INFO 6205 Ranking System Project Report

Bo Cao

Task Requirement

Your task is to develop a ranking system which is able to evaluate the following expression where xi, xj are elements from a set of competing elements X:

$$P(xi, xj)$$

where P(xi, xj) is the probability that xi would beat xj if they met in a head to head matchup at neutral territory.

A by-product of this expression would be the ability to build a table of elements ordered such that if an element xi appears above another element xj, then you may infer that $P(xi, xj) > 0$. Note that there may be a set of undecidable cycles (like rock, paper, scissors) in which case you will have to mark those elements as equal in your table.

It is desirable also that the value of P is not just a single number, but a probability density function (pdf), in other words there should perhaps be some sort of bounds on the P, maybe a uniform pdf between two values. As you get more data, the bounds will become narrower (i.e. your precision will improve). The input to your system will be a set of prior encounters with a result. These results can be win-loss or they can be scores, as in for example the English Premier League (EPL). If appropriate, you may also consider home team advantage if you feel that it matters.

Your choice of element is entirely up to you. There may be bonus points for originality. However, My recommendation would be to concentrate on the EPL, especially in light of COVID-19 which has abruptly terminated (or at least postponed) the season. Although Liverpool must end the season at the top of the table (it is mathematically impossible for any other team to pass them), the next five positions are important for the summer, as are the last three positions, which teams will be relegated.

It is desirable that the order of data input not affect the result.

Preliminary knowledge

1. Poisson distribution

Poisson distribution is a discrete probability distribution commonly found in statistics and probability. It was published by French mathematician Siméon-Denis Poisson in 1838.

Probability function of the Poisson distribution is:

$$P(X=k)=\frac{\lambda^k}{k!}e^{-\lambda},k=0,1,\cdots$$

Parameter λ of the Poisson distribution is the average number of random events per unit time (or unit area). The Poisson distribution is suitable for describing the number of random events that occur per unit time.

Both the expectation and variance of the Poisson distribution are λ.

The characteristic function is

$$\psi(t)=exp\{\lambda(e^{it}-1)\}.$$

An examples of Probability function of Poisson distribution images:

2. CSV File

Comma-separated values (CSV, sometimes referred to as character-separated values, because the separator character can not be a comma), its file stores the table data (numbers and text) in plain text. Plain text means that the file is a sequence of characters and contains no data that must be interpreted like a binary number. The CSV file consists of any number of records, separated by some newline character; each record is composed of fields, and the separators between fields are other characters or strings, and the most common are commas or tabs. Generally, all records have the exact same sequence of fields.

3. EPL

The Premier League, often referred to as the English Premier League or the EPL outside England, is the top level of the English football league system. Contested by 20 clubs, it operates on a system of promotion and relegation with the English Football League (EFL). Seasons run from August to May with each team playing 38 matches (playing all 19 other teams both home and away). Most games are played on Saturday and Sunday afternoons.

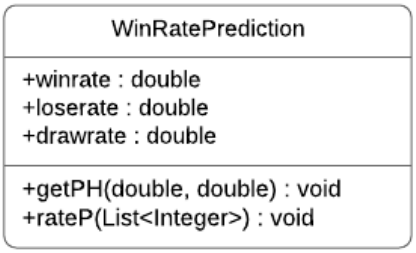
Project description

This project wants to achieve two core functions：

- Calculate and guess the probability of winning any two teams when encountering through the historical game records between each team.
- Based on the win rate between any two teams to simulate the 2020 EPL unfinished game and output the ranking results of the competition team.

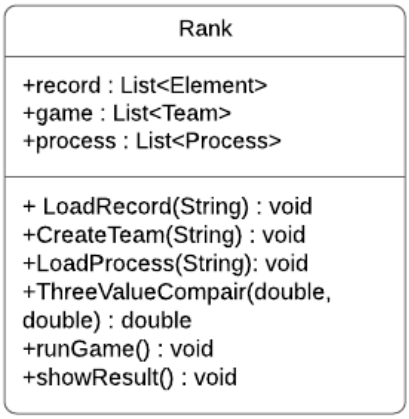
The above functions are implemented by *WinRatePrediction* class and *Rank* class.

In the *WinRateProdiction* class, the historical game records of any two teams are calculated by probability density function (pdf) of Poisson distribution to obtain the probability density of three cases (“win”, “lose”, and “draw”). Find the probabilities of the three situations by definite integrals and use them as predicted probabilities.



Class Diagram of *WinRatePrediction*

In the *Rank* class, by reading the historical game records of any two teams, the schedule and the list of participating teams, the entire game process will be simulated according to the calculated predicted win rate, and finally ranked by the points of each team.



Class Diagram of *Rank*

Element, *Process* and *Team* are three auxiliary classes. *Element* class stores the historical game records of any two teams, corresponding to the *record.csv* file; *Process* class stores the game process, corresponding to the *process.csv* file; *Team* class stores the team information, corresponding to the *team.csv* file.

```
4 Tottenham,Chelsea,0,0,0,0
5 Chelsea,Man United,0,0
6 Man United,Chelsea,2,2
7 Chelsea,Leicester,1,1
8 Leicester,Chelsea,1,1
9 Chelsea,Norwich,2
10 Norwich,Chelsea,0
11 Chelsea,Sheffield United,1
12 Sheffield United,Chelsea,1
13 Chelsea,Wolves,2
14 Wolves,Chelsea,0
15 Chelsea,Brighton,2,1
16 Brighton,Chelsea,0,1
17 Chelsea,Southampton,2,0
18 Southampton,Chelsea,0,2
19 Chelsea,Newcastle,2,0
20 Newcastle,Chelsea,0,2
21 Chelsea,Burnley,2,2
22 Burnley,Chelsea,0,0
23 Chelsea,Watford,2
24 Watford,Chelsea,0
25 Chelsea,Crystal Palace,2
26 Crystal Palace,Chelsea,0
27 Chelsea,Man City,0
28 Man City,Chelsea,2
29 Chelsea,West Ham,0
30 West Ham,Chelsea,2
31 Chelsea,Aston Villa,2
```

record.csv

```
3 Bournemouth,Leicester
4 Bournemouth,Tottenham
5 Bournemouth,Crystal Palace
6 Arsenal,Liverpool
7 Arsenal,Norwich
8 Arsenal,Leicester
9 Arsenal,Watford
10 Aston Villa,Arsenal
11 Aston Villa,Sheffield United
12 Aston Villa,Wolves
13 Aston Villa,Crystal Palace
14 Aston Villa,Man United
15 Aston Villa,Chelsea
16 Brighton,Man United
17 Brighton,Man City
18 Brighton,Liverpool
19 Brighton,Arsenal
20 Brighton,Newcastle
21 Burnley,Sheffield United
22 Burnley,Wolves
23 Burnley,Watford
24 Burnley,Brighton
25 Chelsea,Watford
26 Chelsea,Norwich
27 Chelsea,Man City
28 Chelsea,Wolves
29 Crystal Palace,Man United
30 Crystal Palace,Chelsea
31 Crystal Palace,Tottenham
```

process.csv

```
1 Liverpool,82
2 Man City,57
3 Leicester,53
4 Chelsea,48
5 Man United,45
6 Wolves,43
7 Sheffield United,43
8 Tottenham,41
9 Arsenal,40
10 Burnley,39
11 Crystal Palace,39
12 Everton,37
13 Newcastle,35
14 Southampton,34
15 Brighton,29
16 West Ham,27
17 Watford,27
18 Bournemouth,27
19 Aston Villa,25
20 Norwich,21
```

team.csv

Project display

- Code display

```java
import java.io.BufferedReader;

public class Rank implements Comparator<Team>{
    public List<Element> record = new Vector<>();;
    public List<Team> game = new Vector<>();
    public List<Process> process = new Vector<>();

    public void LoadRecord(String recordpath) {
        File IRfile = new File(recordpath);

        if(IRfile.exists())
        {
            if(IRfile.length() != 0)
            {
                FileReader IRfr = null;
                try {
                    IRfr = new FileReader(IRfile);
                } catch (FileNotFoundException e) {
                    e.printStackTrace();
                }
                BufferedReader IRbr = new BufferedReader(IRfr);
                String rline;
                try {
                    while ((rline = IRbr.readLine()) != null) {
                        Element e = new Element(rline);
                        record.add(e);
                    }
                } catch (IOException e) {
                    e.printStackTrace();
                }
                try {
```

```java
public class WinRatePrediction {
    public double winrate;
    public double loserate;
    public double drawrate;
    public static final double e = Math.E;

    public double getPH(double x, double n) {
        double l = 1.0;
        if(x != 0.0)
            for(double i = 1.0; i <= x; i+=1.0)
                l = l * i;
        return Math.pow(n, x) * Math.pow(e, -1*n) / l;
    }

    public void rateP(List<Integer> num) {
        double sum2 = 0;
        double sum1 = 0;
        double sum0 = 0;

        for(Integer i : num)
        {
            if(i == 2)
                sum2 += 1.0;
            else if(i == 1)
                sum1 =+ 1.0;
            else
                sum0 += 1.0;
        }

        double mean = (sum2 + sum1 + sum0) / 3;
```

```java
public class Element {

    public String fteam;
    public String lteam;
    public double fteamWinrate;
    public double lteamWinrate;
    public double drawrate;
    public List<Integer> frecord;

    public WinRatePrediction w;

    public Element(String s) {
        String[] line = s.split(",");
        fteam = line[0];
        lteam = line[1];
        frecord = new Vector<>();
        for(int i = 2; i < line.length; i++)
            frecord.add(Integer.parseInt(line[i]));
        fteamWinrate = 0.0;
        lteamWinrate = 0.0;
        drawrate = 0.0;
        w = new WinRatePrediction();
    }

    public void predictRate() {
        w.rateP(frecord);
        fteamWinrate = w.winrate;
        lteamWinrate = w.loserate;
        drawrate = w.drawrate;
    }
```

```java
package edu.neu.info6205.RankingSystem;

public class Team {

    public  String teamName;
    public int totalScore;

    public Team(String name, int score) {
        this.teamName = name;
        this.totalScore = score;
    }

    public Team(String s) {
        String[] line = s.split(",");
        this.teamName = line[0];
        this.totalScore = Integer.parseInt(line[1]);
    }

    public void addscore(int n) {
        totalScore += n;
    }
}
```

```java
package edu.neu.info6205.RankingSystem;

public class Process {

    public String team1;
    public String team2;

    public Process(String s) {
        String[] line = s.split(",");
        team1 = line[0];
        team2 = line[1];
    }
}
```

- Result display

```
1. Liverpool, 109
2. Man City, 78
3. Leicester, 71
4. Chelsea, 64
5. Man United, 64
6. Tottenham, 57
7. Wolves, 56
8. Sheffield United, 56
9. Arsenal, 55
10. Crystal Palace, 52
11. Newcastle, 52
12. Burnley, 50
13. Southampton, 49
14. Everton, 46
15. Brighton, 37
16. West Ham, 37
17. Bournemouth, 36
18. Watford, 32
19. Aston Villa, 29
20. Norwich, 28
```

- Project deficiencies

The project only uses the four-year game record as a benchmark. If the record is increased, the accuracy of the prediction probability will be greatly improved.
If you use a database instead of a CSV file, it will facilitate data processing and storage.
If the interval of each part is shortened when the probability density function is integrated, the accuracy of the probability will be improved.