SoccerCPD TimeSeries Project

Mohamed BENYAHIA - Omar EL MANSOURI

December 2024

1 Introduction and contributions

In fluid team sports like soccer, analyzing team formations helps understand tactics, but challenges arise from dynamic role changes, temporary switches, and abnormal situations. Prior studies either assumed fixed formations for long durations, detected formations too frequently, or struggled with reliable change-point detection.To address these issues, SoccerCPD performs **Role Assignment**: Assigning player roles frame-by-frame. Then, two-Step **Change-Point Detection**:

- Detecting formation changes using role-adjacency matrices
- Detecting role changes using role permutations

Mohamed did 55% and Omar did 45% of the work. In our work, the authors were unable to share the entire dataset due to security issues, so we only had access to the data of one sample match. We tried several preprocessing steps on it to increase the model's performance, and added a new visualization method (see Fig2-6). We've also tried a naive rupture detection method in order to understand the added value of the paper (using naively the mean over frames and applying ruptures on it). Moreover , we have tried an other form of CPD that was not treated in the paper (by trying kernel CPD), and played with the hyper-parameters of the method to test its influence. Finally, we re-utilized the core functions from the provided GitHub repository, as the original implementation lacked a well-structured, readable, and easily generalizable design.

2 Method

Consider a match/session divided into T time frames. For each time frame $t \in \{1, \ldots, T\}$, the 2D coordinates of player $i \in \{1, \ldots, N\}$ are $\mathbf{x}_{t,i} = \begin{pmatrix} x_{t,i} \\ y_{t,i} \end{pmatrix}$. The centroid at time t is $\overline{\mathbf{x}}_t = \frac{1}{N} \sum_{i=1}^{N} \mathbf{x}_{t,i}$, and the centered coordinates are $\mathbf{x}_{t,i}^{\text{norm}} = \mathbf{x}_{t,i} - \overline{\mathbf{x}}_t$, centering player positions around the origin to focus on relative positioning.

.

2.1 Role Assignment via Expectation-Maximization

Each player role $r \in \{1, 2, \ldots, N\}$ is modeled with a 2D Gaussian distribution:

$$p\left(x^{\text{norm}} \mid r\right) = \mathcal{N}\left(x^{\text{norm}}; \mu_r, \Sigma_r\right),$$

where $\mu_r \in R^2$ is the mean position and $\Sigma_r \in R^{2\times 2}$ is the covariance matrix for role r.

An iterative Expectation-Maximization (EM) algorithm assigns roles and estimates Gaussian parameters:

Initialization : Assign each player i to a distinct role $r = i$ across all frames.

E-Step: Role Reassignment : For each frame t, construct a cost matrix $C_t^{(k)}$ with entries

$$C_t^{(k)}(i, r) = -\log \mathcal{N}\left(x_{t,i}^{\text{norm}}; \mu_r^{(k)}, \Sigma_r^{(k)}\right).$$

Apply the Hungarian algorithm to find the optimal permutation σ_t^* that minimizes the total cost:

$$\min_{\sigma_t \in S_N} \sum_{i=1}^{N} C_t^{(k)}(i, \sigma_t(i)).$$

Update role assignments:

$$r_{t,i}^{(k+1)} = \sigma_t^*(i).$$

Optionally,we compute a switch rate to assess assignment reliability.

M-Step: Parameter Re-estimation : Update Gaussian parameters based on new assignments:

$$\mu_r^{(k+1)} = \frac{1}{|I_r^{(k+1)}|} \sum_{(t,i)\in I_r^{(k+1)}} x_{t,i}^{\text{norm}},$$

$$\Sigma_r^{(k+1)} = \frac{1}{|I_r^{(k+1)}|-1} \sum_{(t,i)\in I_r^{(k+1)}} \left(x_{t,i}^{\text{norm}} - \mu_r^{(k+1)}\right)\left(x_{t,i}^{\text{norm}} - \mu_r^{(k+1)}\right)^{\top}.$$

Optionally,we filter out assignments based on switch rates defined as:

$$\text{Switch Rate}(t) = \frac{\sum_{i=1}^{N} 1\{\beta_t(p_i) \neq \text{Most Frequent Role}(p_i)\}}{N},$$

where $\beta_t(p_i)$ is the role assigned to player p_i at time t, and Most Frequent Role(p_i).

Role Alignment Across Sessions : Apply the Hungarian algorithm to determine the optimal role permutation π that aligns roles in session s with the base group $\star$ (longuest period in the match without interruption) .

2.2 Delaunay Triangulation and Adjacency Matrices

For each valid frame t,we compute the Delaunay triangulation D_t of the role positions $\{r_k(t)\}_{k=1}^{N}$. Extract the edge set E_t and form the adjacency matrix

$$A(t) \in \{0,1\}^{N\times N}, \quad a_{k\ell}(t) = \begin{cases} 1 & \text{if } (k,\ell) \in E_t, \\ 0 & \text{otherwise.} \end{cases}$$

This results in a time series of adjacency matrices $\{A(t)\}_{t\in T}$, representing the topological neighbors of roles.

2.3 Change Point Detection (CPD)

Distance Metric : Define the Manhattan distance between adjacency matrices:

$$d_M(A(t), A(t')) = \sum_{k=1}^{N}\sum_{\ell=1}^{N} |a_{k\ell}(t) - a_{k\ell}(t')|.$$

Similarity Graph Construction : Construct a similarity graph where each node corresponds to a time frame and edge weights are given by d_M. Build a Minimum Spanning Tree (MST) to connect all frames with minimal total distance.

Scan Statistic : Define a scan statistic $R(s)$ for each potential split s as the number of MST edges crossing the split. Identify the change-point $\tau = \arg\max_s R(s)$. If $R(\tau)$ exceeds a threshold :(p-value less than 0.01, Both segments resulting from the detected change-point having a duration of at least 5 minutes , and the Manhattan distance between the matrices exceeding a predefined threshold (empirically set to 7.0).) , then declare τ as a significant change-point and recursively apply CPD to the resulting segments.

3 RoleCPD

Within each formation period identified by CPD, detect role-specific change-points. First , we encode temporary role assignments as permutations $\pi_t \in S_N$, where

$$\beta_t(p) = \pi_t(X_p), \quad \forall p \in P.$$

and we follow exactly the same procedure as before using the Hamming distance between permutations instead of manhattan distance:

$$d_H(\pi_t, \pi_{t'}) = |\{X \in X \mid \pi_t(X) \neq \pi_{t'}(X)\}|.$$

and follow the same steps as FormCPD.

4 Data

4.1 Data description

The authors used the GPS data collected from the two seasons (2019 and 2020) of South Korean League 1 and 2 . The data from 809 sessions (match halves) with at least one moment when ten outfield players were simultaneously measured are split into 864 formation periods and 2,152 role periods. However, the authors couldn't share the entire dataset due to the security issue, so we just had access to the data of one sample match **17985.ugp**.
The dataframe **ugp** ,extracted from the ugp file, tracks a player's movements and speed in a game session.
The dataframe **player_periods** tracks player participation and session transitions.
The dataframe **roster** ,which is extracted from the **player_records.csv** file, lists players participating in match number 17985.
The **role_tags_true.csv** file contains **ground truth** data annotated by domain experts, including formations and role labels.

4.2 Preprocessing methods :

- **Handling Missing Values :** Missing values (NaN) in a dataframe were addressed by replacing them with valid entries extracted from other relevant dataframes, where available. This approach ensures the completeness of the data while maintaining consistency , thereby minimizing the impact of missing information on subsequent analyses.

- **Data Smoothing via Moving Average** : To preprocess the time-series data, a moving average with a window size of 3 was applied to the x and y coordinates of each player, grouped by player_id. This smoothing reduced noise while preserving movement trends, ensuring cleaner positional data for analysis.

- **Detecting and removing outliers :** A joint z-score was calculated for the positional coordinates (x,y) with a threshold of 3 used for outlier detection. Rows where either coordinate exceeded the threshold were flagged as outliers. These outliers were subsequently removed, resulting in a cleaned dataset, enhancing data quality for subsequent analysis. Then, we compared original data vs outliers plots, statistics, and boxplots.

The image shows two boxplots, comparing data before and after removing outliers.The left boxplot has a wider spread with more outliers, while the right boxplot shows a more condensed spread without outliers, giving a clearer representation of the data.

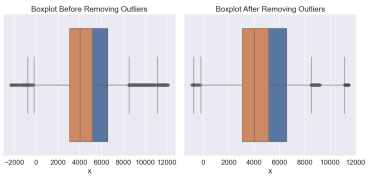

Figure 1: Boxplot before and after removing outliers

5 Results

5.1 Effect of smoothing and outliers removal:

First, we transform the ground truth change points into some kind of Gantt chart to have a better visualization. We can clearly see that the first role period end was detected at 19:18:10 but with our preprocessing it was detected at 19:18:40 which is closer to the ground truth 19:19:00. (Figs 2,3,4)

5.2 Comparison between different types of formation change point detection ('gseg_union', 'kernel_linear'...) :

(Figs 5,6)

5.3 Experimenting with various hyperparameter values:

The algorithm only detects 3 formation periods instead of 4 (Figs 7, 8) while changing max switch rate parameter . We tried to run the algo by modifying the thresholds of the scan stat (p-value, min Manhattan distance, and the min segment duration) but we didn't observe any big difference. For FormCPD , we changed Manhattan distance to Frobenius distance, the algo only detect 2 formations instead of the previous 4. It was expected because Manhattan is less sensitive to outliers and more adapted to sparse data.

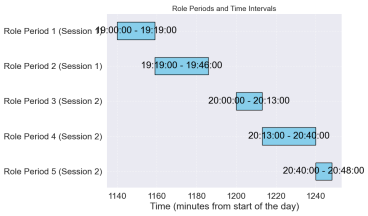

Figure 2: Ground truth annotations

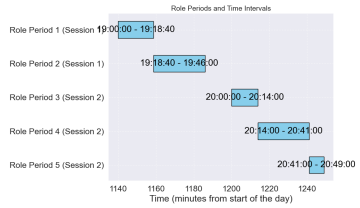

Figure 3: Predicted change points with preprocessing (gSegAvg)

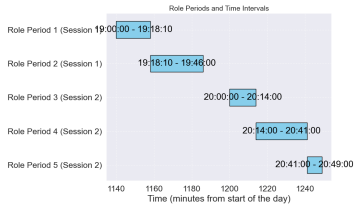

Figure 4: Predicted change points without preprocessing (gSegAvg)

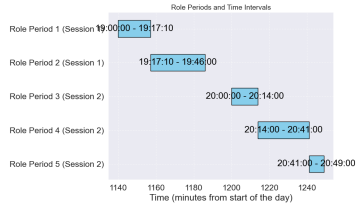

Figure 5: Predicted change points (gSeg Union)

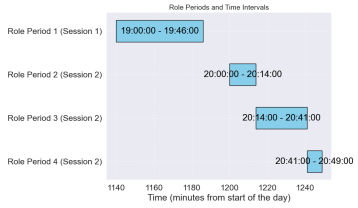

Figure 6: Predicted change points (Kernel Linear)

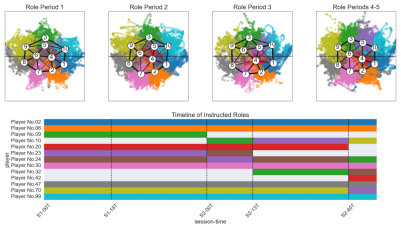

Figure 7: Detected Formation and Role periods (MAX_SWITCH_RATE=0.8)

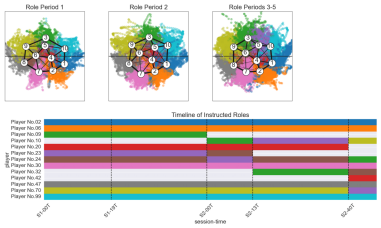

Figure 8: Detected Formation and Role periods (MAX_SWITCH_RATE=0.1)