In play tennis price modeling

Algorithm of prediction model

Our model can be described through a set of paths from the current state of the tennis to win or loss for the first player. We will use Betfair prices for calculation quite effective market probabilities for player 1 to win in a match.

$$P_{match} = 1/Odds_{betfair} \qquad (1)$$

Define:

p - probability of the player 1cwinning a point on serve.

q - probability of the player 1 winning a point on return.

P_g - probability of the player 1 winning a full game.

P_{cg} - - probability of the player 1 winning the current game.

We can define that P_{match} equals the sum of products of probabilities for each path from the current state of match to the win of the player 1. The probability of the player 1 to win the current game is

$$P_{cg} = \sum_{i=1}^{n} (p * q * p * q * (1-p)*(1-q)....)_i \qquad (2)$$

where n - quantity of paths to win the current game for the player 1

the chain of wins p,q,p,q and losses $(1-p)*(1-q)$ of the player 1 to finish the current game

Then we can define the probability to win whole match as

$$P_{match} = \sum_{j=1} (P_{cg} * P_g * (1- P_g)*....)_j \qquad (3)$$

This is a sum of probabilities products for all paths to win the match by the player 1

We can get P_{match} from Betfair and we should define the p,q and P_g for existed Pmatch .

First of all we will create a function for calculation P_{match} for any combination of p,q and P_g. We can use recursive approach for tracing the tennis match graph. We will use graphs like the graph at the *page 16 setbyseyanalysis.pdf*. And we can add tie brakes and all possible situation in a tennis match to this graph. We will use a computer function instead of O'Malley's analytical approach. It allows to have more flexible function. After that we will be able to create Cost function as a difference between P_{match} from (1) from Betfair and P_{match} from the function (3) .

$$\mathbf{F}\text{cost}(p,q,P_g, P_{match\ bf}) = 0 \qquad (4)$$

Thereby, we have the Cost function (4) with 3 unknown p,q,P_g . Our task is to find a minimum of this function. We can use different standard approaches for this like Gradient Descent ,LBFGS, etc.

If we will know p,q,P_g we will be able to calculate the P_{match_p} from the next point and calculate the predicted price.

$$Odds_{predict} = 1/ P_{match\_p} \qquad (5)$$

The main steps of the prediction algorithm:

1. We obtain value of Match odds for player 1 and calculate probability Player A to win the match.
   probwin = 1/Odds

2. Collect all possible paths to win for player 1 from the current moment.
   We use recursive function for this. For example players begin the point and player 1 can win or can can loss .We track both possibilities and collect product of probabilities. If any path goes to the loss of the player 1 we remove it. We will consider only paths to win.

   This is an example of tracing algorithm for one game in Python:

```
probsum = 0   # The collector for probabilities

#This is a class for nodes in the tennis match graph
class TreeNode:
    def  __init__(self, data,parent = None,left=None, right=None):
        self.data = data      #Contains score and server name
        self.left = left       #connect to the next point after the win
        self.right = right    #connect to the next point after the loss
        self.parent = parent




 #The function for calculation of probability to win for player A from the current moment
# prob_serv - probability of the player A winning a point on serve.
#prob_return- probability of the player B winning a point on return.

def prob_win(stnode,prob_serv,prob_return):
    global probsum
    point_A = stnode.data[0]      # player A score
    point_B = stnode.data[1]     # player B score
    server = stnode.data[2]       #Who is server

    if point_B > 100 or  point_A > 100 : return   #The limit for deuce. The probability of the win the
                                                  game will not change significantly after this limit
    if point_B >= 40 and point_B - point_A >= 20:   #The win of  Player B .We won't track this path
        return
    if point_A >= 40 and point_A - point_B >= 20:    # Player A win the game
        prob_multip = stnode.data[3]
        cnode = stnode
        while 1:        #Go to back and calculate probability of  the win
            pn = cnode.parent
```

```
        if pn == None: break
        prob_multip *= pn.data[3]
        cnode = pn
    probsum += prob_multip     #Collect total probability to win the game for Player A
    return

  #The first case is player A win the point
  if point_A < 30:
    dlt = 15       #The score for the first and second point 15,30
  else:
    dlt = 10    #The score for the third and other points 40,..
  if server == "A":          #Player A served
    win_node = TreeNode([point_A+dlt,point_B, "B",prob_serv],stnode) #Create the next node in the
                                                                      graph
  else:
    win_node = TreeNode([point_A+dlt,point_B, "A",prob_return],stnode)
  stnode.left = win_node   # Connect the current node to the next node
  prob_win(win_node,prob_serv,prob_return)     #The recursive call. Go to the next node in the graph
after the win of Player A

  #The second  case is Player B win the point
  if point_B < 30:
    dlt = 15       #The score for the first and second point 15,30
  else:
    dlt = 10     #The score for the third and other points 40,..
  if server == "A":
    lose_node = TreeNode([point_A,point_B+dlt, "B",1-prob_serv],stnode)
  else:
    lose_node = TreeNode([point_A,point_B+dlt, "A",1-prob_return],stnode)
  stnode.right = lose_node
  prob_win(lose_node,prob_serv,prob_return)      #The recursive call. Go to the next node in the graph
after the loss of Player A
```

We will expand this function for the whole match.

3. Create a cost function:

This is an example of Cost function for one game

```
def cost_func(params,probwin ):
  global probsum
  probsum = 0
  prob_serv = params[0]
  prob_return = params[1]
  stnode = TreeNode([0,0, "A",1])
  prob_win(stnode,prob_serv,prob_return)
  return abs(probsum-probwin
```

4. We will use optimization algorithms to find prob_serv and prob_return when cost_func ~0 where:

   probwin = $1/ODDS_{betfair}$

5. If we know prob_serv and prob_return we can calculate probabilities Player A to win the match.

   After the Player's A win in this point
   stnode = TreeNode([15,0, "B",1])
   P_1 = prob_win(stnode,prob_serv,prob_return)

   After the the Player's A loss in this point
   stnode = TreeNode([0,15, "B",1])
   P_2 = prob_win(stnode,prob_serv,prob_return)

6. Calculate predictions for the next prices:

   $ODDS_1 = 1/ P_1$

   $ODDS_2 = 1/ P_2$