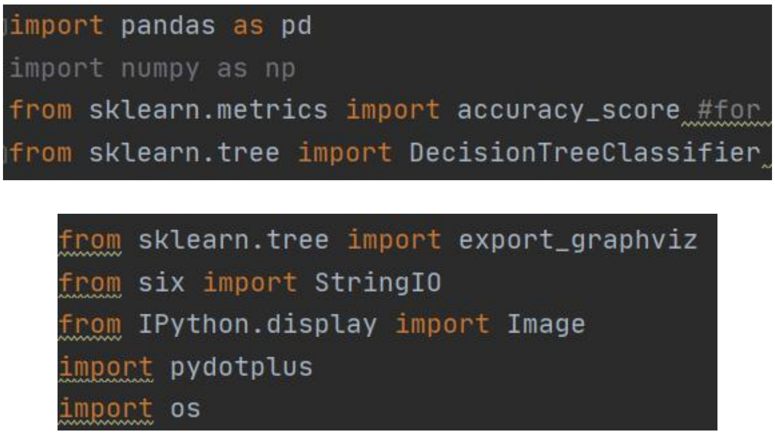
Graphviz	Pydotplus	Pandas	Numpy	six

Table 1

1.2 The libraries

pandas	numpy	sklearn.metrics
sklearn.tree	six	Ipython.display

Table 2



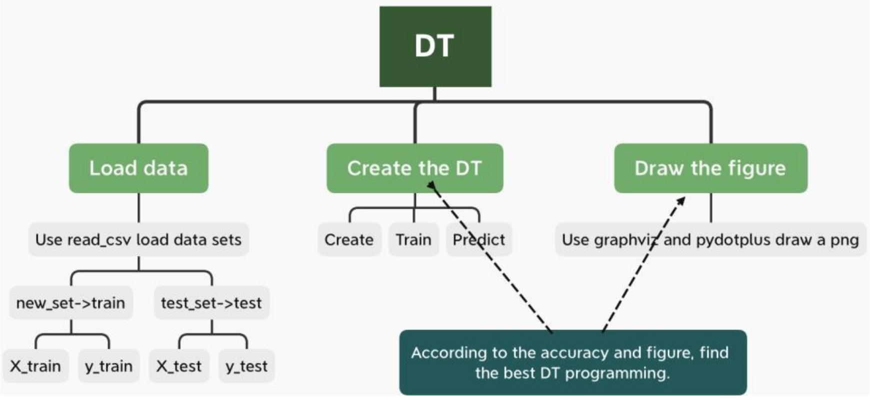
```
import pandas as pd
import numpy as np
from sklearn.metrics import accuracy_score #for
from sklearn.tree import DecisionTreeClassifier
```

```
from sklearn.tree import export_graphviz
from six import StringIO
from IPython.display import Image
import pydotplus
import os
```

Figure2. Loading libraries

1.3 The code

1.3.1 The steps to create a Decision Tree



DT
Load data
Create the DT
Draw the figure
Use read_csv load data sets
new_set->train
test_set->test
X_train
y_train
X_test
y_test
Create
Train
Predict
Use graphviz and pydotplus draw a png
According to the accuracy and figure, find the best DT programming.

Figure3. How to program a Decision Tree

1.3.1.1 Load the data

Using pandas to load data, we uses the function which name is *read_csv* to load two data sets--new_data and test_data. The two sets of data are divided into two groups(X, y) according to the types of features.

1.3.1.2 Create DT

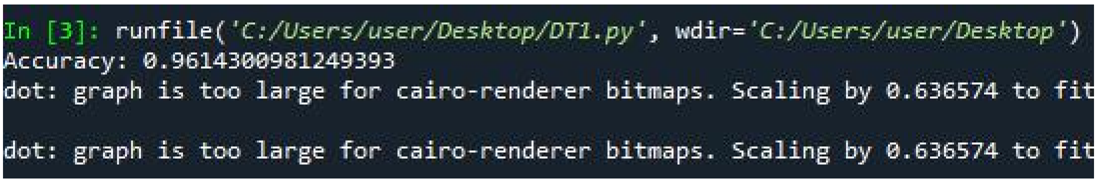
According train set create Decision Tree using DecisionTreeClassifier, then apply the X_test value to the resulting DT program, and compare the generated y_pred with y_test to determine the accuracy.

1.3.1.3 Draw the picture

Using graphviz and pydotplus to darw a picture in png.

1.3.1.4 Optimize the programming

In the above process, I found a problem that due to the large Train data, the computer automatically adjusted the accuracy of 0.9607986009909647 to 0.636574 when generating PNG images(Figure 4). This indicates that there are some problems with our program, and we can set some parameters to improve and optimize our program.



```
In [3]: runfile('C:/Users/user/Desktop/DT1.py', wdir='C:/Users/user/Desktop')
Accuracy: 0.9614300981249393
dot: graph is too large for cairo-renderer bitmaps. Scaling by 0.636574 to fit

dot: graph is too large for cairo-renderer bitmaps. Scaling by 0.636574 to fit
```

Figure4.

I add a parameter--max_depth to my code which determine the depth of the decision tree. Adding the parameter makes the programming less complex, make it easy to read.

When I set different values of the parameter in the process, I found that the accuracy changed.

The value of max_depth	Accuracy
Don’t set the parameter	0.9607986009909647 changes to 0.636574
3	0.9487515787428349
4	0.9537064024094044
7	0.9665792286019625
8	0.966870688817643
9	0.9667249587098028
10	0.9655591178470806
15	0.9622073253667541 changes to 0.688889

Table 3

As the data in table 3, when the value of max_depth is 8, the accuracy is the biggest. It suits best.

1.4 The Final Result

The final Decision Tree algorithm is to generate an algorithm based on new_data and then determine the accuracy according to train_set, set the parameters -- the number of layers of the decision tree, max_depth, change the decision tree to rub the tree to change the accuracy, and determine the best result.

When max_depth equals to 8, the accuracy is the biggest. The graph is shown in Figure5.

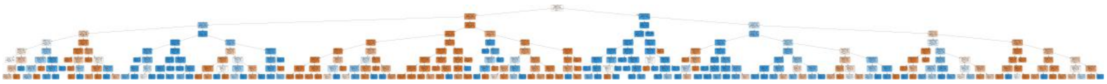


Figure5

2. Artificial Neural Network

2.1 Packages that need to be installed

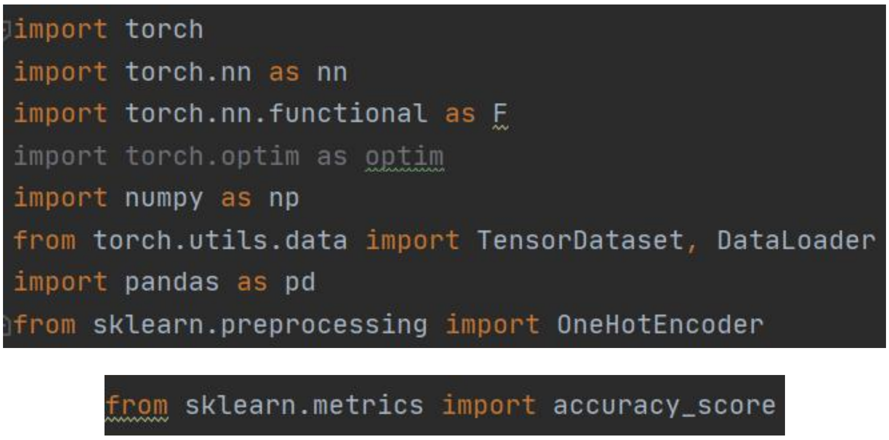
torch	numpy	pandas

Table 4

2.2 The libraries

torch.utils.data	sklearn.preprocessing	torch
numpy	pandas	sklearn.metrices

Table 5



```
import torch
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim
import numpy as np
from torch.utils.data import TensorDataset, DataLoader
import pandas as pd
from sklearn.preprocessing import OneHotEncoder
```

```
from sklearn.metrics import accuracy_score
```

Figure6

2.3 The code

2.3.1 The steps to create a Artificial neural network

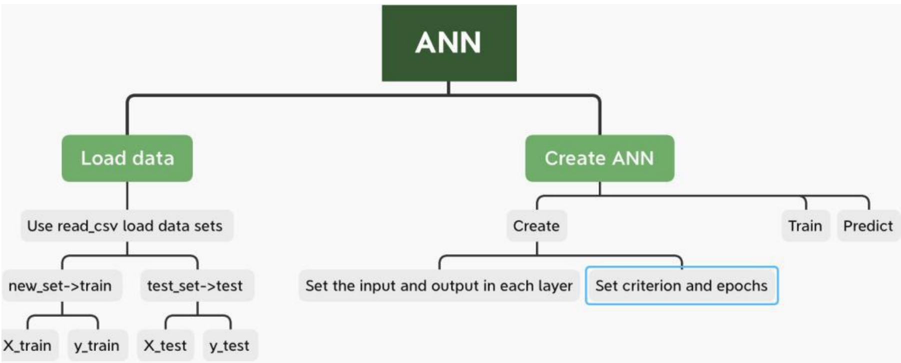


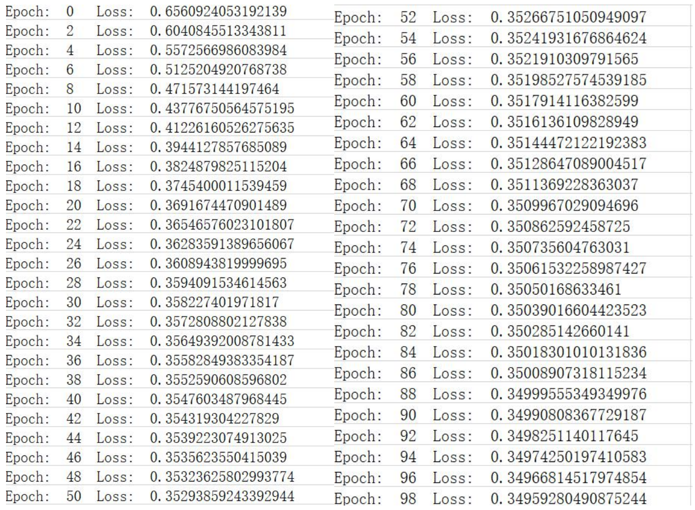
Figure7

2.3.1.1 Load the data

Using pandas to load data, we uses the function which name is *read_csv* to load two data sets--new_data and test_data. Using *torch.as_tensor* to transform DataFrame to Tensor. The two sets of data are divided into two groups(X, y) according to the types of features.

2.3.1.2 Create ANN

Neural networks have many layers, and in this case I chose a three-layer structure, an input layer, a hidden layer, and an output layer.Set the input number as 16 which is the feature numbers in X_train, and set the middle output as 32, the final output as 2(0,1). Then set some parameters Criterion, Optimizer and learning rate. We set the train models for 100 epochs, every two echos we'll output the lose. The result is shown in Figure 8.



```
Epoch:  0   Loss:  0.6560924053192139
Epoch:  2   Loss:  0.6040845513343811
Epoch:  4   Loss:  0.5572566986083984
Epoch:  6   Loss:  0.5125204920768738
Epoch:  8   Loss:  0.471573144197464
Epoch:  10  Loss:  0.43776750564575195
Epoch:  12  Loss:  0.41226160526275635
Epoch:  14  Loss:  0.3944127857685089
Epoch:  16  Loss:  0.3824879825115204
Epoch:  18  Loss:  0.3745400011539459
Epoch:  20  Loss:  0.3691674470901489
Epoch:  22  Loss:  0.36546576023101807
Epoch:  24  Loss:  0.36283591389656067
Epoch:  26  Loss:  0.3608943819999695
Epoch:  28  Loss:  0.3594091534614563
Epoch:  30  Loss:  0.358227401971817
Epoch:  32  Loss:  0.3572808802127838
Epoch:  34  Loss:  0.35649392008781433
Epoch:  36  Loss:  0.35582849383354187
Epoch:  38  Loss:  0.3552590608596802
Epoch:  40  Loss:  0.3547603487968445
Epoch:  42  Loss:  0.354319304227829
Epoch:  44  Loss:  0.3539223074913025
Epoch:  46  Loss:  0.3535623550415039
Epoch:  48  Loss:  0.35323625802993774
Epoch:  50  Loss:  0.35293859243392944
Epoch:  52  Loss:  0.35266751050949097
Epoch:  54  Loss:  0.35241931676864624
Epoch:  56  Loss:  0.3521910309791565
Epoch:  58  Loss:  0.35198527574539185
Epoch:  60  Loss:  0.3517914116382599
Epoch:  62  Loss:  0.3516136109828949
Epoch:  64  Loss:  0.35144472122192383
Epoch:  66  Loss:  0.35128647089004517
Epoch:  68  Loss:  0.3511369228363037
Epoch:  70  Loss:  0.3509967029094696
Epoch:  72  Loss:  0.350862592458725
Epoch:  74  Loss:  0.350735604763031
Epoch:  76  Loss:  0.35061532258987427
Epoch:  78  Loss:  0.35050168633461
Epoch:  80  Loss:  0.35039016604423523
Epoch:  82  Loss:  0.350285142660141
Epoch:  84  Loss:  0.35018301010131836
Epoch:  86  Loss:  0.35008907318115234
Epoch:  88  Loss:  0.34999555349349976
Epoch:  90  Loss:  0.34990808367729187
Epoch:  92  Loss:  0.3498251140117645
Epoch:  94  Loss:  0.34974250197410583
Epoch:  96  Loss:  0.34966814517974854
Epoch:  98  Loss:  0.34959280490875244
```

Figure8

Then we apply the X_test value to the resulting ANN algorithm, and compare the generated y_pred with y_test to determine the accuracy.

The final accuracy is 0.9621587486641406.

III.Comparison and discussion.

Comparing the two algorithms above, we can draw some conclusions.

1. When you want to get a clear and visual graph, choose the Decision Tree. It can explain how to get the final result clearly.
2. The construction and experiment time of Decision Tree is short than Artificial Neural Network. The DT algorithm run in less time than ANN.
3. Artificial Neural Network algorithm don't need the debugging process to optimize the resulting images.
4. Artificial Neural Network algorithm is suitable for noisy data, and Decision Tree algorithm is not very sensitive to missing data.

I also look up some relevant information on the Internet and make Table 6.

Decision Tree	Advantages	Feasible and effective results can be achieved for large data sources in a relatively short time.
		Not sensitive to missing values
		High efficiency, decision trees only need to be built once and used repeatedly

	Disadvantages	Continuous fields are harder to predict.
		When there are too many categories, errors may increase more quickly.
		It does not perform well when dealing with data with strong characteristic correlation
Artificial Neural Network	Advantages	High accuracy
		Suitable for noise data set
	Disadvantages	Training is slow
		Weak hermeneutics

Table 6

IV. Conclusion and Summary

In the project, we apply the Decision Tree algorithm and Artificial Neural Network algorithm to predict the winner of the two fighting time in LOL according to some related features like first blood, first tower and so on.

When we create these two algorithms, we not only get the relevant data, but also understand the difference between the two algorithms, which is more conducive for us to select the algorithm to predict the value according to the characteristics of the data.