

Project1

Classification of Unlabeled LOL Match Records with “Win/Loss”

Introduction

LoL is an online 5 vs. 5 competitive PC game in which one of the two teams is bound to defeat the other and win the game, and during the game, certain factors are to influence the result, for instance, the number of tower, inhibitor, baron, dragon and rift herald kills each team has.

In this project, classification algorithms and techniques must be applied to tackle an LoL classification problem and better strategies to win this game should be investigated.

3 million match records of solo gamers as training set are given, which are comprised of all publicly available game statistics of a match played by some gamer, including an important field called ‘winner’(‘1’ indicating team 1 won the match, or vice versa). To be more specific, the task is to create one or more classifiers that takes as input from any fields from the csv file given(except field ‘winner’), and labels the record as a ‘1’ or ‘2’.

Two classifiers decision tree and ANN are used to achieve this goal.

Algorithms

Decision Tree:

Decision tree is the most powerful and popular tool for classification and prediction.

A Decision tree is a flowchart like tree structure, where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (terminal node) holds a class label.

In this project, the decision tree can be constructed by the following steps:

1. Library importing and data loading
2. Train/Test split: Split training and testing data-set into features and target
3. Building a decision tree model
4. Model evaluation: Use accuracy_score to display accuracy

ANN(Artificial Neural Network):

Artificial neural networks are one of the main tools used in machine learning. As the “neural” part of their name suggests, they are brain-inspired systems which are intended to replicate the way that we humans learn. Neural networks consist of input and output layers, as well as (in most cases) a hidden layer consisting of units that transform the input into something that the output layer can use. They are excellent tools for finding patterns which are far too complex or numerous for a human programmer to extract and teach the machine to recognize.

In this project, the the classifier of ANN can be constructed by the following steps:

1. Imports and data-set
2. Train/Test split:
 - split training and testing data-set into features and target
 - convert split data from Numpy arrays to PyTorch tensors
3. Defining a neural network model:
 - Architecture: 3-layer MLP
 - <a>Fully connected hidden layer(16 input features(number of features in X),100 output features(arbitrary))
 - Output layer(100 input features(number of output features from the previous layer))
 - <c>Output features(number of distinct classes)

4. Model training:

Criterion creation: for loss measurement

Optimizer creation: for optimization: use Adam with a learning rate of 0.01

Epoch=500, keep track of time and loss for every 50 epochs

5. Model evaluation: use accuracy_score to display accuracy

Requirements

Pytorch: for model training

Pandas: for data loading and manipulation

Matplotlib: for data visualization

Numpy: for array manipulation

Sklearn: for train-test split

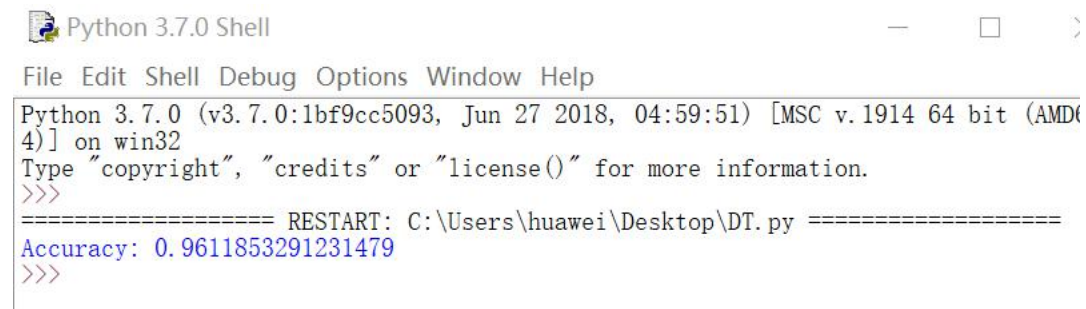
Results

<Decision Tree>

Test	Accuracy	Time
1	0.9597	1.49'
2	0.9607	1.77
3	0.9615	1.88'
4	0.9617	1.79'
5	0.9605	1.73

On average, the accuracy of the decision tree model is 0.9608, and training time is 1.732s

<Running screen shot>



```
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:59:51) [MSC v.1914 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\huawei\Desktop\DT.py =====
Accuracy: 0.9611853291231479
>>>
```

<ANN(Artificial Neural Network)>

Test1			Test2		
EPOCH	Accuracy	Time	EPOCH	Accuracy	Time
100	0.9626	2.27'	100	0.9625	2.34'
490	0.9699	9.68'	490	0.9698	8.78'
500	0.9701	9.58'	500	0.9698	9.05'
510	0.9694	10.11'	510	0.9689	9.35'
600	0.9686	11.12'	600	0.9686	11.11'
700	0.9697	12.91'	700	0.9697	12.70'
800	0.9687	14.54'	800	0.9684	14.41'
900	0.9692	15.92'	900	0.9689	16.00'
1000	0.9688	17.71'	100	0.9692	18.06'

Epoch=500 was chosen, so the training accuracy on average is 0.96995, and training time is 9.315s

<Running screen shot>

```
Epoch: 0 Loss: 0.666183352470398
Epoch: 50 Loss: 0.35072433948516846
Epoch: 100 Loss: 0.3484320044517517
Epoch: 150 Loss: 0.34756264090538025
Epoch: 200 Loss: 0.3467084765434265
Epoch: 250 Loss: 0.34565192461013794
Epoch: 300 Loss: 0.3445204496383667
Epoch: 350 Loss: 0.34340009093284607
Epoch: 400 Loss: 0.34238797426223755
Epoch: 450 Loss: 0.3415358364582062
The accuracy is 0.9692023705430876
```

Comparison

On average, ANN has a bit higher accuracy than decision tree classifier, but has much longer training time, especially with epoch value increasing. So in general, decision tree performs better in this given data-set. In the ANN algorithm, because the model was trained for 500 or more epochs, time and loss was kept track of as well, for every 50 epochs, so the calculation time will be longer.

Discussion

In the ANN training part, epoch value is actually difficult to determine in that small epoch value may lead to under-fitting and accordingly large epoch value may lead to over-fitting and long training time. Time permitting, I would try to use a 'for' loop to go through large number of epoch values and to weigh between the training time and training accuracy instead of roughly compare between the accuracy and training time every 100 epoch values.

Last, in this project, I have reviewed all the lab tutorials of this course and made fully use of them which strengthened my knowledge of these two classifiers and improved

my ability to fully understand the artificial neural network.

吴贤铭智能工程学院

19 智能制造

王一文