Forums

Select a Forum

Search | Watch Thread | My Post History | My Watches | User Settings

View: Flat (newest first) | Threaded | Tree

Previous Thread | Next Thread

Forums Development Forums Marathon Match 121 v1.0 Marathon Match 121

[1 2] NEXT >

Post your approach

Post your approach | Feedback: (+7/-0) | [+] [-] | Reply Thu, Nov 5, 2020 at 12:52 AM BDT

iwashi31
9 posts

Thank you for the interesting problem! I fully enjoyed it :)

My solution is separated into 2 phases, one is estimating attack/defence strengths, the other is predicting each game's score based on estimated strengths.

The summary of my estimating attack/defence strengths is like the following steps.
1. Assign arbitrary values to each strength.
2. Try incrementing/decrementing any strength of any team.
3. Simulate tournament (= calculate expectation values of scored, conceded, and points of each team) and measure the difference between the simulation's result and input values.
4. Undo step 2 if the difference got bigger than before.
5. Iterate step 2 to step 4 enough times.

I predicted each game's score by simulated annealing. The target function to minimize is
{total difference of scored, conceded, and points between input values and the one derived from the answer} - C * log({probability of this answer occurring under the assumption of estimated strengths}).
C is a coefficient and takes a value depending on N.

Re: Post your approach (response to post by iwashi31) | Feedback: (+5/-0) | [+] [-] | Reply Thu, Nov 5, 2020 at 1:26 AM BDT

eulerscheZahl
2 posts

I precomputed the expected number of goals for each integer pair of attack,defence. With a bit of math exact values are easy to fin

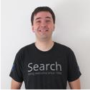
Support

Then I assigned an initial strength of 5 to each team. The strengths are floats here, not integers.

A Markov chain gave me decent values as I simulated the tournament a few times and updated the strengths accordingly to get the statistical number of goals closer to the given one. I used linear interpolation to estimate the goals from the known values for integer pairs.
Just printing the expected number of goals for each match (rounded to the closest integer) should give you something around 91 points on the final scoreboard.

I tried adjusting single matches to take the points into account. There are obvious cases (e.g. in seed 59 the lowest ranked team has 2 draws and 0 wins and it's also clear against which teams the draws happened). Apart from that I had little success to use the information of total points.

Re: Post your approach (response to post by iwashi31) | Feedback: (+3/-0) | [+] [-] | Reply Thu, Nov 5, 2020 at 2:35 AM BDT



mugurelionut
292 posts

My solution consists of the exact two steps as iwashi31, with the 2nd step optimizing almost the _exact_ same function via SA, too :)

Step 1: Estimate attack and defence strengths (SA)

I start from an already good assignment. With a bit of probability and DP I compute for each total number of scored and received goals the probability of the team having each possible attack and defence strength. Then, for each team, I assign initially the attack/defense strength corresponding to the largest probability for its total number of scored/received goals.

I also compute for each match between a team with attack strength A and a team with defence strength D the probability of scoring each number of goals, and based on this, the expected number of goals scored (as a real number).

Then I optimized the attack/defense strengths via SA, using the sum of squared differences between the expected number of goals scored/received and the target number of goals scored/received. A move consisted of changing the attack/defense strength randomly, but in the direction required (e.g. increasing the attack strength if the expected number of scored goals is less than the target, etc.).

The attack/defence strengths determined here are considered fixed for Step 2.

Step 2: Assign results for each match (SA)

I start by assigning the maximum probability result (highest probability number of goals scored on each side). Then I optimize via SA the following function: sum of squared differences between (points, scored goals, received goals) and their targets - C * log(probability of the configuration). A move consisted of randomly changing the score of a random match (from X:Y to X':Y').

log(probability of the configuration) has the nice advantage of being decomposable as the sum of log(probability of each match), so it was nicely usable for SA.

C is a constant which depends on N and X. In the last day I ran multiple tests with multiple values of C in order to pick the best value for each group of cases. This brought me a late score jump of ~0.7-1 percent, but I wish I had started doing this earlier, since I could only consider ~1K tests before the match ended.

Other things I tried:
* For N >= 20 and X <= 2 I replaced Step 2 by assigning the results which maximize the expected score (2 * probability of an exact result + 1 * probability of same outcome) and skipping the SA.
* I tried replacing log(probability of config) with expected score in the SA optimization formula. This worked better in some cases, but I tried this too late to have time to optimize the parameters, so overall it performed worse than using log(probability).

I didn't use simulations at all in any part of my solution.

Re: Post your approach (response to post by iwashi31) | Feedback: (+6/-0) | [+] [-] | Reply Thu, Nov 5, 2020 at 3:30 AM BDT

sullyper
264 posts

I run some simulations at the beginning and train a model (simple linear regression) to predict attack and defense strength.

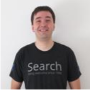
I use those predictions to run more simulations and train another model to predict directly the number of goals that teams A will score against team B.

When X <= 3, I also train a model in parallel to predict the outcome of a match. If the predicted score does not match the outcome, I would then change the predicted score to match the outcome by adding or removing a goal based on what my model confidence was.

For N <= 10 I tried to use a solver to find the most likely solution that matches the number of points/scored/conceded.
If my model predicted that A score 2.7 against B for example, I would allow two options:
- A scored 2 with a probability 0.3 (or a score of -log(0.3)
- A scored 3 with a probability 0.7 (or a score of -log(0.7)

Unfortunately, it seems that my last submission failed on 2 tests (from my score dropped), given I have no failure locally in 2k tests, and that my time limit is set to 9.6 and seems to be correctly respected locally and on TC, I think the failure comes from memory limit, I wish TC was again providing feedback on memory usage on the test input.
That could cost me a lot =(

Re: Post your approach (response to post by sullyper) | Feedback: (+2/-0) | [+] [-] | Reply Thu, Nov 5, 2020 at 4:18 AM BDT



mugurelionut
292 posts

>>> *Unfortunately, it seems that my last submission failed on 2 tests*

I was curious why you kept a significantly lower scoring submission as your final one. In some problems huge score variance can justify this, but it didn't seem to be the case here.

>>> *For N <= 10 I tried to use a solver*

Did you succeed to use a solver? Is that what caused your potential memory issues/test failures?

You can definitely pose the problem as an integer program, so it depends if the solver can find a good solution within the given time and memory limits. If you were successful, then this sounds promising and exciting (up until now I thought it's a nearly impossible task).

But, to be honest, what I actually missed in this problem is not having any ML libraries available. I would have liked to try some prediction models here and there, but I felt the work needed to get something working in the TC eval setup was too much for the uncertainty of the outcome.

Re: Post your approach (response to post by mugurelionut) | Feedback: (+2/-0) | [+] [-] | Reply | Edit Thu, Nov 5, 2020 at 4:43 AM BDT

survival07
17 posts

This was my first marathon participation.

I first tried to apply some mathematical heuristic to find least and maximum number of games a team will lose and based on that find attack and defense strength. Then by greedily create the winning or losing result of a team with the team having highest point left to be the winner. This approach got me provisional score of approx 13

then I learned about Simulated Annealing and applied a naive SA by optimizing a cost function which is weighted sum of squared difference of (total goals scored, total goals conceded, total points made) with the input tuple of these values. The weights that I set were (0.25, 0.25, 0.50) for the above three values.This method got me 63.33. In this approach I randomly assigned attack and defense strength of each team from the possible range.

Thanks everyone for sharing your solutions. I am learning a lot from this forum.

Re: Post your approach (response to post by mugurelionut) | Feedback: (+1/-0) | [+] [-] | Reply Thu, Nov 5, 2020 at 4:47 AM BDT

sullyper
264 posts

Yep, that was my solver, custom made ILP solver.
From both submission I switched from a DFS approach to a BFS approach.
- DFS Use less memory but usually take longer to find the optimal
- BFS stops as soon as it finds a solution

Locally BFS was slightly better because sometimes the DFS would find a non optimal solution which was actually really unlikely and was worst. BFS guarantee to change the output only if I find the best solution (which obviously does not always find).

I shouldn't have switched =/

Given the code is the same, the idea is the same, just the solving method, those 2 pts are definitely a failure, now there are two options:
- memory deallocation takes too long, possible, but I keep a lot of margin, and I have some time check locally.
- too much memory, I also have a check locally, but it's less accurate, it takes snapshot of the memory and I am not confident it works well. That's why it would be nice if TC could give feedback on the test examples that I can compare.

Anyway, I shouldn't have submitted in the last 5min knowing I wouldnt get the result, the expected gain was like 0.2, probably not worth it =/

Re: Post your approach (response to post by eulerscheZahl) | Feedback: (+1/-0) | [+] [-] | Reply Thu, Nov 5, 2020 at 10:47 AM BDT

dimkadimon
4478 posts

Can you share your maths that computes the expected number of goals for each attack/defence pair? Personally I would just run many simulations to compute this, but exact methods are always nice.

Re: Post your approach (response to post by survival07) | Feedback: (+2/-0) | [+] [-] | Reply Thu, Nov 5, 2020 at 10:49 AM BDT

dimkadimon
4478 posts

I hope you enjoyed your first MM and I look forward to seeing you in future matches.

Re: Post your approach (response to post by sullyper) | Feedback: (+1/-0) | [+] [-] | Reply Thu, Nov 5, 2020 at 10:50 AM BDT

dimkadimon
4478 posts

I am sorry to hear about your 2 case failure. I'll see if I can get any memory/error information for you.

Re: Post your approach (response to post by dimkadimon) | Feedback: (+2/-0) | [+] [-] | Reply Thu, Nov 5, 2020 at 12:13 PM BDT

eulerscheZahl
2 posts

Here's my code for the expected number of goals with some comments added:

```
static double[,] goalsExpected = new double[11, 11];
static double[,][] distributionCurve = new double[11, 11][];
static void ComputeProbablity(int x)
{
    for (int attack = 1; attack <= 10; attack++)
    {
        for (int defense = 1; defense <= 10; defense++)
        {
            distributionCurve[attack, defense] = new double[4];
            // for each pair of attack, defense
            // count the number of goal my testing each pair of random values within the possible range
            double goals = 0;
            for (int a = 1; a <= attack; a++)
            {
                for (int d = 1; d <= defense; d++)
                {
                    if (a > d) goals++;
                }
            }
            // and divide by the total number of pairs to get the probability that one attack gives a goal
            double p = goals / (attack * defense);

            // what's that probability that N attacks cause a goal?
            for (int n = 0; n <= 3 * x; n++)
            {
                // n goals, 3*a-n misses. Binomial coeff as order doesn't matter
                double probN = Math.Pow(p, n) * Math.Pow(1 - p, 3 * x - n) * Binom(3 * x, n);

                // apply some rounding to match the testcase generator
                goalsExpected[attack, defense] += probN * Math.Floor((double)n / x + 0.5);
                distributionCurve[attack, defense][(int)((double)n / x + 0.5)] += probN;
            }
        }
    }
}
```

Re: Post your approach (response to post by dimkadimon) | Feedback: (+1/-0) | [+] [-] | Reply Thu, Nov 5, 2020 at 1:57 PM BDT

sullyper
264 posts

That's on me, I should have been more careful and take a risk to submit another solution 5min from the end, but yes it would be nice to get the feedback again as it used to be the case on the old platform. (Also might be time to consider increasing a bit the memory limit, but that's a different debate that not every body might agree with).

Re: Post your approach (response to post by iwashi31) | Feedback: (+4/-0) | [+] [-] | Reply Fri, Nov 6, 2020 at 5:57 PM BDT

ika
258 posts

Precomputations

For each attack / defence / number of goals find probability P[a][b][g] of happening that
For each attack / defence find expected number of goals = sum g*P[a][b][g]

Same two phases as others

1. guess attack/defence of each team
2. find most probable outcomes of all matches

Nothing interesting in the first phase - tried to minimize sum of the squared differences between expected and real scored/goals. Simplest hill climbing hear worked for me well.

In the second phase

1. Note that exact solution should compute **outcome of each match maximizing 2*guessed scores + 1*guessed results. This is hard to even imagine..**

2. I tried to compute most probable outcome of each match that is for each pair of teams find (X,Y) maximizing the conditional probability: **probability of result X:Y between teams i,j | given that scored / conceded / point arrays are a,b,c.**

3. What I really did is to calculate: **probability of result X:Y between teams i,j | given that scored / conceded / point of team i are a[i]/b[i]/c[i]**

Implementation

For N<=20 1GB memory allows to run DP algorithm:

A[i][j][scored][conceded][points] = probability of team i, played with all teams before j and ended up with scored / conceded / points

B[i][j][scored][conceded][points] = probability of team i, played with all teams after j and ended up with scored / conceded / points.

C[i][j][scored][conceded][points] = probability of team i, played with all teams except j and ended up with scored / conceded / points (easy when we have A and B.).

P[i][j][X][Y] probability of result X:Y (easy when we have C. Note that only certain values needed)

For N>20 I approximated C[i][j][s][c][p] with C1[i][j][s][p] * C2[i][j][c][p].
What I didn't manage to implement in these 3-4 days: compute the only meaningful values of C for larger N. Problem is memory limitation here.

Moving from C1[i][j][s][p] * C2[i][j][c][p] approximation to C[i][j][s][c][p] for N<=20: +1.45
Moving from 1-10 strengths to 1-512: +0.3

Re: Post your approach (response to post by dimkadimon) | Feedback: (+1/-0) | [+] [-] | Reply Sat, Nov 7, 2020 at 8:48 AM BDT

sullyper
264 posts

I got evaluated with 20 failures, so 1%
=(

Re: Post your approach (response to post by **sullyper**) | Feedback: (+1/-0) | [+] [-] | Reply Sat, Nov 7, 2020 at 11:48 AM BDT

dimkadimon
4478 posts

Yes very dangerous to submit a new idea in the last few minutes. I wouldn't do it, then again I am not red so don't listen to my advice :)

I can confirm that you have two failures (score=-1) in the provisional tests with seeds: 88405934287910 and 88405934287912. You have 20 failures in the system tests. Unfortunately I can't see any other information like memory usage or reason of failure.

Forums Development Forums Marathon Match 121 v1.0 Marathon Match 121 Post your approach [1 2] NEXT >
Previous Thread | Next Thread