# Unsupervised learning to detect teams from individual player trajectories

Comp – 652: Machine learning

Professor: Donina Precup
Student: Gayane Petrosyan

12/15/2011
McGill University

# Table of Contents

# 1. Introduction

There are a lot of different cheating types for multiplayer online games. Depending on the game, different activities can be considered as a cheating. Secret aliasing is more specific to massively multiplayer online role-playing game (MMORPG: for example *World of Warcraft*). If two or more players secretly engage, co-operative during the play it is considered cheating in many games, in particular when players engage in secondary communication. The idea is to find out those hidden aliases by observing user actions. As the first step of this challenging idea this project will try to distinguish actual teams (not secret) in team based game by looking into the actions the players take.

# 2. Orbius game and Challenge

Orbius is Mammoth's first subgame. Mammoth is a large scale multiplayer game research framework developed at McGill University. Its main purpose is to provide an environment for academic experimentation. Orbius is 2d multiplayer game where players are divided into teams. Each player moves in virtual environment of the game seeking objects which will give them points to win. After experimental run of the game, 3 full game logs were available, where 28 players were divided into 7 teams[1].

So, the goal of this project is to process player's movement information and find features which will distinguish one team of player from the other.

[1] Special thanks to Marc Lanctot; Michael A. Hawker and Prof. Joerg Kienzle for providing me with Orbius data and introducing me to the game

# 3. Feature selection and preprocessing

## 3.1 Players trajectories

The features which need to be selected should be able to emphasize strategies of the players. If we look closer to the trajectories of the players, we will see that players walk around the game map, they stop for a long period of time in some particular place or they walk around the same region multiple time. The selected features should be able to preserve the decisions which players made, because that is exactly what carries information about the strategy used by players and teams.

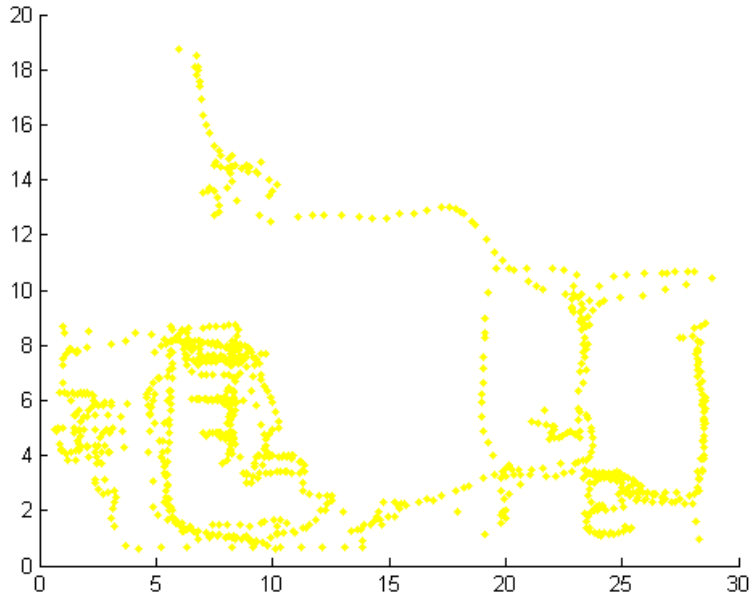



Figure 1: Trajectory of one player

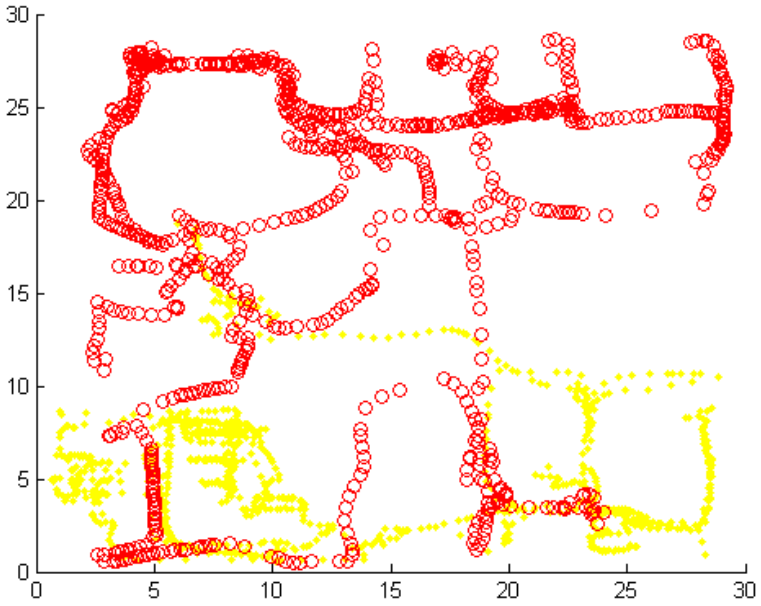



Figure 2: Trajectories of two players from different teams

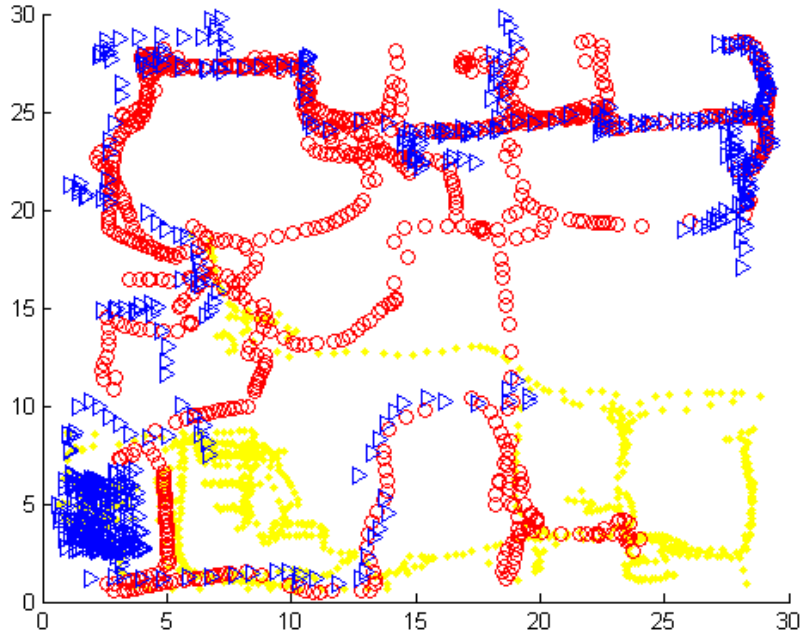



Figure 3: Trajectories of three players from different teams

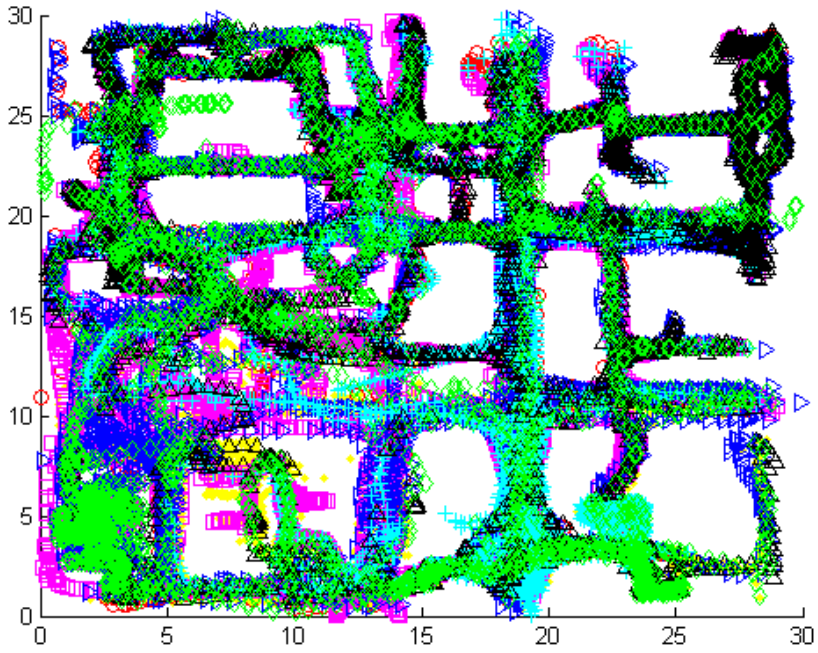



Figure 4: Trajectories of all players

## 3.2 Filtering movement data by intervals of time

In Orbius game different players have done different number of steps, which maybe can be visible from the figures above too. For that purpose and also for compressing data for machine learning algorithm the movement data of the players were divided into intervals of the time. For each interval the position of the player has been fixed. For this particular setup the interval was taken was 5 seconds. As can be seen in Figure5 and Figure6 this step preserves main trends of the trajectory, but reduces the size of the dataset and, what is not less important, now all players have the same number of positions.

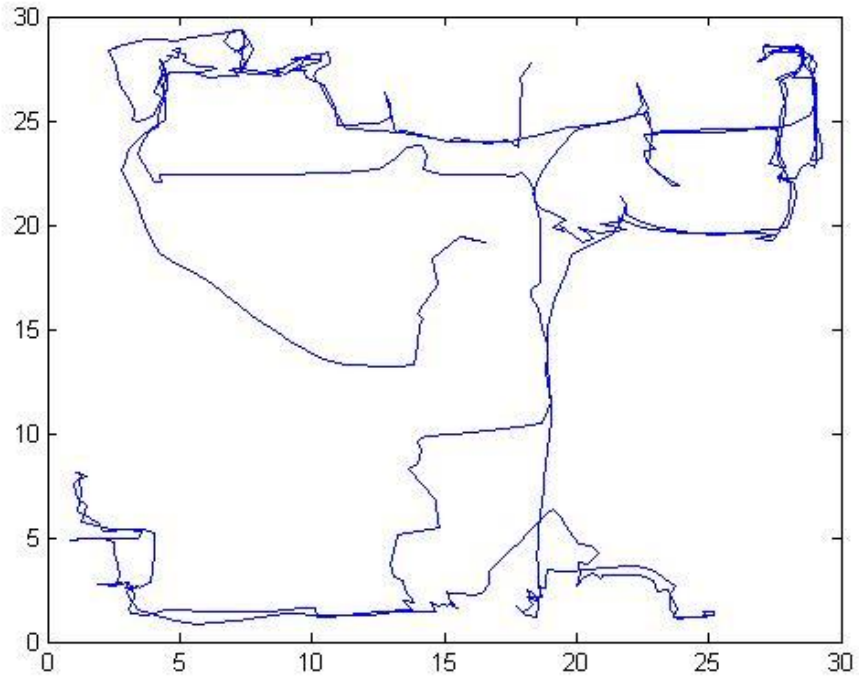



Figure 5: Trajectory before preprocessing

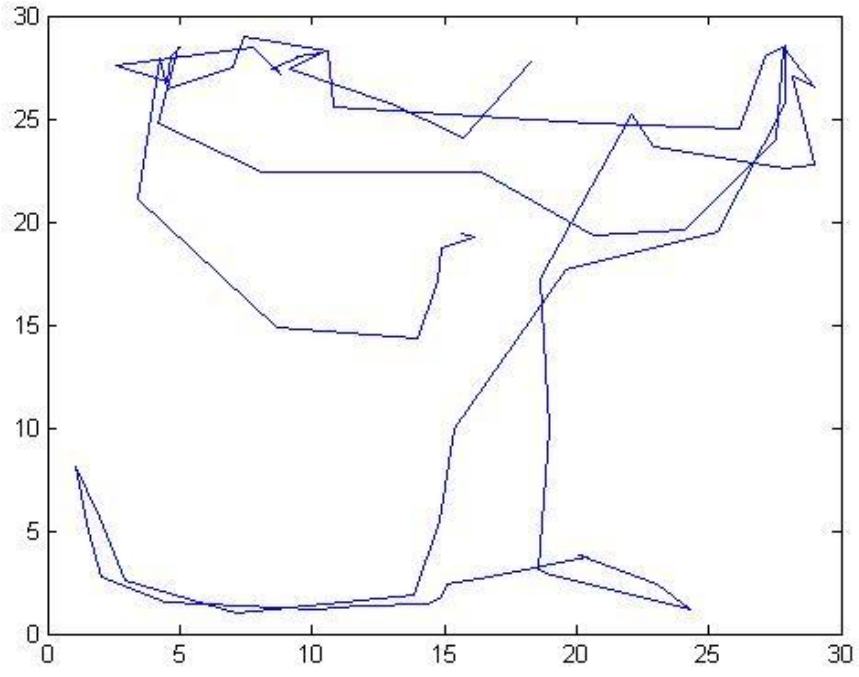



Figure 6: Trajectory after preprocessing

## 3.3 Calculating a distance from origin

As different players start the game from different position, it is hard to judge their actions based on the positions. That is why as a measure of player's movement the distance of the player from his origin was taken. This decision didn't spoil the data as all the trends of the players which we need, still will be present. That is, player stays in the same location for a long time or player explores some region of the game world. Even more, it will make comparison between players easier. For example, in the start of the game all players are near their start positions so all of them will have equal values and not random positions.

## 3.4 Features distribution

After all the preprocessing steps we have two features: point in a time and the distance of the player from its origin. Before deciding what machine learning algorithm can be used let's look the distribution the features have. The distribution of distance looks like mixture of Gaussians and the distribution of time point are uniform.

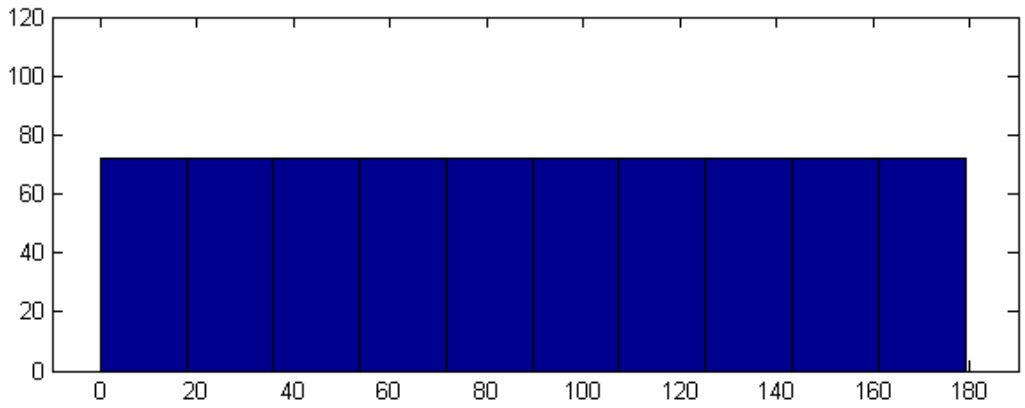



Figure 7: Distribution of time feature

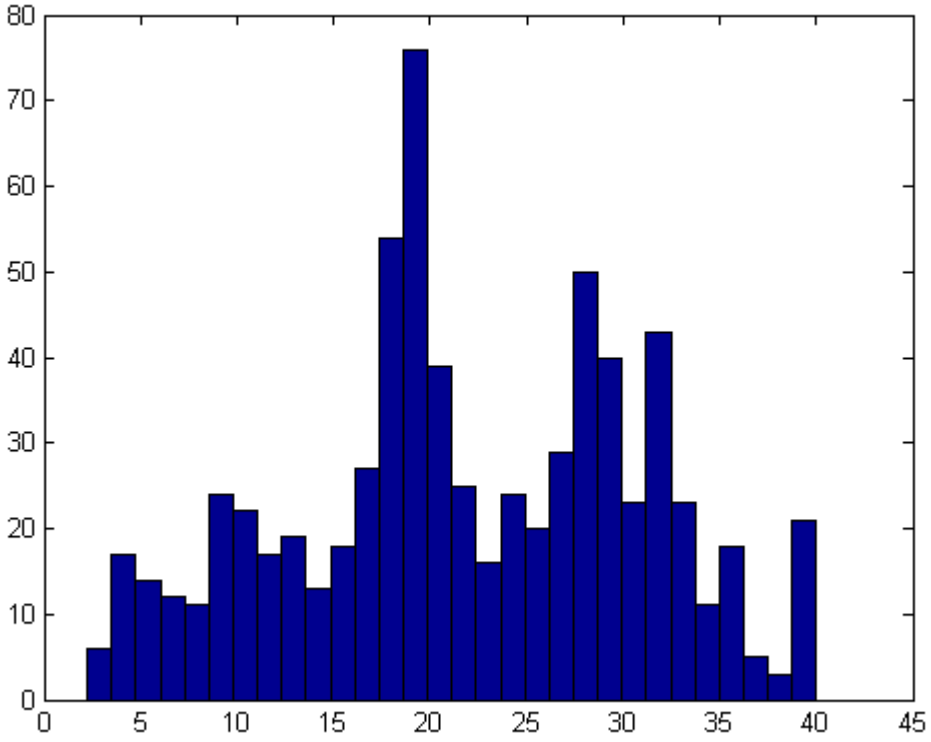
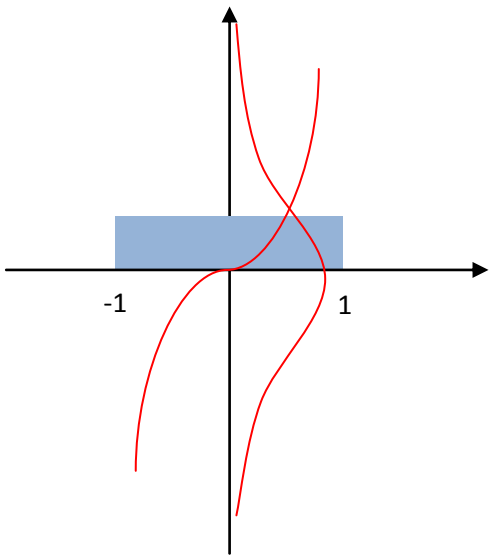



Figure 8: Distribution of distance feature

To be able to successfully use mixture of Gaussians it would be better to turn uniform distribution into Gaussian. For that purpose I used Inverse of the error function which has sigmoidal form. If we apply this function to data which is uniformly distributed from -1 to 1, then we will get Gaussian distribution.



# 4. General approach

The general approach is to make models of teams based on 2 games and try to predict teams for all players from 3 games. If we have models of the team then we can calculate the probability of the player to be in each team. The idea is the same as in anomaly detection, where it is assumed that anomalies are not in highly concentrated areas, thus they will have low probability in Gaussian distribution. In this case the model is the same, but opposite cases are the one which are interesting.

## 4.1 Mixture of Gaussian

For building the models of the teams', first of all, data for each team from training dataset was combined to one. Afterwards, to build model of team for each of them mixture of Gaussian have been used. Using multiple iterations of EM the optimal parameters of 3 Gaussians for each team have been found. That means that for each team there are three clusters formed by 3 Gaussians. Followings are some examples of this step.

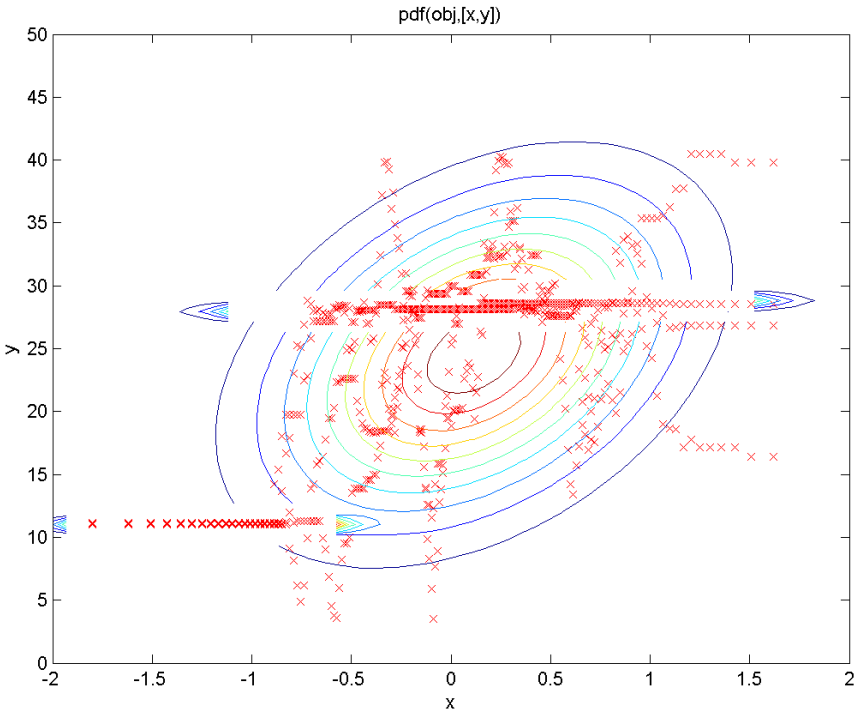



Figure 9: 3 Gaussians for one particular team 1

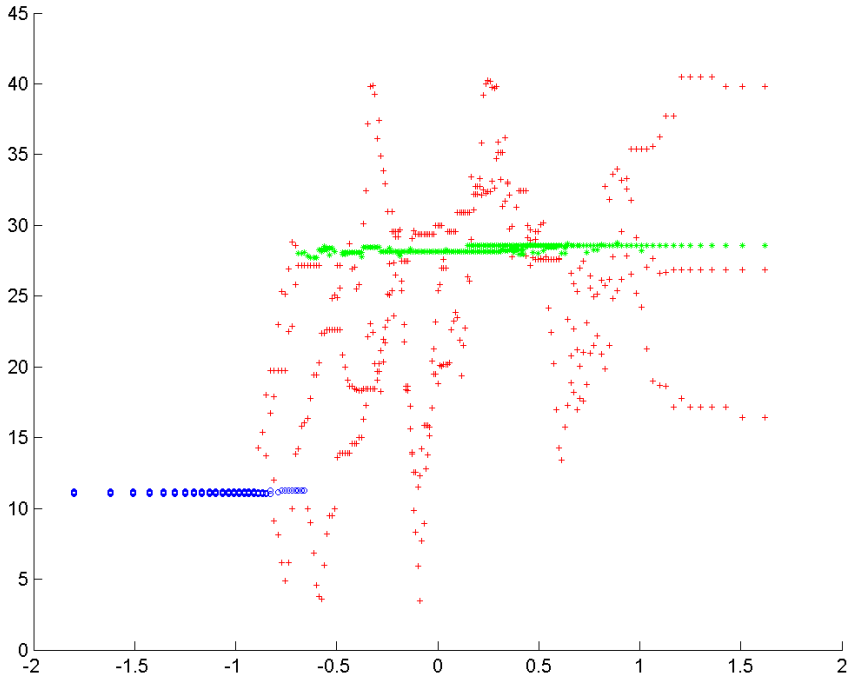



Figure 10: Clusters formed by mixture of Gaussians for team1

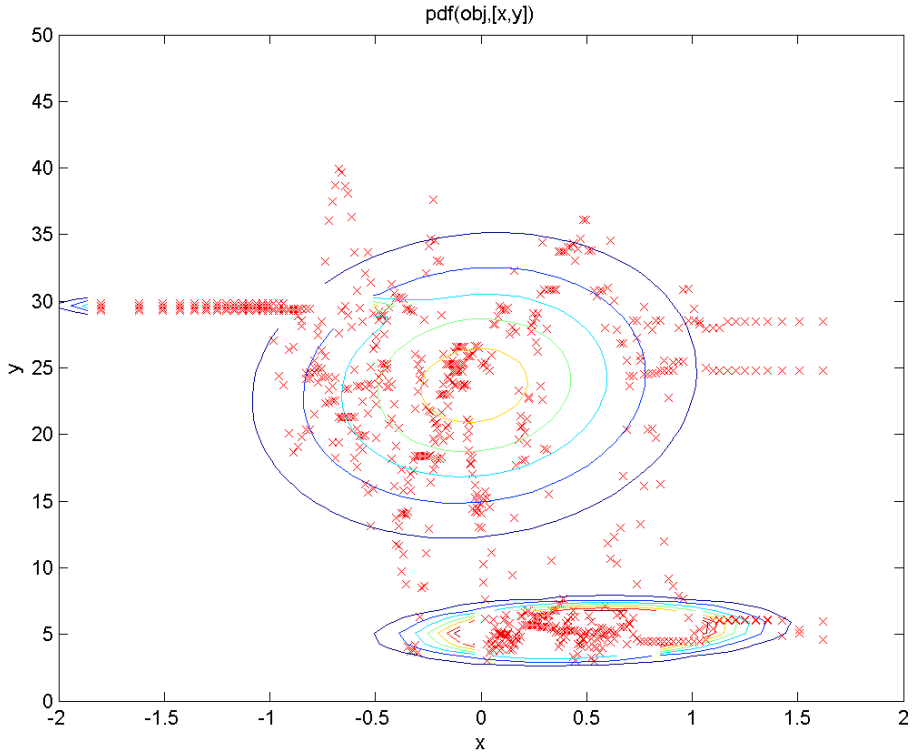



Figure 11: 3 Gaussians for one particular team 2

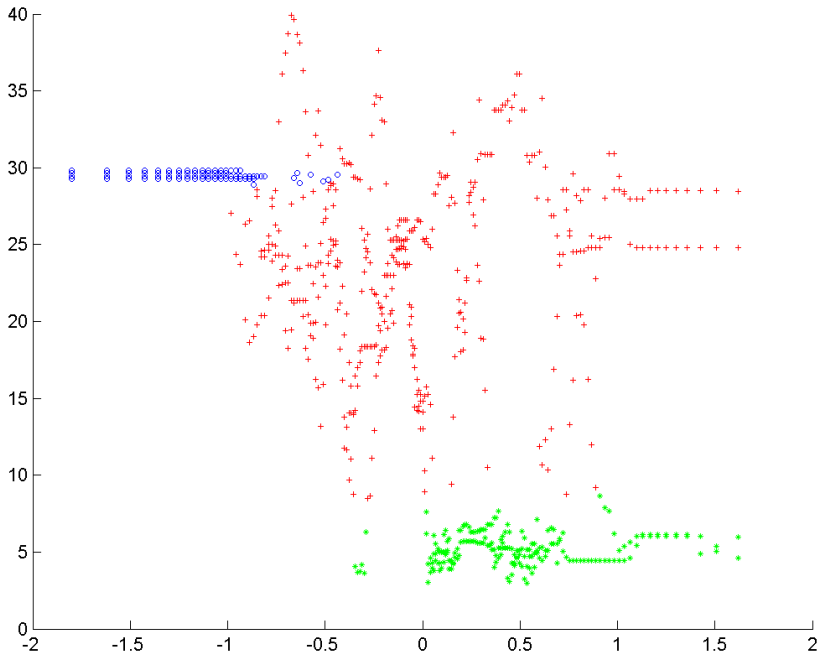



Figure 12: Clusters formed by mixture of Gaussians for team2

## 4.2 Grouping

After different models of the team are available, it is time to try to group the players. As a first step for that we need to calculate for each player the log likelihood of him being in particular team. The log likelihood of the player to be in particular team is expressed by following formula.

$$\sum_{x_i \in playerX} \max_{k,c} \left( \log P(x_i / y_i = k;\ \mu_{k,c}, \Sigma_{k,c} \right) + \log P(y_i = k, \phi_{k,c})$$

Here c is the one of the 3 clusters of the team; k is the team and $x_i$ are all the data of the player.

Eventually this will form a matrix log likelihoods of each player for each team.

Grouping the players into team has two sides. Each team should be formed by its most probable players and each player should be in its most probable team. For this, a repetitive algorithm is used, which contains two steps. First, it assigns to each team it's most probable players, and then from the one assigned during first step each player picks the one which is post probable for him and removes himself from other groups. In

the log likelihood matrix the value for this player for removed teams is replaced with small number in a way that they will not be selected for this player as a most probable during the first step.

The following picture tries to demonstrate the above described algorithm.

## 5. Results

The team models were used for both training data and testing data. As an error rate the sum of the absent members percentage for each team, have been used. Note, that it doesn't matter in what particular teams the players will be classified unless players of the same team still remain together.

Training vs. Testing

| | | | |
|---|---|---|---|
| 2 | 4 | 1 | 3 |
| 5 | 7 | 8 | 6 |
| 9 | 12 | 10 | 25 |
| 16 | 26 | 11 | 15 |
| 18 | 19 | 20 | 14 |
| 24 | 21 | 23 | 22 |
| 13 | 17 | 28 | 27 |

Error percentage: 0.214286

| | | | |
|---|---|---|---|
| 2 | 1 | 3 | 10 |
| 14 | 8 | 12 | 5 |
| 11 | 6 | 9 | 16 |
| 4 | 13 | 26 | 21 |
| 18 | 17 | 15 | 28 |
| 19 | 7 | 22 | 24 |
| 27 | 20 | 25 | 23 |

Error percentage: 0.535714

As can be seen the results are not that impressive. Testing error is around 0.5. So, more testing is necessary to find out whether the value of the algorithm.

# 6. Future work directions

First, to prove/disprove the algorithm more models of teams are necessary. In this case only 14 models are used which seems not enough for predictions.

Secondly, some other data besides the movement of the players can be used. For example, in Orbius game players can disturb each other. This is direct indictor that most probably they are not in the same team. This will surely strengthen the algorithm.

Third improvement which I can see is the elimination of similar teams. If the team models have similar parameters then while grouping the players it is the matter of chance where the players will end up. So, one part of the team players can be classified to one team and the other part to the second one. Thus, thought the team models were almost the same, it would seem like players are in different teams. That is why it is very important to eliminate similar team models.

## 7. References


[1] Marc Lanctot, Nicolas Ng Man Sun, Clark Verbrugge (2006)
PATH-FINDING FOR LARGE SCALE MULTIPLAYER COMPUTER GAMES

[2] McGill University (2005). Mammoth: The massively multi-player prototype. http://mammoth.cs.mcgill.ca

[3] Bishop C.M. (2009)
Pattern Recognition and Machine Learning

[4] [online] Andrew Ng, Machine Learning video lectures, Stanford Center for Professional Development http://academicearth.org/courses/machine-learning

[5] [online] Joaquin Quiñonero Candela, PASCAL Bootcamp in Machine Learning(2007), video lecture on The EM algorithm and Mixtures of Gaussians.
http://videolectures.net/bootcamp07_quinonero_emal/

[6] Zoubin Ghahramani Gatsby Computational Neuroscience Unit
University College London, UK . Unsupervised Learning(2004)

[7] Wikipedia,Cheating in online games
http://en.wikipedia.org/wiki/Cheating_in_online_games