Classification Models for Predicting the Result of DOTA2 Matches

Abstract

Classification models is widely used in predicting games.In this paper, ten hero being selected before the game start will be used as variables and the result of the game will be the response.Logistic model and pnn(Probabilistic neural network) will be used to fit the samples. And I will compare these two models to select a better one for a further application; try to find the best hero to be chosen in certain situation before the game.

Introduction

Dota2 is a competitive game of action and strategy, played both professionally and casually by millions of passionate fans worldwide. The price for champion team in TI6(an annual Dota 2 eSports championship tournament) is worth 9 million dollars.Five of the players will play for team radiant, the other will play for team dire. Before a game start, every player will be given time to select their heroes. There are currently 113 heroes available and all of them can not be selected twice in a game. Heroes that are selected by each side have a great impact on the result of the game. Having a better pick means higher probability to win the game.
In order to win the game, one needs to know what hero he or she should choose given a certain situation (what heroes has been chosen for now and which of those is your partner and which is your enemy). A model ,which can calculate the probability for winning the game based on hero that are selected, can be used as an advice for selecting a hero.

DATA

1.Data Form

To collect data for the model, I used python to access the api of dota2.The original data form are shown as below:

radiant	player1	player2	player3	player4	player5
hero name	95	26	32	41	101

dire	player6	player7	player8	player9	player10
hero name	2	56	91	100	86

result
1

Number below 'player#' represents which hero is selected. If the result = 1 then it means radiant win, if result = 0 then it means dire win.

One problem about this form of data is that the type of variable is list, so that they are not suitable for logistic model. In this case we transform the data to a form shown as below:

	hero1	hero2	...	hero112	hero113	hero114
status	1	0	...	-1	0	0

If the n^{th} hero is selected for radiant, the status will be equal to 1 (like hero1 for example). If it is selected for dire, the status will be equal to -1. If it is not selected, then the status will be equal to 0.

2.Prepare the Data

By using Python, I get 3000 samples to train the model. After deleting samples that has missing data, there are 1721 samples left.

From introduction we know that there are 113 heroes available at the moment. However, the data we get from Python indicates that there are 114 heroes. The most possible reason for this contradiction is that there is one hero is not available at this moment.

To find out which hero is not available, I calculate the times chosen for each heroes and find that 24^{th} hero is never been selected in 1721 games. Therefor I delete the data for 24^{th} hero.

Another thing should be worried about is that if we know whether the other 112 heroes are selected or not , we know the status of the hero left. For example , if we know number 1 to 4 is selected by radiant, 5 to 9 is selected by dire, and 10 to 113 is not selected, then we know that hero number 114 is selected by radiant.

By fitting a linear regression model using the data, part of the result is shown below:

```
Call:
lm(formula = b2$V115 ~ ., data = b2)

Residuals:
    Min      1Q  Median      3Q     Max
-0.9337 -0.4513  0.1283  0.4257  0.9791

Coefficients: (1 not defined because of singularities)
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.5329632  0.0120603  44.192  < 2e-16 ***
V2           0.0542439  0.0424929   1.277  0.20195
V3           0.0519302  0.0407033   1.276  0.20220
V4          -0.0909497  0.0726379  -1.252  0.21072
```

```
V111         -0.2418593  0.0902879  -2.679  0.00746 **
V112          0.0779876  0.0721319   1.081  0.27978
V113          0.0747235  0.0863859   0.865  0.38717
V114                 NA         NA      NA       NA
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.484 on 1608 degrees of freedom
Multiple R-squared:  0.1225,    Adjusted R-squared:  0.06135
F-statistic: 2.004 on 112 and 1608 DF,  p-value: 1.019e-08
```

From the result we know that singularity does exist. As a result, I remove the data for 114th hero.

3.Final Data

After what I did above, the final table for data is shown below:

Match Number	hero1	hero2	...	hero23	hero25	...	hero113
1			...			...	
2			...			...	
3			...			...	
4			...			...	

Model Selection

Not only the heroes but also the in-game performance of players will influence the result of game. Like we discussed above, a classification model that has type output is not what we want. We need a model that can output probability. According to this fact, logistic regression and pnn(Probabilistic neural network) are suitable.We will discuss these two model and compare them to find a better model.

Logistic regression

Logistic regression, or logit regression, or is a regression model where the outcome variable is categorical. Often this is used when the variable is binary (e.g. yes/no, survived/dead, pass/fail, etc.)

Logistic regression measures the relationship between the categorical response variable and one or more predictor variables by estimating probabilities.

1.train the model

```
logistic<-glm(b3$V115 ~ ., data=b3, family=binomial(link='logit'))
summary(logistic)
##
## Call:
## glm(formula = b3$V115 ~ ., family = binomial(link = "logit"),
##     data = b3)
##
```

```
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.0772  -1.0860   0.5773   1.0357   2.1592
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.154458   0.053598   2.882  0.00395 **
## V11         -0.015436   0.255814  -0.060  0.95188
## V12         -0.258052   0.189861  -1.359  0.17410
## V13         -0.026737   0.251556  -0.106  0.91536
## V24         -0.402872   0.240961  -1.672  0.09454 .
## V31          0.315679   0.260059   1.214  0.22480
## V32          0.549990   0.199128   2.762  0.00575 **
## V42          0.367521   0.202444   1.815  0.06946 .
## V43          0.321519   0.316156   1.017  0.30917
## V44          0.376537   0.172988   2.177  0.02951 *
## V45         -0.023954   0.315612  -0.076  0.93950
## V46         -0.169855   0.244916  -0.694  0.48798
## V47         -0.373949   0.223667  -1.672  0.09454 .
## V48          0.285996   0.211735   1.351  0.17678
## V49          0.228454   0.250564   0.912  0.36190
## V50         -0.324634   0.268564  -1.209  0.22675
## V51          0.738503   0.289210   2.554  0.01066 *
## V57          0.437220   0.236793   1.846  0.06483 .
### V67          0.617060   0.240022   2.571  0.01015 *
## V68          0.585540   0.298865   1.959  0.05009 .
## V69         -0.039245   0.340636  -0.115  0.90828
## V70          0.487104   0.202023   2.411  0.01590 *
## V71          0.291735   0.199092   1.465  0.14283
## V72         -0.585541   0.322040  -1.818  0.06903 .
## V73         -0.013429   0.219099  -0.061  0.95113
## V74          0.004374   0.170552   0.026  0.97954
## V75          0.364966   0.210476   1.734  0.08292 .
## V80         -0.611799   0.327023  -1.871  0.06137 .
## V81          0.352152   0.228159   1.543  0.12272
## V82         -0.258251   0.285220  -0.905  0.36523
## V83          0.210755   0.257773   0.818  0.41359
## V84          0.107636   0.218700   0.492  0.62260
## V85         -0.243246   0.298362  -0.815  0.41492
## V86         -0.445480   0.217563  -2.048  0.04060 *
## V87          0.095509   0.263179   0.363  0.71667
## V88          0.388856   0.235327   1.652  0.09845 .
## V89         -0.474083   0.362295  -1.309  0.19068
## V101        -0.104128   0.260802  -0.399  0.68970
```

```
## V102          0.442445   0.246931   1.792  0.07317 .
## V103          0.036767   0.370045   0.099  0.92085
## V111         -1.291314   0.464851  -2.778  0.00547 **
## V112          0.364871   0.319178   1.143  0.25297
## V113          0.344444   0.375571   0.917  0.35908
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2382  on 1720  degrees of freedom
## Residual deviance: 2156  on 1608  degrees of freedom
## AIC: 2382
##
## Number of Fisher Scoring iterations: 4
```

Because we transform the 10 variable to 112 variable, the result requires many spaces. In this case, only part of the result is shown.

2.Evaluate the Model

Normally, to evaluate the model, first thing we do is to find the p-value of the variable and see if they are significant. However, in this model, we transformed the 10 variable to 112. As a result, it is reasonable that some of the p-value are not small enough.

Instead, I use a group of test-data to see if the model is good or not. The output of test-data is shown below:

```
##         1         2         3         4         5         6         7
## 0.6903123 0.4667828 0.6879116 0.5377311 0.5093042 0.2214033 0.4684324
##         8         9        10        11        12        13        14
## 0.6744380 0.5587661 0.3811555 0.4043075 0.4598607 0.3109666 0.2768680
##        15        16        17        18        19        20        21
## 0.1839187 0.6627218 0.5534760 0.5114099 0.4909689 0.5996546 0.7441690
##        22        23        24        25        26        27        28
## 0.5567861 0.7784050 0.8324328 0.7147310 0.5226326 0.8516701 0.3776174
##        29        30        31        32        33        34        35
## 0.4931790 0.7277976 0.4815520 0.3412750 0.8350813 0.6418698 0.3174345
##        36        37        38        39        40        41        42
## 0.4536143 0.8083799 0.4985076 0.3266262 0.6422034 0.9140428 0.7136978
##        43        44        45        46        47        48        49
## 0.5616526 0.7889653 0.3902771 0.6490930 0.7564582 0.6163420 0.8100803
##        50        51        52        53        54        55        56
```

```
## 0.1911676 0.4677390 0.4513419 0.5807610 0.5094336 0.4371390 0.3541504
##        57        58        59        60        61        62        63
## 0.2404191 0.5549234 0.3849697 0.3104483 0.4551840 0.4959623 0.6794007
##        64        65        66        67        68        69        70
## 0.3656159 0.8078476 0.7692555 0.3378307 0.5988161 0.6548169 0.4996608
##        71        72        73        74        75        76        77
## 0.4554183 0.3502766 0.4138683 0.3774347 0.2915163 0.2198424 0.3865586
##        78        79        80        81        82        83        84
## 0.2692183 0.6934214 0.4103612 0.4525397 0.4255129 0.6542397 0.6867094
##        85        86        87        88        89        90        91
## 0.5933203 0.4294795 0.5650687 0.3589957 0.5014846 0.3955783 0.4755699
##        92        93        94        95        96        97        98
## 0.8506399 0.8066782 0.3550650 0.6437411 0.4631570 0.5825170 0.6073741
##        99       100
## 0.4895782 0.6802402
```

Not that most of the probability is around 40% to 60%, it is hard to say if the model is good or not. In this case we only take the data that have probability below 20% or higher than 80%.

After testing samples, we find that among 157 samples that have the probability below 20% or higher than 80%, 128 sample correctly predict the result. So that the accuracy of predicting is 81.1%% percent. This does not mean that our model is not good. On the contrary, assume we have an output of 80% winning, if our output is correct, that means 20 out of 100 games will end up with dire victory (radiant fail).In this case the accuracy will equals to 80%. Comparing 81.1% with 80%, we find that the model is quite good.

Probabilistic neural network

A probabilistic neural network (PNN) is a feedforward neural network, which is widely used in classification and pattern recognition problems. In the PNN algorithm, the parent probability distribution function (PDF) of each class is approximated by a Parzen window and a non-parametric function. Then, using PDF of each class, the class probability of a new input data is estimated and Bayes’ rule is then employed to allocate the class with highest posterior probability to new input data. By this method, the probability of mis-classification is minimized. This type of ANN was derived from the Bayesian network and a statistical algorithm called Kernel Fisher discriminant analysis. It was introduced by D.F. Specht in the 1966. In a PNN, the operations are organized into a multilayered feedforward network with four layers:input layer,hidden layer, pattern layer/summation layer, and output layer.

Normally a probabilistic neural network will only give the type output. However, I

notice that the output layer is depends on the summation layer, which calculates the probability for each type. Because of this characteristic , we can get the probability for radiant victory.

Train the model

Because R does not include a pack for PNN, I wrote one based on codes in PNN tool from MATLAB.The code is shown in appendix.

Evaluate the model

Part of the prediction is shown below:

```
##                [,1]      [,2]
##  [1,] 0.4756339 0.5243661
##  [2,] 0.4769131 0.5230869
##  [3,] 0.4750188 0.5249812
##  [4,] 0.4778438 0.5221562
##  [5,] 0.4771934 0.5228066
##  [6,] 0.4783342 0.5216658
##  [7,] 0.4760474 0.5239526
##  [8,] 0.4739535 0.5260465
##  [9,] 0.4767609 0.5232391
## [10,] 0.4778052 0.5221948
## [11,] 0.4770813 0.5229187
```

We notice that the results are almost the same. There might be two reasons that are responsible for it:

1. The data is wrong.
2. The necessary normalization treatment in PNN model is not suitable for for variables that only have 2 or 3 possible input. As a result, same cells in hidden layer are activated.

To find out which is the true reason, a simplified data with only 3 variables is created to train the model.Each of the variables still has 3 possible input -1,0,1. After training the model again, I find that the results stay the same, which means this data is simply not suitable for PNN.

Conclusion for Models

Logistic regression has a high accuracy for telling the probability of radiant victory. On the other hand, the data set is not suitable for PNN. As a result, Logistic regression is chosen for further application.

Further application

Building a predicting model is the first step. The problem of what hero should one choose is the purpose of training this model. From the logistic regression model we trained above, we are able to find which hero will results in the highest probability of

winning based on what heroes are selected by other 9 players.

For example,
We use the first sample in the data set:

radiant	player1	player2	player3	player4	player5
hero name	4	46	71	86	87

dire	player6	player7	player8	player9	player10
hero name	11	30	49	57	99

Assume that player 1 is the last player to choose the hero, and to find out what he or she should choose, we calculate all the probability for each his available choices and find the hero that leads to the highest winning rate:

```
b5<-b3[1,c(1:112)]
b5[1,4]<-0
b6<-as.data.frame(c(1:104))
b7<-merge(b5,b6)
t0<-as.data.frame(c(1:104)+9)
n<-1
for (i in 1:103){
  m<-1
  while (m==1){
    if(sum(b7[,n])==0){
      b7[i,n]<-1
      m<-0
      t0[i,1]<-n
    } else{
      n<-n+1
    }
  }
}
o1<-predict(logistic,,type='response',newdata=b7)
o2<-as.data.frame(o1)
o3<-cbind(o2,t0)
o4<-filter(o3,o1==max(o2))
if(o4[1,2]>23){
  o4[1,2]<-o4[1,2]+1
}
names(o4)[1:2]<-c("probability","hero")
o4[1,]
##     probability  hero
## 1    0.7846699     51
```

As the code shown above , the best choice is to choose hero51 and the probability to

win is 78.4%.

Future work

Finding the best hero based on the other 9 heroes selected is just a start. The next step is to find the best hero based on different number of heroes selected in order to help the players who do not get the last pick. To achieve this, game theory will be added to the model to find the global best choice considering all probability including what heroes others may pick.

Reference

Reference

1. Using Machine Learning to Recommend Heros for Dota2 Matches.http://kevintechnology.com/post/71621133663/using-machine-learning-to-recommend-heroes-for

2. A data mining project on Dota2 replays to predict match outcome with partial game state data.
https://github.com/dwarakanandan/Data-Mining-DOTA2

3. To win or not to win?A prediction model to determine the outcome of a DotA2 match.https://cseweb.ucsd.edu/~jmcauley/cse255/reports/wi15/Kaushik_Kalyanaraman.pdf

Appendix

PNN Code in R

```
mapminmaxDirect<-function(xData, minNum , maxNum)
{
  if(!is.matrix(xData))
  {
    stop('error input current data')
  }

  N<-nrow(xData)


  maxX <- matrix(apply(xData,2,max),nrow = 1)

  minX <- matrix(apply(xData,2,min),nrow = 1)


  maxXMat <- matrix(rep(maxX , N),nrow = N ,byrow = TRUE)
```

```
  minXMat <- matrix(rep(minX , N),nrow  = N , byrow = TRUE)


  xTransform <- (maxNum - minNum)/(maxXMat - minXMat)*(xData - minXMat)+
    minNum
  result<-list()



  result$maxNum<-maxNum
  result$minNum <- minNum
  result$xTransform <- xTransform
  result$minX<-minX
  result$maxX <- maxX

  return(result)

}
mapminmaxReverse<-function(xData , modelMapminmax)
{
  if(!is.list(modelMapminmax))
  {
    stop('error input data')
  }

  if (!setequal(c("maxNum","minNum","xTransform","minX","maxX" ),
                names(modelMapminmax)))
  {
    stop('error input data list')
  }

  N <- nrow(xData)

  maxX<-modelMapminmax$maxX
  minX<-modelMapminmax$minX
  maxNum<-modelMapminmax$maxNum
  minNum<-modelMapminmax$minNum

  maxXMat <- matrix(rep(maxX , N),nrow = N ,byrow = TRUE)
  minXMat <- matrix(rep(minX , N),nrow  = N , byrow = TRUE)

  preX<-(xData - minNum)/(maxNum - minNum)*(maxXMat - minXMat)+minXMat
  return(preX)
```

```
}



mapminmaxPre<-function(xData , modelMapminmax)
{
  if(!is.list(modelMapminmax))
  {
    stop('error input data')
  }

  if (!setequal(c("maxNum","minNum","xTransform","minX","maxX" ),
                names(modelMapminmax)))
  {
    stop('error input data list')
  }

  maxX<-modelMapminmax$maxX
  minX<-modelMapminmax$minX
  maxNum<-modelMapminmax$maxNum
  minNum<-modelMapminmax$minNum

  N<-nrow(xData)

  maxXMat <- matrix(rep(maxX , N),nrow = N ,byrow = TRUE)
  minXMat <- matrix(rep(minX , N),nrow  = N , byrow = TRUE)


  xNew <- (maxNum - minNum)/(maxXMat - minXMat)*(xData - minXMat)+
    minNum
  return(xNew)

}
pnn<-function(xData , classX , testData ,Sigma)
{
  if(!is.matrix(xData) || !is.matrix(classX) || !is.matrix(testData))
  {
    stop('error input current data')
  }

  if(ncol(xData)!=ncol(testData))
  {
    stop('error input data')
```

```
}

nxData<-nrow(xData)
ntestData<-nrow(testData)

disMatrix<-matrix(0,ntestData , nxData)



minNum <-(-1)
maxNum <- 1

result<-mapminmaxDirect(xData ,minNum , maxNum )
xNewData <- result$xTransform
testNewData<-mapminmaxPre(testData , result)
result
xNewData
testNewData

for (i in c(1:ntestData))
{
  oneTest<-testNewData[i,,drop = FALSE]
  oneTestMat <- matrix(rep(oneTest , nxData),nrow = nxData , byrow = TRUE)

  disMatrix[i,]<-sqrt(apply((oneTestMat - xNewData)^2,1,sum))

}

probMatrix<-exp(-disMatrix/(2*Sigma^2))

totalClass <-sort(unique(classX))

lastProbMatrix<-matrix(0,ntestData , length(totalClass))

for (i in c(1:ntestData))
{
  pro<-probMatrix[i,,drop = FALSE]
  for (j  in c(1:length(totalClass)))
  {
    ijIndex<-which(classX == totalClass[j])
    lastProbMatrix[i,j]<-sum(pro[ijIndex])
  }

}
```

```
  pos<-lastProbMatrix/rowSums(lastProbMatrix)
  return(pos)

}
```