Deep Learning in League of Legend

Ruohua Yin

2017 Fall

Deep Learning in League of Legend Data

Relation between win and factors

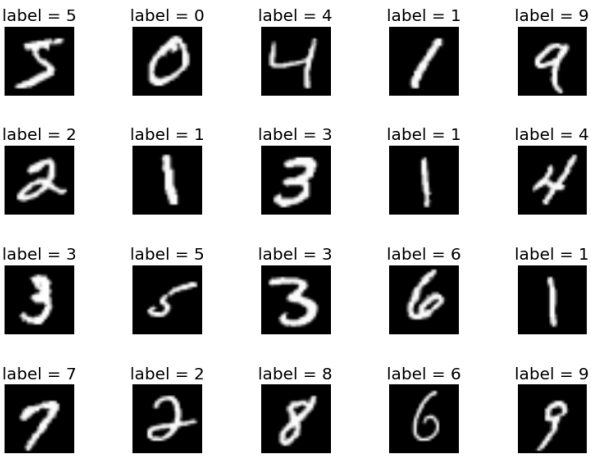
Abstract

League of Legends is a popular MOBA (Multiplayer Online Battle Arena) game published by Riot. The game is a team based strategy game where the main goal is to bring down the enemy's Nexus (base).

It's a very complicated gaming system, which pits 2 teams of 5 players each. Each player selects a champion from a rooster of several dozens. Champions have unique strengths and skills making team coordination essential for winning.

There are multiple choices in every single second that each player can make during the 30 minutes' match, which the combination of the choice decide whether they are going to make the advantage to the team, then finally make the team to bring down enemy's Nexus.

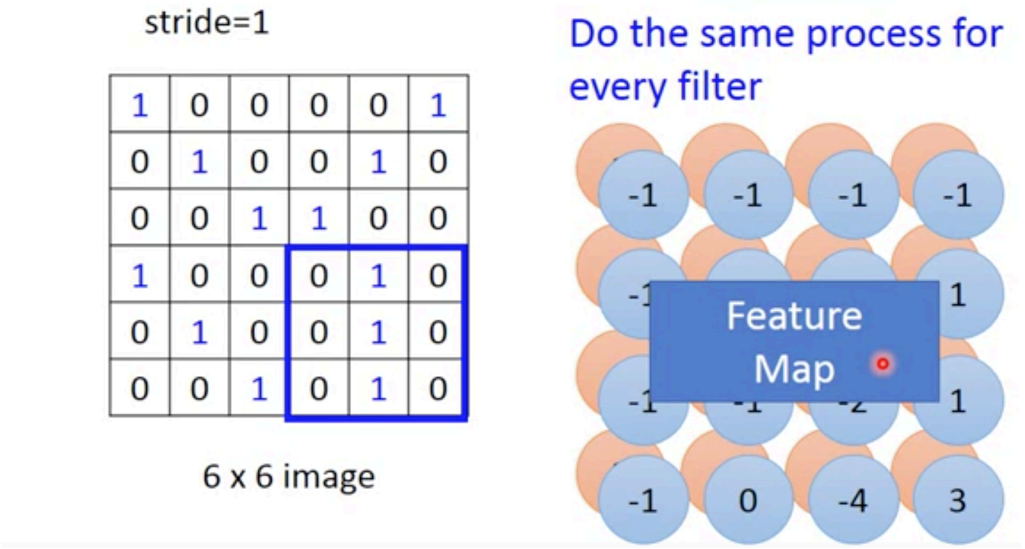
My project will involved with an measurement on how much does the things like quantity of tower kills, dragon kills, herald kills and other factors determine the victory, and an index will be generated for each factor, which could be seen as a guidance to the team on where to put their emphasize during the match. Then a deep learning model will be implemented by several steps, the game data over 51,000 matches will be reshaped in order to fit in the convolution neural network. CNN is a widely used model to image recognition. How the CNN could perform good in game data as well will be explained in the following part.



Background research of related work

I. https://towardsdatascience.com/tagged/league-of-legends
There is a vast amount of data in just a single LoL game. This dataset takes the most relevant information and makes it available easily for use in things such as attempting to predict the outcome of a LoL game, analyzing which in-game events are most likely to lead to victory, understanding how big of an effect bans of a specific champion have, and more.

II. I'd like to show up the MNIST project because the way they organize the neural network really enlighten me of trying to apply the model into the game data rather than apply it to what CNN does best always, image recognition.



The way CNN reduce parameter and amount of connection compared with MLP which hold all the layer fully connected could not only apply on image recognition but also game data, because the game data is generated in a very complicate system as well.

Just like MNIST dataset do convolution & maxpool twice from a 1*28*28 dataset to a smaller group of data, then feed in 2 or 3 fully connected layer network. The game data could also be reshape first in a square or rectangular. The filter of size 2 or 3 could also represent somewhat a feature to some extent in different layers. The way I organize my CNN model will be mentioned in the following part of the project.

Data Source

I. This is the master data I'm going to use, which is collected by someone else by Riot game API. More data is collected myself by Riot API according to the need in the project.

https://www.kaggle.com/jaytegge/league-of-legends-data-analysis/data

General Info

This is a collection of over 50,000 ranked games from the game League of Legends, as well as json files containing a way to convert between champion and summoner spell IDs and their names. For each game, there are fields for:

- Game ID
- Game Duration (in seconds)
- Season ID
- Winner (0 = team1, 1 = team2)
- First Baron, dragon, tower, blood, inhibitor and Rift Herald (0 = team1, 1 = team2)
- Champions and summoner spells for each team (Stored as Riot's champion and summoner spell IDs)
- The number of tower, inhibitor, Baron, dragon and Rift Herald kills each team has
- The 5 bans of each team (Again, champion IDs are used)

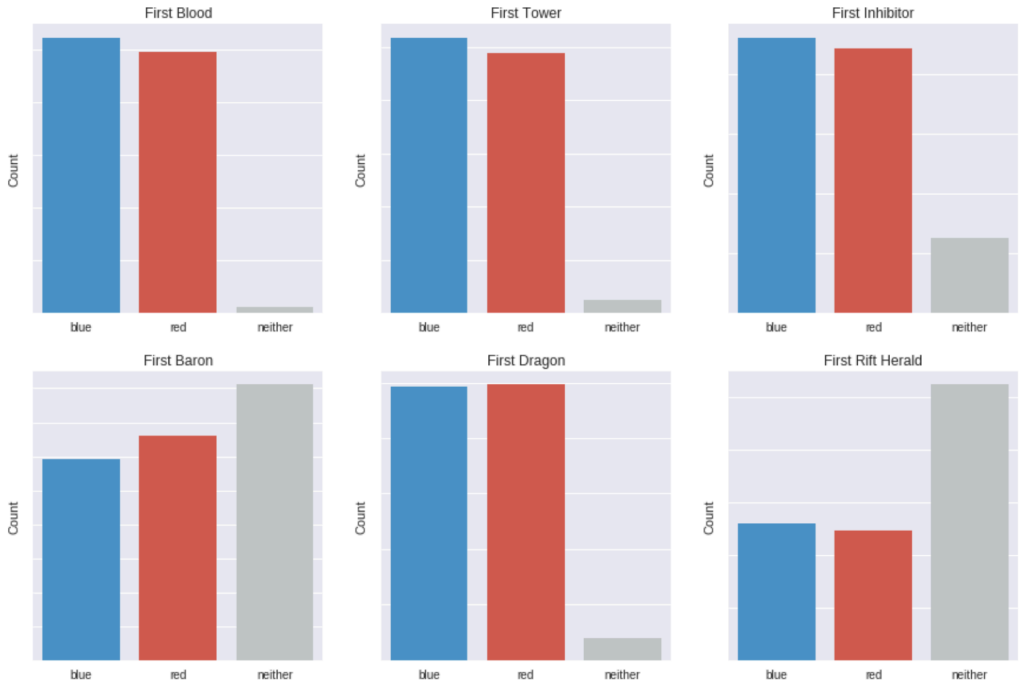
This dataset was collected using the Riot Games API, which makes it easy to lookup and collect information on a users ranked history and collect their games. However finding a list of usernames is the hard part, in this case I am using a list of usernames scraped from 3rd party LoL sites.

Possible Uses

There is a vast amount of data in just a single LoL game. This dataset takes the most relevant information and makes it available easily for use in things such as attempting to predict the outcome of a LoL game, analysing which in-game events are most likely to lead to victory, understanding how big of an effect bans of a specific champion have, and more.

First Baron, dragon, tower, blood, inhibitor and RiftHerald, and the overall number of these factor is great important factors who are most likely to lead to victory.

	firstBlood	firstTower	firstInhibitor	firstBaron	firstDragon	firstRiftHerald
blue	26113	25861	23054	14758	24690	12948
red	24822	24416	22160	16474	24800	12363
neither	555	1213	6276	20258	2000	26179



The dataset record the champion ID for every match, every single champion have their weakness and strength, which the 5 choice for each team decide their features.

② Champion features gain by Riot API

RATE LIMITS	20 requests every 1 seconds 100 requests every 2 minutes

The data I'm searching for is their strength which is represent by difficulty, attack, defense and magic. The previous dataset hold all champion id for 10 players in each game. And these 2 dataset could be put together to define how was the two teams' champion choice look like.

```
# Champion ID in season 9
a= np.shape(ID_array)
b = np.reshape(ID_array,139)
b
```

```
array([110, 111, 112, 113, 114, 236, 115, 117,  90, 238,  91, 119,  92,
       516,  96,  10,  98,  99,  11,  12,  13,  14,  15,  16,  17,  18,
        19, 240, 120, 121,   1, 122,   2, 245,   3,   4, 126,   5, 127,
         6,   7,   8,   9,  20,  21,  22,  23,  24,  25,  26,  27,  28,
```

	t1_ban1	t1_ban2	t1_ban3	t1_ban4	t1_ban5	t2_ban1	t2_ban2	t2_ban3	t2_ban4	t2_ban5
0	Riven	Janna	Cassiopeia	Draven	Kayn	Fiora	Vayne	Karma	Soraka	Caitlyn
1	Caitlyn	Darius	Teemo	Xayah	Warwick	Master Yi	Vayne	Zed	Caitlyn	Illaoi
2	Lulu	Janna	Twitch	Soraka	Blitzcrank	Yasuo	Zed	Kha'Zix	Maokai	Evelynn
3	Zed	Vayne	Ornn	Fiora	Cho'Gath	Camille	Tristana	Kayn	Janna	Caitlyn
4	Malzahar	Lee Sin	Thresh	Morgana	Cho'Gath	Garen	Master Yi	Braum	Darius	Tristana

Request static data of champions in json from api.riotgames.com Then the data will be stored in .json file locally.

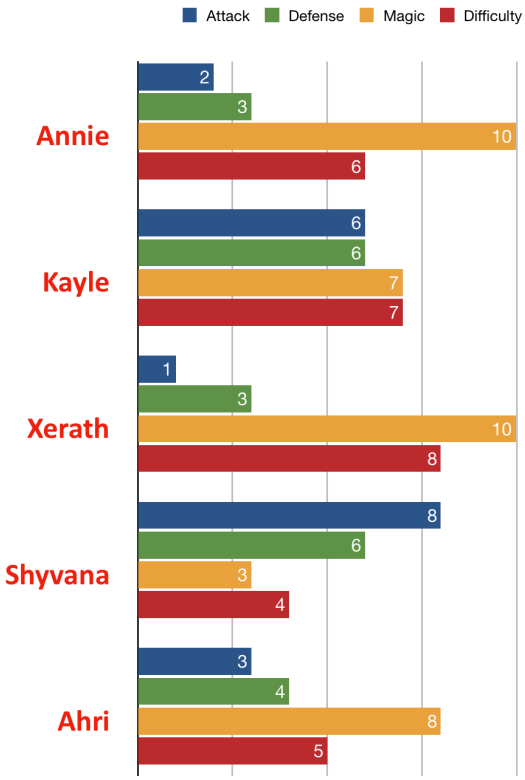
```
In [258]: from urllib.request import urlopen
          import json
          api = "RGAPI-96c97826-3019-4499-a4ed-8c07d9eb2807"
          content = urlopen("https://na1.api.riotgames.com/lol/static-data/v3/champions?locale=en_US&tags
                            =info&dataById=true&api_key="+api)
          info_json = json.loads(content.read())
          info_json
```

Review Info of the champions. The data collection is going to observe the balance of these 4 features for a match.

```
c_info = pd.read_json('data.json')
c_info = pd.read_json((c_info['data']).to_json(),orient='index')
c_info.head()
```

	id	info	key	name	title
1	1	{'attack': 2, 'defense': 3, 'magic': 10, 'diff...	Annie	Annie	the Dark Child
10	10	{'attack': 6, 'defense': 6, 'magic': 7, 'diffi...	Kayle	Kayle	The Judicator
101	101	{'attack': 1, 'defense': 3, 'magic': 10, 'diff...	Xerath	Xerath	the Magus Ascendant
102	102	{'attack': 8, 'defense': 6, 'magic': 3, 'diffi...	Shyvana	Shyvana	the Half-Dragon
103	103	{'attack': 3, 'defense': 4, 'magic': 8, 'diffi...	Ahri	Ahri	the Nine-Tailed Fox


```
{
    "type": "champion",
    "version": "7.24.2",
    "data": {
        "1": {
            "info": {
                "difficulty": 6,
                "attack": 2,
                "defense": 3,
                "magic": 10
            },
            "title": "the Dark Child",
            "id": 1,
            "key": "Annie",
            "name": "Annie"
        },
```

Methods

I. Multivariate linear model

Multilinear model is a proper tool to help me recognize which in-game events are most likely to lead to victory, and the dataset 1 provide just the enough and proper kind of data to build up the model.

① Matches with game duration less than 900 seconds will be discarded.

```
#Discard the match data which Duration is less than 900 seconds.
games_valid = df.loc[df['gameDuration']>900]
games_valid.head()
```

	winner	gameDuration	firstBlood	firstTower	firstInhibitor	firstBaron	firstDragon	firstRiftHerald	t1_towerKills	t1_inhibitorKills	...	t2_towerKills	t2_inhibitorK
0	0	1949	0	1	1	1	1	0	11	1	...	5	
1	0	1851	1	1	1	0	1	1	10	4	...	2	
2	0	1493	0	1	1	1	0	0	8	1	...	2	
3	0	1758	1	1	1	1	1	0	9	2	...	0	
4	0	2094	0	1	1	1	1	0	9	2	...	3	

② Tower kills, inhibitor kills, baron kills and dragon kills of the two team will calculate the difference between them. The data left to build up the model is shown below.

	winner	diff_towerKills	diff_inhibitorKills	diff_baronKills	diff_dragonKills	diff_riftHeraldKills
0	0	-6	-1	-2	-2	1
1	0	-8	-4	0	-2	-1
2	0	-6	-1	-1	0	0
3	0	-9	-2	-1	-2	0
4	0	-6	-2	-1	-2	0

The data column in green is the variables, column in red is the dependent variable which represent how much advantages they have over the other team.

```
# inhibitorKills  between diff_towerKills & diff_dragonKills
inhibitor_model = ols("diff_inhibitorKills ~ diff_towerKills + diff_dragonKills + diff_baronKills   -1",
                      data = games_linear ).fit()
inhibitor_model.summary()
```

OLS Regression Results

Dep. Variable:	diff_inhibitorKills	**R-squared:**	0.774
Model:	OLS	**Adj. R-squared:**	0.774
Method:	Least Squares	**F-statistic:**	5.729e+04
Date:	Thu, 02 Nov 2017	**Prob (F-statistic):**	0.00
Time:	13:58:29	**Log-Likelihood:**	-72194.
No. Observations:	50227	**AIC:**	1.444e+05
Df Residuals:	50224	**BIC:**	1.444e+05
Df Model:	3		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
diff_towerKills	0.2566	0.001	280.077	0.000	0.255	0.258
diff_dragonKills	0.0249	0.003	9.085	0.000	0.020	0.030
diff_baronKills	0.2048	0.006	36.869	0.000	0.194	0.216

Omnibus:	3723.974	**Durbin-Watson:**	1.997
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	20957.430
Skew:	-0.006	**Prob(JB):**	0.00
Kurtosis:	6.164	**Cond. No.**	8.54

Is the relationship significant?

Result from above The coefficient for math is significantly different from 0 because both p-values are 0.000, which are smaller than 0.05. The result state that the coefficients has a very small possibility to be 0.

Result from R We can also see the result summary generated by language R. see as follows.

```
Residuals:
    Min      1Q  Median      3Q     Max
-7.7418 -0.5868  0.0038  0.5941  8.5675

Coefficients:
                  Estimate Std. Error t value Pr(>|t|)
(Intercept)     -0.0038132  0.0045571  -0.837    0.403
diff_towerKills  0.2565263  0.0009175 279.599   <2e-16 ***
diff_dragonKills 0.0249833  0.0027447   9.102   <2e-16 ***
diff_baronKills  0.2050973  0.0055661  36.848   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.019 on 50223 degrees of freedom
Multiple R-squared:  0.7738,    Adjusted R-squared:  0.7738
F-statistic: 5.727e+04 on 3 and 50223 DF,  p-value: < 2.2e-16
```

model assumptions violated?¶

Usually there are 4 assumptions for a model:

① The Y-value(or the errors,"e") are independent

② The Y-values can be expressed as a linear function of X variable

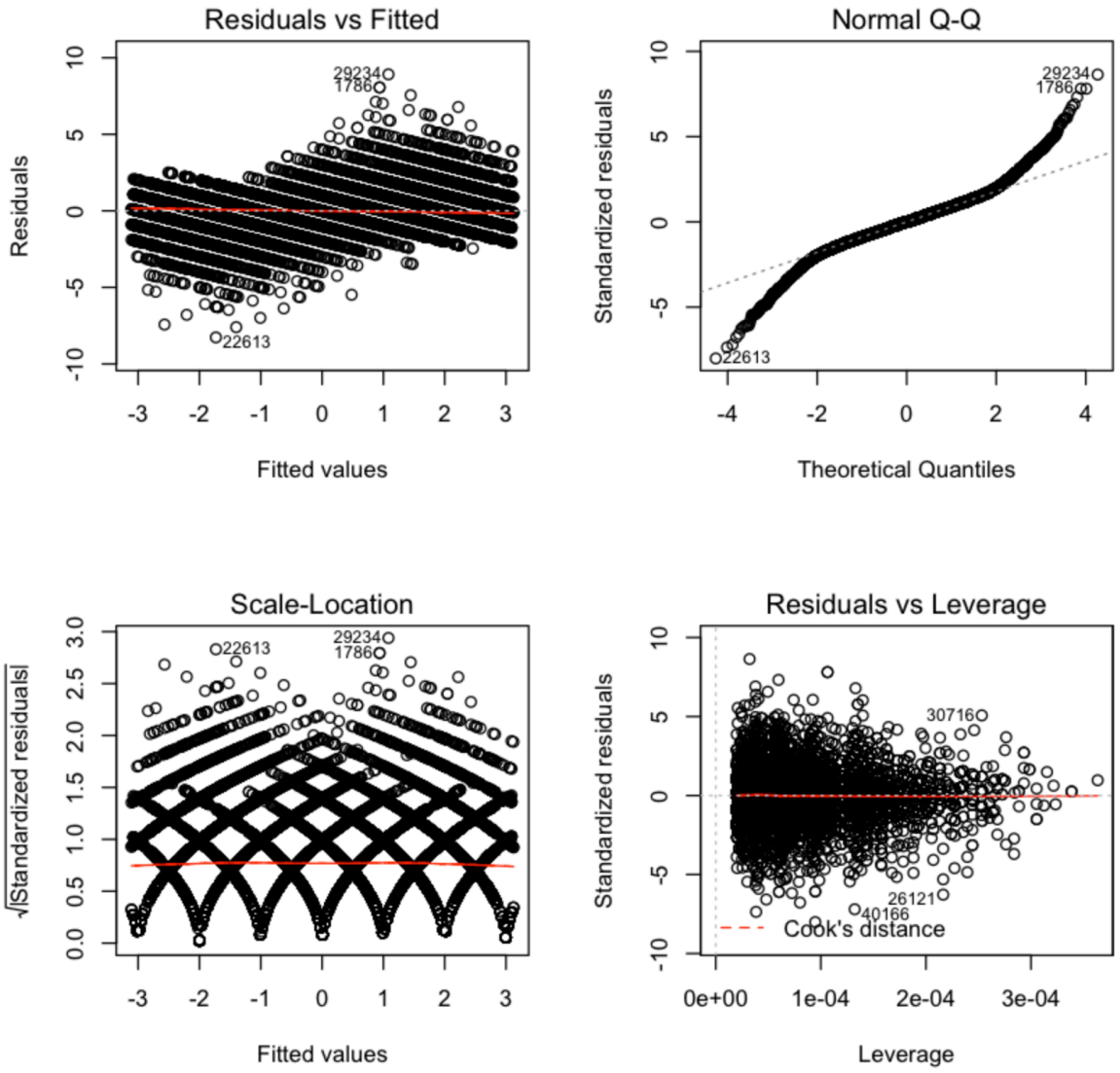
③ Variation of observations around the regression line(the residual SE) is constant

④ For given value of X,Y values (or the error) are Normally distributed

The first assumptions base on the knowledge of study design or data collection

The left 3 assumptions could be check by examinin the residuals or errors.

Language R could print such 4 diagram for the model I built by the plot method.



For the first diagram, no pattern between Residuals and Fitted values, the red line is flat.

For second diagram, we can find out whether the expected residuals are normally distributed. If the Y-values is, the plot should be on definitely a diagonal line. We can see that from -2 to 2, the line meet the conditions

Third and Forth diagram could help us to identify non-linear relation and other troubles.

Finding violations with Library(gvlma) in R

```
library(faraway)
vif(mymodel)
```

diff_towerKills	1.89529039636018
diff_baronKills	1.31361320828136
diff_dragonKills	1.64275707629374

Rank the most significant predictor variable.

①One possibility is to measure the importance of a variable by the magnitude of its regression coefficient. This approach fails because the regression coefficients depend on the underlying scale of measurements.

towerKills, baronKills, dragonKills have different scales.

②Another possibility is to measure the importance of a variable by its observed significance level (P value). However, the distinction between statistical significant and practical importance applies here, too. Even if the predictors are measured on the same scale, a small coefficient that can be estimated precisely will have a small P value, while a large coefficient that is not estimate precisely will have a large P value.

** So I try to standardize each variable in the model then compare the coefficient of them.

```
mymodel_s <- lm(scale(games$diff_inhibitorKills) ~ scale(games$diff_towerKills) +
                scale(games$diff_baronKills) + scale(games$diff_dragonKills))
summary(mymodel_s)
```

```
Call:
lm(formula = scale(games$diff_inhibitorKills) ~ scale(games$diff_towerKills) +
    scale(games$diff_baronKills) + scale(games$diff_dragonKills))

Residuals:
    Min      1Q  Median      3Q     Max
-3.6147 -0.2740  0.0018  0.2774  4.0003

Coefficients:
                              Estimate Std. Error t value Pr(>|t|)
(Intercept)                 -2.344e-16  2.122e-03   0.000        1
scale(games$diff_towerKills)  8.169e-01  2.922e-03 279.599   <2e-16 ***
scale(games$diff_baronKills)  8.962e-02  2.432e-03  36.848   <2e-16 ***
scale(games$diff_dragonKills) 2.476e-02  2.720e-03   9.102   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.4756 on 50223 degrees of freedom
Multiple R-squared:  0.7738,    Adjusted R-squared:  0.7738
F-statistic: 5.727e+04 on 3 and 50223 DF,  p-value: < 2.2e-16
```

II. Logistic Linear Model

Logistic Linear Model is a more suited model for game data because each match has a result about win&lose.

The meaning of a single variable model is just like how much tower difference it need to gain a winning result. I'm going to build up a model with previous three variables and take win/lose as y.

```
predictors = ['diff_towerKills','diff_dragonKills','diff_baronKills']
model1 = sm.Logit(games_linear['winner'],games_linear[predictors]).fit()
model1.summary()
```

```
Optimization terminated successfully.
         Current function value: 0.079694
         Iterations 10
```

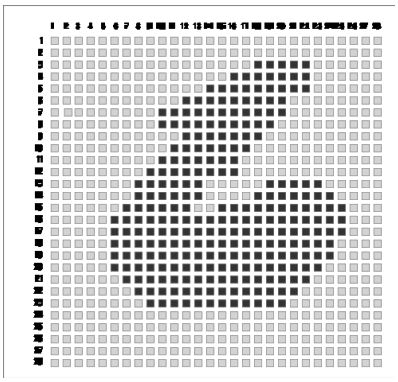
Logit Regression Results

Dep. Variable:	winner	**No. Observations:**	50227
Model:	Logit	**Df Residuals:**	50224
Method:	MLE	**Df Model:**	2
Date:	Thu, 02 Nov 2017	**Pseudo R-squ.:**	0.8850
Time:	13:58:30	**Log-Likelihood:**	-4002.8
converged:	True	**LL-Null:**	-34811.
		LLR p-value:	0.000

	coef	std err	z	P>|z|	[0.025	0.975]
diff_towerKills	1.0759	0.015	70.698	0.000	1.046	1.106
diff_dragonKills	-0.0525	0.015	-3.516	0.000	-0.082	-0.023
diff_baronKills	0.4853	0.028	17.293	0.000	0.430	0.540

III. Convolutional Neural Network.

The reason I chose CNN model to do deep learning in my data is that there do have similarity between a image(which is CNN usually applied to) and game data.



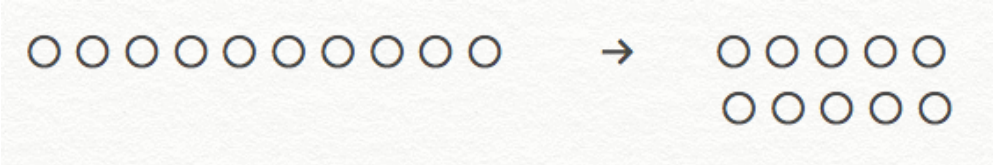
In MNIST data, a picture including a hand written number is represent in 28*28 pixel, 1 and 0 represent the black and white, so there are totally 28*28*2 in one piece of data. If we apply traditional neural network on this, we got 784 input numbers and times of parameter in the first layer. Convolution and maxpool is in charge of make the number

of input decreased, but also guarantee the simplified input can keep the features of the original picture.

As the game data always got a complicated dimension of data. CNN could be applied to game data after the game data is reshaped.

winner	gameDuration	firstBlood	firstTower	firstInhibitor	firstBaron	firstDragon	firstRiftHerald	t2_towerKills	t2_inhibitorKills	t2_baronKills	t2_dragonKills	t2_riftHeraldKills
0	1949	0	1	1	1	1	0	5	0	0	1	1
0	1851	1	1	1	0	1	1	2	0	0	0	0
0	1493	0	1	1	1	0	0	2	0	0	1	0
0	1758	1	1	1	1	1	0	0	0	0	0	0
0	2094	0	1	1	1	1	0	3	0	0	1	0
0	2059	0	0	1	1	0	0	6	0	0	3	0
0	1993	1	0	1	1	1	1	2	0	0	0	0

data in red is the 10 column of data in one match, I reshape the data in 2*5 column.



```
data = np.genfromtxt("games0.csv",delimiter=",",missing_values="?",filling_values=1.)
#Separate the X
dataX = data[:,2:-1]
#Separate the Y
##first column record whether the team win the game
pre_dataY = data[:,0]
#Convert the Y to one hot
dataY = np.zeros((pre_dataY.size,2))

# lose -> column[0] = 1, win -> column[1] = 1
for i in range(len(pre_dataY)):
    if pre_dataY[i] == 0:
        dataY[i][0] = 1
    else:
        dataY[i][1] = 1
```

```
dataY.shape
```

```
(51536, 2)
```

```
dataX = dataX.reshape([-1, 2, 5, 1])
dataX.shape
```

There are 51536 piece of data in total, I got to divide them into 85% train data and 15% test data.

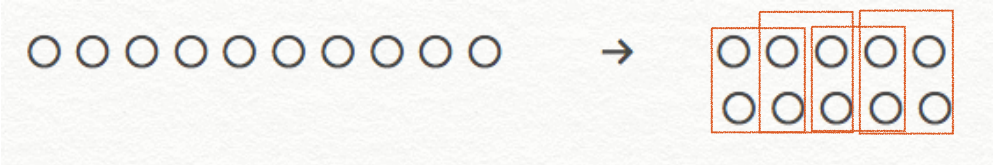
```
#Number of data points used for testing
num_test = int(0.15 * len(data))

#Split data into train and  test
trainX = dataX[:-num_test]
testX = dataX[-num_test:]

trainY = dataY[:-num_test]
testY = dataY[-num_test:]
```

```
trainY.shape
```

```
(43806, 2)
```

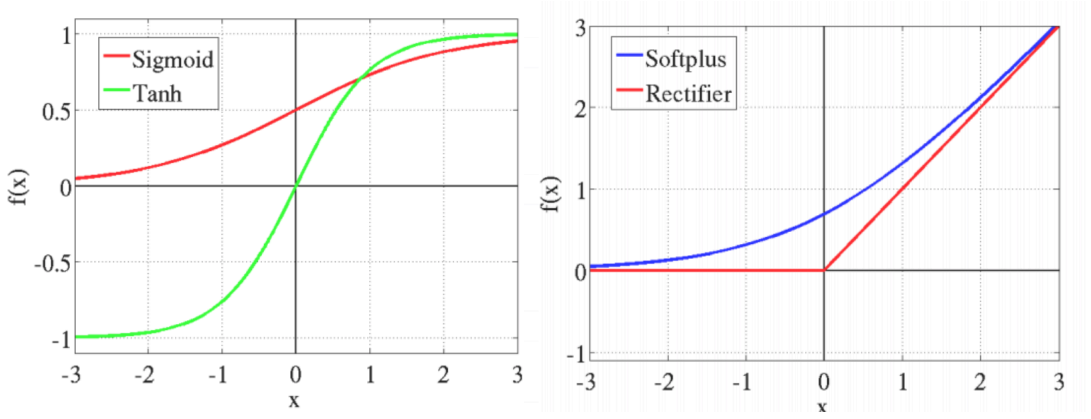


Then the filter for the convolution layer will be 2 and gain 4 data out for each layer.

1 Convolution Layer + 1 maxPool Layer + 2 fully connected Network Layer

```
network2 = input_data(shape=[None, 2, 5, 1], name='input')
network2 = conv_2d(network2,4,2,activation='relu',regularizer="L2")
network2 = max_pool_2d(network2,1)
network2 = fully_connected(network2, 16, activation='tanh')
#network1 = dropout(network1,0.8)
network2 = fully_connected(network2, 2, activation='tanh')
network2 = regression(network2, optimizer='adam', learning_rate=0.01,loss='mean_square', name='target')
```

"relu" function is applied as the activation function for the first convolution layer. "Tanh" is applied for 2 fully_connected network.



```
model = tflearn.DNN(network2, tensorboard_verbose=0)
model.fit({'input': trainX}, {'target': trainY}, n_epoch=20,
          validation_set=({'input': testX}, {'target': testY}),
          snapshot_step=100, show_metric=True, run_id='Ruohua Yin')
```

```
Training Step: 13699  | total loss: 0.05506 | time: 7.837s
| Adam | epoch: 020 | loss: 0.05506 - acc: 0.9278 -- iter: 43776/43806
Training Step: 13700  | total loss: 0.05410 | time: 8.876s
| Adam | epoch: 020 | loss: 0.05410 - acc: 0.9272 | val_loss: 0.04937 - val_acc: 0.9414 -- iter: 43806/43806
--
```

The accuracy is 92.72% for the model, 94.14% for the validation data(train data)

IV. Conclusion

The accuracy review that it's quiet a good model implemented with CNN, but the result is reasonable and unsurprising because actually I'm using the in-game data and corresponding result about win& lose to build up the model. Which it's destination is not a prediction model, it's just like when I know all the 10 kinds of data, in other words, If I'm told these things entirely before the end of the game except for the game result, The model can tell you the result with a accuracy over 92%.

V. Reference

[1] http://data-speaks.luca-d3.com/2017/05/machine-learning-to-analyze-league-of.html

Machine Learning to analyze League of Legends

[2] https://www.kaggle.com/erenan/basic-data-exploration-matches-2014-2017 Basic data exploration: matches 2014-2017 [3]https://www.leagueofgraphs.com/ League of Graph

[3] https://developer.riotgames.com/api-methods/#lol-static-data-v3/GET_getChampionList

Riot API instructions

[4] https://towardsdatascience.com/tagged/league-of-legends