Kenneth Yam ID: 61791166
Rodrigo Erquiaga ID: 43484513
David Gathright ID: 15478532

CS 179 Final Report: Using TrueSkill to predict DotA tournament winnings

TrueSkill is a ranking system created by Microsoft for XBox matchmaking. This ranking system is a replacement for the Elo Ranking System that was made for one-on-one games. Games such as DotA 2 have multiple people of different skill ratings team up against another group of players with different skill ratings, and the Elo ranking system was inaccurate. What the TrueSkill ranking system does is copy Elo Ranking System, but adds weights to individual players to make the system work for games that have more than two people going head-to-head. Furthermore, in the case of ties, what TrueSkill does is determine how much the ratings will increase or decrease by adding the possibility of the game ending in a tie: the lower the chance of a tie, the lower the ratings change after a match.

What TrueSkill also helps to combat is the eventual inflation of skill ratings that the Elo Ranking System has trouble with. By adding another factor into account, the player's rating is limited and players who have played for much longer will not have their rating be too high relative to other players.

Our group chose to apply TrueSkill to a set of DotA 2 tournaments from 2017-2018 to see how well its algorithms apply to professional scene games. Our primary focus was to see how well TrueSkill could predict or justify the results of the final tournament for the season, The International 2018, by training on all of the previous Major and Minor tournaments. It is worth noting that The International 2018 featured 3 teams who did not play in the circuit, so their training is lacking compared to the other teams. As a method of simplification, since we are focusing exclusively on matches from this competitive circuit, every team always consists of the same 5 members. This allows us to treat all the players the same instead of differentiating between each player in the team, since they will all be initialized from the same rating and trained on the same results.

Our primary resource in this project was the TrueSkill library, downloaded by executing the command "pip install trueskill" through PyPI. The page for the code can be found at its website (https://pypi.org/project/trueskill/) with links to its documentation (https://trueskill.org/) and GitHub. We also referred to a paper (https://www.microsoft.com/en-us/research/project/trueskill-ranking-system/) about TrueSkill published by Microsoft to describe how it works and why it was built.

Another important resource we used was the website Liquipedia, which contains extensive information about the DotA 2 professional scene. Specifically, we utilized its data on every tournament in the 2017-2018 professional circuit (https://liquipedia.net/dota2/Dota_Pro_Circuit/2017-18) to train our TrueSkill player and team ratings by hard-coding the results of each match. We also used its data on The International 2018 tournament (https://liquipedia.net/dota2/The_International/2018) when we were testing the accuracy of TrueSkill's ratings and win prediction.

The main parts of the TrueSkill library we used were the Rating objects and the rate function. Each Rating object represents a player's skill, with a mean and standard deviation for how well they expect the player to perform compared to others. These can also be used to compare groups of players and predict who will win. We initialized all players to the same skill level for this experiment. The rate function updates some amount of Ratings based on a given set of ranks for a match. For our purposes, every match was between two teams of 5, so it updated the rating of each member of each team according to which team won; ranks were either 0 for winning or 1 for losing here. The specific code we implemented is described in more detail below.

The code we wrote was largely focused on hard-coding every match's results, since we had no easy way to extract the data elsewhere. We first established the names of every team who participated in the 2017-2018 professional circuit as well as The International 2018 in a list, then enumerated over it to create a dict linking these names to integers. We then iterated over all of these established teams, creating a set of TrueSkill Ranking objects for each; these represent the players of each time, while the list containing them represents the whole team itself. All of these steps are shown in the code below, with the total team names being cut off for readability.

```
In [4]: import trueskill as ts

        # Team Indexes
        team_names = [ 'Virtus.Pro',
            'Team Liquid',
            'PSG.LGD',
            'Team Secret',
            'Mineski',
```

```
team_index = {}

for index, name in enumerate(team_names):
    team_index[name] = index
```

```
# Creating teams
teams = [[ts.Rating() for _ in range(5)] for _ in range(len(team_names))]
```

The next step of our code was to hard-code all of the matches that occurred in the various tournaments. Our method was to create a list of lists, where each interior list represented a series of matches between two teams, then iterate through this exterior list and adjust our TrueSkill Rating objects accordingly. Specifically, the structure of each interior list was: [name of team who won the series, name of team who lost the series, wins the series-winning team had, wins the series-winning team lost]. These integers representing wins and losses are necessary since many of the series were best-of-three rounds, with some going 2-0 or 2-1; additionally, a few best-of-five rounds occurred at the end of tournaments. A sample of several of these matches is shown in code below (overall, there were easily over a hundred matches), along with our loop for updating TrueSkill Rating objects.

```
In [3]: matches = []

        # StarLadder i-League Invitational Season 3
        matches.append( [ 'Complexity Gaming', 'Mineski', 2, 1 ] )
        matches.append( [ 'Team Liquid', 'Team Secret', 2, 0 ] )
        matches.append( [ 'Team Liquid', 'Mineski', 3, 1 ] )

        # PGL Open Bucharest
        matches.append( [ 'Mineski', 'Immortals', 2, 1 ] )
```

```
for winner, loser, win_count, lose_count in matches:
    for i in range( win_count ):
        teams[team_index[winner]], teams[team_index[loser]] = \
        trueskill.rate([teams[team_index[winner]], teams[team_index[loser]]], ranks=[0, 1])
        teams[team_index[winner]] = list(teams[team_index[winner]])
        teams[team_index[loser]] = list(teams[team_index[loser]])
    for i in range ( lose_count ):
        teams[team_index[winner]], teams[team_index[loser]] = \
        trueskill.rate([teams[team_index[winner]], teams[team_index[loser]]], ranks=[1, 0])
        teams[team_index[winner]] = list(teams[team_index[winner]])
        teams[team_index[loser]] = list(teams[team_index[loser]])
```

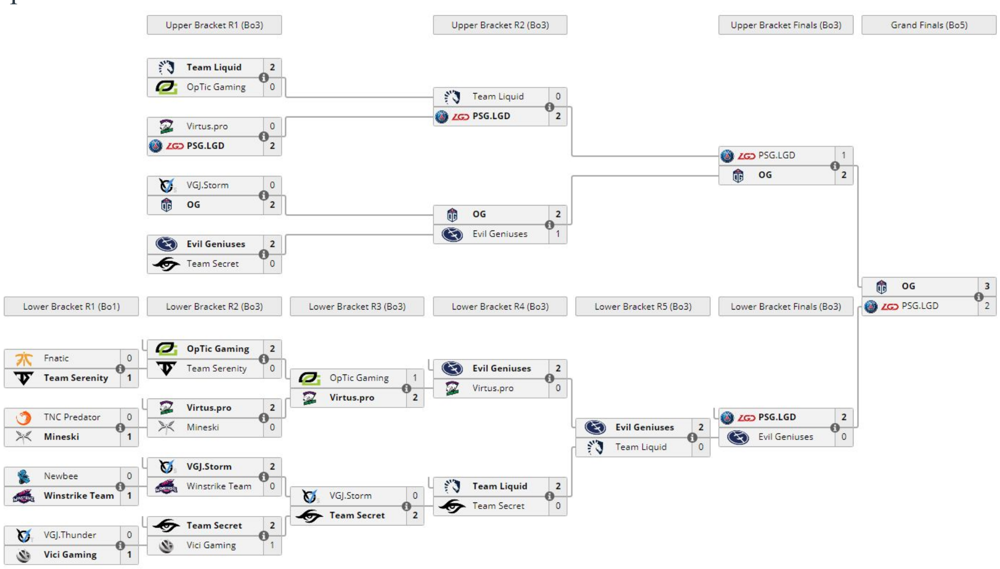
For the above code, it is noteworthy that we couldn't just take the results from trueskill.rate as they were; since they returned tuples and we formatted everything as lists, we had to perform several additional updates after every TrueSkill rate function adjusted our Ratings.

We used the following functions to take two teams and predict who will win. Since in our implementation every player on a team has the same skill, we were able to simplify the prediction by only considering the first player of each team when predicting who will win. The function win_chance establishes who is more likely to win, and update_winner uses this prediction to report who will win as well as update the TrueSkill Ratings based on this win.

```python
def win_chance(p1, p2): #chance p1 beats p2
    deltaMu = p1.mu - p2.mu
    rsss = sqrt(p1.sigma**2 + p2.sigma**2)
    return trueskill.global_env().cdf(deltaMu/rsss)
```

```python
def update_winner(nameA, nameB):
    teamA = teams[team_index[nameA]]
    teamB = teams[team_index[nameB]]
    if win_chance(teamA[0], teamB[0]):
        winner = nameA
        loser = nameB
        teams[team_index[winner]], teams[team_index[loser]] = \
        trueskill.rate([teams[team_index[winner]], teams[team_index[loser]]], ranks=[0, 1])
        teams[team_index[winner]] = list(teams[team_index[winner]])
        teams[team_index[loser]] = list(teams[team_index[loser]])
        return nameA
    else:
        winner = nameB
        loser = nameA
        teams[team_index[winner]], teams[team_index[loser]] = \
        trueskill.rate([teams[team_index[winner]], teams[team_index[loser]]], ranks=[0, 1])
        teams[team_index[winner]] = list(teams[team_index[winner]])
        teams[team_index[loser]] = list(teams[team_index[loser]])
        return nameB
```

We chose to test the accuracy of TrueSkill's relative skill levels by having it predict the outcome of The International 2018 tournament after training on the preceding competitive tournaments, the 2017-2018 professional circuit. The brackets and results of The International 2018 can be seen below:



The results of these predictions, in code and output, can be seen below. The initial brackets were created, then a simulated run of the entire tournament by TrueSkill's expectations was run.

```
In [6]: ## Tournament: The International

predict_matches_upper = [] # [winner, loser] # win_count is number of wins for winner, lose_count is number of wins for loser
predict_matches_lower = []

predict_matches_upper.append( [ 'Team Liquid', 'OpTic Gaming'] )
predict_matches_upper.append( [ 'Virtus.Pro', 'PSG.LGD'] )
predict_matches_upper.append( [ 'VGJ.Storm', 'OG'] )
predict_matches_upper.append( [ 'Evil Geniuses', 'Team Secret' ] )
predict_matches_lower.append( [ 'Team Serenity', 'Fnatic'] )
predict_matches_lower.append( [ 'TNC Predator', 'Mineski' ] )
predict_matches_lower.append( [ 'Newbee', 'Winstrike Team' ] )
predict_matches_lower.append( [ 'VGJ.Thunder', 'Vici Gaming' ] )

new_predict_matches_upper = []
new_predict_matches_lower = []

for nameA, nameB in predict_matches_lower:
    winner = update_winner(nameA, nameB)
    print "Winner: ", winner
    new_predict_matches_lower.append([winner])

lower_index = 0
upper_list = []
for nameA, nameB in predict_matches_upper:
    winner = update_winner(nameA, nameB)
    print "Winner: ", winner
    upper_list.append(winner)
    if(winner == nameA):
        new_predict_matches_lower[lower_index].append(nameB)
    else:
        new_predict_matches_lower[lower_index].append(nameA)
```

```
Winner:  Team Serenity
Winner:  TNC Predator
Winner:  Newbee
Winner:  VGJ.Thunder
Winner:  Team Liquid
Winner:  Virtus.Pro
Winner:  VGJ.Storm
Winner:  Evil Geniuses
[['Team Liquid', 'Virtus.Pro'], ['VGJ.Storm', 'Evil Geniuses']]
[['Team Serenity', 'OpTic Gaming'], ['TNC Predator', 'PSG.LGD'], ['Newbee', 'OG'], ['VGJ.Thunder', 'Team Secret']]
----------
Winner:  Team Serenity
Winner:  TNC Predator
Winner:  Newbee
Winner:  VGJ.Thunder
Winner:  Team Liquid
Winner:  VGJ.Storm
[['Team Liquid', 'VGJ.Storm']]
[['Team Serenity', 'TNC Predator'], ['Newbee', 'VGJ.Thunder']]
----------
Winner:  Team Serenity
Winner:  Newbee
Winner:  Team Liquid
[]
[['Team Serenity', 'Virtus.Pro'], ['Newbee', 'Evil Geniuses']]
----------
Winner:  Team Serenity
Winner:  Newbee
[]
[['Team Serenity', 'Newbee']]
----------
Final Winner:  Team Liquid
```

TrueSkill's final result turned out to be inaccurate but did correct several lower brackets correctly. Based on its training data of previous tournaments, most of the winning teams seemed to also win in our calculated tournament. Below is the final TrueSkill scores of out predictions:

```
In [10]: for index, team in enumerate(teams):
             print "Team {} mean TrueSkill:".format(index), team[0].mu
```

```
Team 0 mean TrueSkill: 29.7266201909
Team 1 mean TrueSkill: 31.4992459206
Team 2 mean TrueSkill: 27.997601819
Team 3 mean TrueSkill: 26.9520503255
Team 4 mean TrueSkill: 26.1325673454
Team 5 mean TrueSkill: 26.5876109028
Team 6 mean TrueSkill: 28.2936128197
Team 7 mean TrueSkill: 26.7731492121
Team 8 mean TrueSkill: 22.7948229309
Team 9 mean TrueSkill: 23.4336432981
Team 10 mean TrueSkill: 30.6708711798
Team 11 mean TrueSkill: 25.0
Team 12 mean TrueSkill: 24.3348317173
Team 13 mean TrueSkill: 27.8392429988
Team 14 mean TrueSkill: 27.8893870802
Team 15 mean TrueSkill: 25.6256049918
Team 16 mean TrueSkill: 26.4380613811
Team 17 mean TrueSkill: 24.6679098568
Team 18 mean TrueSkill: 25.3264564076
Team 19 mean TrueSkill: 29.1483039643
Team 20 mean TrueSkill: 24.9405307874
Team 21 mean TrueSkill: 25.7541467678
Team 22 mean TrueSkill: 25.0
Team 23 mean TrueSkill: 21.5634275066
Team 24 mean TrueSkill: 20.7690069592
Team 25 mean TrueSkill: 20.0488784391
Team 26 mean TrueSkill: 25.7702310817
Team 27 mean TrueSkill: 25.0
Team 28 mean TrueSkill: 23.997556503
Team 29 mean TrueSkill: 21.8723506255
Team 30 mean TrueSkill: 21.862504987
Team 31 mean TrueSkill: 23.0341241571
Team 32 mean TrueSkill: 23.0308035056
Team 33 mean TrueSkill: 20.0630819037
Team 34 mean TrueSkill: 20.6374128265
Team 35 mean TrueSkill: 29.9932080102
Team 36 mean TrueSkill: 24.1638344358
Team 37 mean TrueSkill: 26.5888590113
```

As a final note, it is worth considering how our methods could have been improved. First of all, we considered the skill of each player in each team to be the same, and the only differences in skill to be between different teams. This is an obviously inaccurate assumption, as players are expected to have different skill levels. Another problem was that our evaluation of each player's skill started with the 2017-2018 professional circuit. While considering each individual player's entire career (or at least considering a more extended view of their career) would improve our accuracy, it would also take an extreme amount of work beyond our scope for this project. That being said, it is possible to gain more information on each individual professional player and improve accuracy by using the OpenDota API. This is a very in-depth API giving access to numerous aspects of every recorded DotA 2 match as well as the play history of its players in one large database. A sample piece of code querying OpenDota for information on professional players can be seen below; unfortunately, we did not have much time or experience to follow up on this potential lead.

```
In [9]: url = "https://api.opendota.com/api/proPlayers"
        header = {'User-Agent': 'Mozilla/5.0'}
        request = urllib2.Request(url,headers=header)
        data = json.load(urllib2.urlopen(request))
```