A matrix-based ranking method with application to tennis

Geir Dahl, University of Oslo, 2012

Summary by:

Abhishek Rawat
Pushkar Sathe
Ankush Karmarkar

Chennai Mathematical Institute
April, 2019

1 Introduction

In this paper, a linear algebra based method for ranking players in sports(tennis) is introduced. In tennis, the existing method which is used by the Association of Tennis Professionals (ATP), does not take into account the quality of the player defeated. The method discussed in this method will take that into account.

We now introduce the framework of the ranking methodology used in this paper. Consider a set V of n players and a set of m matches between these players. Consider a directed graph $G = (V, E)$ where each vertex in vertex set V represents player and directed edges in edge set E represent the matches. In a match between i and j, the graph contains edge (i, j) if i wins, and edge (j, i) if j wins. If there is no edge between i and j, it means that these two players have not played against each other. To account for more than one match between i and j, a non-negative number a_{ij} is associated to each edge. The number a_{ij} is the number of matches between i and j where i won. This graph G is called the 'match graph'.

Let $A = [a_{ij}]$ be the $n \times n$ matrix whose entry in position (i, j) is a_{ij}. A is called a match matrix. A may be viewed as the weighted incidence matrix of the graph G equipped with edge weights a_{ij} $(i, j \leq n)$. The goal of this paper is to compute a score for each player based on match matrix A.

2 The ranking method and its properties

2.1 *The score equations*

Let M be the set of matches considered. M may be partitioned into

$$M = \bigcup_{i \neq j} M(i, j)$$

where $M(i, j)$ is the set of matches in which i defeated j. $|M(i, j)| = a_{ij} \geq 0$. A number $\beta_m \geq 0$ is associated with every match $m \in M$. β_m is called the weight of m and it measures the importance of match m.

The ranking method finds a score x_i for each player $i \leq n$. These scores are determined by a linear system of equations containing one equation for each player. The number $\alpha \geq 0$ is a bonus parameter such that αx_j denotes the percentage of the score of the opponent j that player i defeated. We call a vector $x = (x_1, x_2, ..., x_n) \in \mathbb{R}^n$ a score vector if it satisfies the following system of linear equations

$$x_i = \sum_{j \neq i,\ m \in M(i,j)} (\beta_m + \alpha x_j) \quad (i \leq n). \tag{1}$$

Claim: The score vector x defined above is unique.

$$\begin{aligned} x_i &= \sum_{j\neq i,\ m\in M(i,j)} (\beta_m + \alpha x_j) \\ &= b_i + \alpha \sum_{j\neq i,\ m\in M(i,j)} x_j \end{aligned}$$

where $b_i = \sum_{j\neq i,\ m\in M(i,j)} \beta_m$ is the total weight of the matches that player i has won.

Thus, (1) may be written in matrix form as

$$x = \alpha A x + b \tag{2}$$

This is an $n \times n$ system of linear equations which are called the score equations. Let $C_\alpha = I - \alpha A$, where I is the $n \times n$ identity matrix. From (2), we get

$$C_\alpha x = b.$$

Let $r_i(A)$ be the i^{th} row sum which denotes the total number of matches won by player i. Similarly, let $s_i(A)$ be the i^{th} column sum which denotes the total number of matches lost by player i. γ_A is then defined as -

$$\gamma_A = \min\{\max_i r_i(A),\ \max_i s_i(A)\} \tag{3}$$

Definition(M-matrix): A square matrix F is called an M-matrix if $F = sI - H$ for some non-negative matrix H and $s > \rho(H)$ where $\rho(H)$ is the spectral radius of H. □

Theorem 1: Let $0 \leq \alpha < \frac{1}{\gamma_A}$. Then the following holds:

(i) C_α is an M-matrix, in particular, it is invertible and has a non negative inverse.

(ii) There is a unique solution x of the score equations (2), and this vector satisfies $x \geq b$.

Proof:

(i) We know that $C_\alpha = I - \alpha A$ and $A \geq 0$. To show that C_α is an M-matrix, we need to estimate $\rho(A)$, where $\rho(A)$ is the spectral radius of A.
$\therefore \rho(A) = \max\{|\lambda| : \lambda \text{ is an eigenvalue of } A\}$

> Recall:
> *Gershgorin's disk theorem:*
> Let B be a square matrix. Let D_k denote the set of numbers that are less than or equal to $\sum_{j=1}^{m} |a_{kj}|$ and D'_k denote the set of numbers less than or equal to $\sum_{j=1}^{m} |a_{jk}|$.
> Let λ be an eigenvalue of B, then $\lambda \in D_k$ for some k and $\lambda \in D'_k$ for some k.

$\therefore$ $|\lambda|$ is less than or equal to the minimum of maximum of sums of rows or columns of matrix A. i.e.

$$\begin{aligned} |\lambda| &\leq \min\{\max_i r_i \, , \max_i s_i\} \\ |\lambda| &\leq \gamma_A \end{aligned}$$

Now, consider

$$\begin{aligned} \rho(\alpha A) &= \alpha\rho(A) \\ &= \alpha|\lambda| \\ &\leq \alpha\gamma_A \\ &< 1 \qquad\qquad \text{because } 0 \leq \alpha < \frac{1}{\gamma_A} \end{aligned}$$

Hence, as the coefficient of I is greater than $\rho(\alpha A)$, C_α is an M-matrix.

(ii) As C_α is an M-matrix, it is invertible and has a non-negative inverse. We know that

$$\begin{aligned} C_\alpha x &= b \\ \implies x &= C_\alpha^{-1} b \\ \because \; C_\alpha^{-1} b &> 0 \qquad \text{As both } C_\alpha^{-}1 \text{ and } b \text{ are non-negative} \\ \therefore \; x &> 0 \end{aligned}$$

Consider (2)

$$x = \alpha A x + b$$

As all terms are unique and non-negative, there is a unique solution for square equations and this vector satisfies $x \geq b$. □

In theorem 1, we have assumed that $\alpha\gamma_A \leq 1$. This can be justified as follows. Consider a player j. This player has lost s_j matches, and his contribution Δ_j to the other players is-

$$\Delta_j = \sum_i (\alpha a_{ij} x_j) = \alpha x_j \sum_i a_{ij} = \alpha x_j s_j$$

Now, we can assume that $\Delta_j < x_j$, i.e., the player increases other players' scores by an amount less than his own score.

Thus, $\alpha x_j s_j < x_j \implies \alpha s_j < 1$

$\implies \alpha \max_j s_j < 1 \qquad \because \alpha s_j < 1$ should hold for all j

Hence, reasonable choice of α should satisfy $\alpha < \frac{1}{(\max_j s_j)} \leq \frac{1}{\gamma_A}$.

Hence, to assure that no player contributes bonus points that are more than p times his score, α can be selected as

$$\alpha < \frac{p}{\gamma_A} \, ,$$

where $p \leq 1$. □

2.2 Combinatorial interpretations

Theorem 2: Assume that $\alpha\rho(A) < 1$ (which holds if $\alpha < \frac{1}{\gamma_A}$), so C_α is invertible, and let $C_\alpha^{-1} = [d_{ij}(\alpha)]$. Then

$$d_{ij}(\alpha) = \sum_{P \in \mathcal{P}_{ij}} \alpha^{l(P)} \qquad\qquad (i, j \leq n) \ ,$$

where -
P is a walk, i.e., a sequence $P : e_1\ e_2, ...,\ e_s$ of consecutive edges such that the terminal vertex of edge e_p is the initial vertex of edge e_{p+1} $(p < s)$,
$l(P)$ is the length of walk p which is the number of edges, and
$\mathcal{P}_{ij}$ denotes the set of walks from i to j in G.
Proof: Consider the coefficient matrix $C_\alpha = I - \alpha A$ of the score equations. If we assume that $\alpha\rho(A) < 1$, then C_α is invertible and its inverse is the sum of the Neumann series $I + \alpha A + \alpha^2 A^2 + ...$, i.e.

$$C_\alpha^{-1} = (I - \alpha A)^{-1} = I + \alpha A + \alpha^2 A^2 + \alpha^3 A^3 + ...$$

Let the $(i, j)^{th}$ entry of the power matrix A^k be denoted $a_{ij}^{(k)}$. $a_{ij}^{(k)}$ represents the number of distinct walks of length k from vertex i to vertex j in graph G.
Thus,

$$C_\alpha^{-1} = [d_{ij}(\alpha)] = I + \alpha A + \alpha^2 A^2 + \alpha^3 A^3 + ...$$

As the $(i, j)^{th}$ entry of A^k is $a_{ij}^{(k)}$ and $l(P)$ is the length of walk P, it follows that -

$$d_{ij}(\alpha) = \sum_{P \in \mathcal{P}_{ij}} \alpha^{l(P)}$$

□

Corollary: Assume that $\alpha\rho(A) < 1$, and let $x = (x_1, x_2, ..., x_n)$ be the unique solution of the score equations (2). Then

$$x_i = \sum_{j=1}^{n} b_j \sum_{P \in \mathcal{P}_{ij}} \alpha^{l(P)} \qquad\qquad (i \leq n).$$

This shows that the score x_i of player i is the sum of the player's match score b_i and bonus score. The bonus score is damped by the factor $\alpha^{l(P)}$ where the exponent is the length of the win sequence that the walk P represents.
The contribution of player j to the score of player i is $b_j d_{ij}(\alpha)$ where $d_{ij} = \sum_{P \in \mathcal{P}} \alpha^{l(P)}$.

3 Single-elimination tournaments

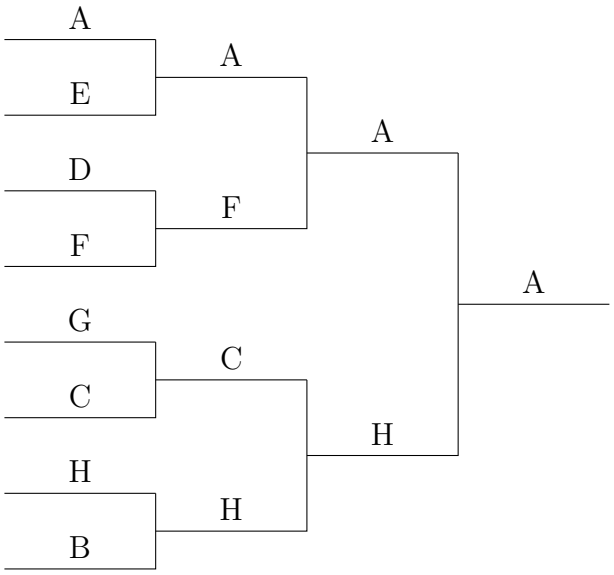
Consider a special case in which players take part in a single-elimination tournament. For simplicity, let us assume that n players take part in the tournament, where $n = 2^k,\ k \in \mathbb{N}$. In such a tournament, a draw is made beforehand and these n players pair

up according to the draw in $n/2$ matches. The winners advance to the next round while the losers are eliminated. In the next round, the remaining $n/2$ players again play a total of $n/4$ matches. Again, the winners advance to the next round and the losers are eliminated. Eventually, there are four quarter-finals, two semi-finals and the final.

Now, consider that the match graph $G = (V, E)$ is determined by the outcome of one single elimination tournament where these n players participate. As each player, except the tournament winner loses exactly one match, each vertex haas indegree 1, while the winner vertex has indegree 0. This is also reflected in the match matrix, whis is a $(0, 1)$ matrix. The column corresponding to the winning player is a zero vector, while that of all other players contain a single 1.

Example: Let $k = 3$, i.e. there are 8 players. Let the initial points of players A to H be 300, 300, 275, 260, 255, 220, 200, 190 respectively.

Let the results of the tournament be as follows -



The corresponding match matrix A for such a tournament is -

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

□

There are various methods of obtaining the final points for single elimination tournaments. We can either use the direct methods like singular value decomposition(S.V.D.) or iterative methods like Gauss-Seidel.

(a) The python code for SVD is given below:-

```
import numpy as np
import pandas as pd

def SVD(A,b,alpha,players):

    n=len(players)
    I=np.identity(n,dtype=float) #identity matrix
    C=I-alpha*A

    U,s,V = np.linalg.svd(C) # SVD decomposition of A , s=sigma

    # Solving Cx=b using the equation V'x=w
    # V'=transpose(V) ,U'=transpose(U)
    # w computed from sigma.w=D
    # x computed using V'x=w

    D = np.dot(U.T,b) # D = U^t*b
    w = np.linalg.solve(np.diag(s),D) # w = V^t*D
    x = np.linalg.solve(V,w) # V'x=w

    df=pd.DataFrame()
    df["Players"]=players
    df["Initial"]=list(map(int,b)) #initial points
    df["Final"]=list(map(int,x)) #final points after all matches
    print(df)

# This is the type of input we need to give for all codes below

A = np.array([[0.,0.,0.,0.,1.,1.,0.,1.],
              [0.,0.,0.,0.,0.,0.,0.,0.],
              [0.,0.,0.,0.,0.,0.,1.,0.],
              [0.,0.,0.,0.,0.,0.,0.,0.],
              [0.,0.,0.,0.,0.,0.,0.,0.],
              [0.,0.,0.,1.,0.,0.,0.,0.],
              [0.,0.,0.,0.,0.,0.,0.,0.],
              [0.,1.,1.,0.,0.,0.,0.,0.]])

players=["A","B","C","D","E","F","G","H"]
b = np.array([300.,300.,275.,260.,255.,220.,200.,190.])
alpha=.25


SVD(A,b,alpha,players) #just calling the function with the given input
```

(b) The python code for Gauss-Seidel method is given below:-

```
import numpy as np
```

```
import pandas as pd

def GaussSeidel(A,b,alpha,players):

    I=np.identity(8,dtype=float) #identity matrix
    C=I-alpha*A #as we want to solve Cx=b

    tolerance=0.0001
    max_iterations=10000

    x_prev=np.zeros_like(b) #some random predecided x

    for i in range(0,max_iterations):

        x_new=np.zeros_like(x_prev)

        for j in range(len(C)):
            s1=np.dot(C[j, :j], x_new[:j])
            s2=np.dot(C[j, j + 1:], x_prev[j + 1:])
            x_new[j]=(b[j]-s1-s2)/C[j,j]

        if np.allclose(x_prev,x_new,rtol=tolerance):
            break #if converges, break out of the loop

        x_prev=x_new

    df=pd.DataFrame()
    df["Players"]=players
    df["Initial"]=list(map(int,b)) #initial points
    df["Final"]=list(map(int,x_new)) #final points after all matches
    print(df)
```

4 Tennis ranking

In men's professional tennis, run by the ATP(Association of Tennis Professionals), players earn ranking points throughout the year basedon their tournament results. In a tournament, a player receives points depending on which round he reaches. These points are accumulated by summing up the points for the last 52 weeks. The points older than that are no longer valid. Tournaments are divided into different categories. The tournaments with the most number of points are the four Grand Slam tournaments: the Australian Open, Roland Garros, the Championships, Wimbledon and the US Open. The winner of each of these gets 2000 points. Below the Grand Slam tournaments there are nine ATP Tour Masters 1000 tournaments (Winner get 1000 points), several ATP Tour 500 tournaments (Winner gets 500 points) and some other smaller tournaments. At the end of each season, a tournament called ATP Finals where just

the best 8 players participate.

The ATP Finals consists of three rounds viz. Group stage, semifinal and final. The eight participants are split into two groups of four. Each player plays a match against every other player in his group, thus playing a total of three matches. The winner of each match receives 200 points. The top two players in each group advance to the semifinals. Winning the semifinal gives an additiional 400 points. The winners of the two semifinals reach the final. The winner of the final receives additional 500 points. Thus, 1500 is the maximum number of points that can be won in this tournament by a single player.

The points distribution of the main tournaments is shown in the table below.

Category	W	F	SF	QF	R16	R32	R64	R128
Grand Slam	2000	1200	720	360	180	90	45	10
ATP Finals	+500 (1500 max)	+400 (1000 max)	200 for each match (600 max)					
Masters 1000	1000	600	360	180	90	45	10	
500 Series	500	300	180	90	45	20		

4.1 Case Study

The case that has been discussed in this paper is the ATP World Tour Finals, 2009 which was held in London. The best 8 players participated in this tournament.
The initial points of Federer, Nadal, Djokovic, Murray, Del Potro, Davydenko, Verdasco and Söderling were 10150, 9205, 7910, 6630, 5985, 3630, 3300, 3010.
The players were divided into two groups. The results of the group stage were:

	Group A			Group B	
Match	Winner	Loser		Winner	Loser
1	Murray	Del Potro		Söderling	Nadal
2	Federer	Verdasco		Djokovic	Davydenko
3	Del Potro	Verdasco		Söderling	Djokovic
4	Federer	Murray		Davydenko	Nadal
5	Murray	Verdasco		Djokovic	Nadal
6	Del Potro	Federer		Davydenko	Söderling

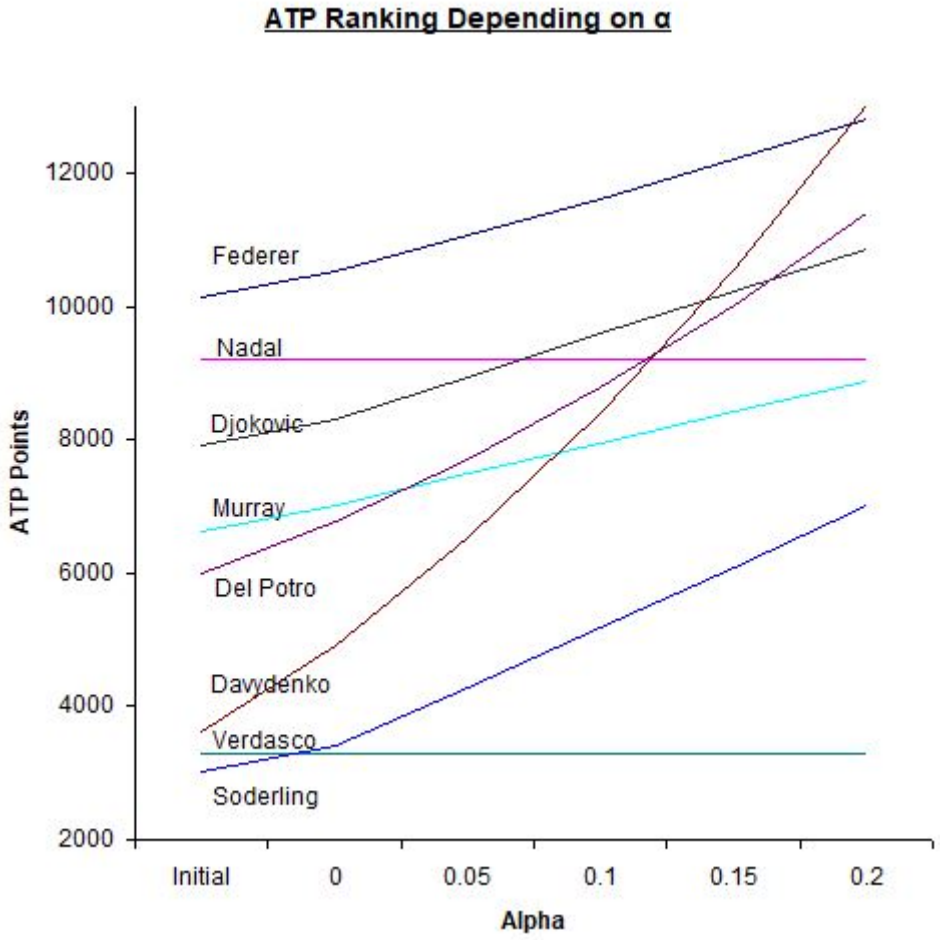
The knock-out results were -

Match	Winner	Loser
Semi-final	Davydenko	Federer
Semi-final	Del Potro	Söderling
Final	Davydenko	Del Potro

The matrix based method for ranking was then applied taking $\alpha = 0, 0.05, 0.1, 0.15, 0.2$. Note that when $\alpha = 0$, the points are the same as ATP points.

		α				
Players	Initial	0	0.05	0.1	0.15	0.2
Federer	10150	10550	11071	11623	12204	12815
Nadal	9205	9205	9205	9205	9205	9205
Djokovic	7910	8310	8952	9593	10235	10877
Murray	6630	7030	7494	7958	8423	8887
Del Potro	5985	6785	7718	8795	10024	11412
Davydenko	3630	4930	6544	8410	10559	13020
Verdasco	3300	3300	3300	3300	3300	3300
Söderling	3010	3410	4285	5177	6089	7018

The graph showing these values is given below.



From this graph, it is very clear that as we increase α, the importance given to the opponent increases. Davydenko who was at rank six at the beginning of the tournament, finishes the tournament at rank 1 when $\alpha = 0.2$. But, when $\alpha = 0$ that is, according to the points system of ATP, Davydenko remains at rank 6 after the tournament.

The Python code to calculate these values is given below.

The program below takes the following parameters as input
1) Players
2) Alpha - a list containing all the alphas if you want to compare them
3) ATP Points - initial ATP points of all the players in order
4) Group Matches - a list of tuples containing the edges of the graph, as defined before, in the correct order of the group matches played (Note: Here the order of matches plays a great impact in defining the final score)
5) Semi Finals - a list of tuples containing the edges of the graph, as defined before, in the correct order of the semi final matches played
6) Finals - a list of tuples containing the edges of the graph, as defined before, of the final match played

This method works step by step, updating the points of players after every match. It then tabulates the points and the ranks of players after all the matches are played.

```
import pandas as pd

def matches(players,alpha,AtpPoints,GroupMatches,SemiFinals,Final):

  n=len(players) #the number of players

  initial=AtpPoints[:] #initial score of players

  df=pd.DataFrame() #to store match points for various alpha
  df["Players"]=players
  df["Initial"]=initial

  df2=pd.DataFrame() #to store ranks for various alpha

  z=list(zip(players,initial))
  sorted=mergesort(z) #in descending order
  ranks=list() #a list of ranks according to various alpha
  temp=[]
  for i in range(0,n):
    temp.append(sorted[i][0])
  ranks.append(temp)
  df2["Initial"]=ranks[0]

  ranks=list()

  x=[GroupMatches,SemiFinals,Final] #list of types of matches
  y=[200,400,500] #list of match points according to match type

  for a in range(0,len(alpha)):
```

```
    AtpPoints=initial[:]
    for i in range(0,len(x)):
      for j in range(0,len(x[i])):
        match=x[i][j]
        won,lost=match #since an edge denotes winner to looser
        AtpPoints[won-1]+=y[i]+round(AtpPoints[lost-1]*alpha[a])
        #this is the scoring system

    df[alpha[a]]=AtpPoints

    #Now ranking
    z=list(zip(players,AtpPoints))
    sorted=mergesort(z) #in descending order
    temp=[]
    for i in range(0,n):
      temp.append(sorted[i][0])
    ranks.append(temp)

    df2[alpha[a]]=ranks[a]

  print(df)
  print("\n")
  print(df2)

#functions for sorting players to rank them
def mergesort(l):
  if len(l) < 2:
    return l
  mid = len(l)//2
  leftlist = l[0:mid]
  rightlist = l[mid:]
  return merge(mergesort(leftlist),mergesort(rightlist))

def merge(l1,l2):
  if l1 == [] or l2 == []:
    return l1 + l2
  if l1[0][1] >= l2[0][1]:
    return [l1[0]] + merge(l1[1:], l2)
  else:
    return [l2[0]] + merge(l1, l2[1:])
```

5 Remarks

The example discussed with tennis can be used by tennis professionals to track and compare their performance with other players. A tennis enthusiast may obtain his "personal ranking" by selecting a value of α. The new ranking can be used as a way of

showing which players are on they way up, or down, on the rankings. This pair wise comparison and ranking can be used in many other application where weightage of the other objects/players/projects etc. has high importance.

References

[1] A Matrix Based Ranking Method with Application to Tennis - Geir Dahl, University of Oslo (2012)

[2] Lloyd N. Trefethen and David Bau, III: Numerical linear algebra, SIAM (1997)

[3] James W. Demmel: Applied numerical linear algebra, SIAM (1997)