SoccerCPD: Change-Point Detection Framework for Analyzing Team Formations and Role Changes in Soccer Matches

Omar El mansouri Mohammed Benyahia

Master MVA

January 18, 2025

Outline

1. Introduction
2. Method
   - Role Assignment via Expectation-Maximization
   - Delaunay Triangulation and Adjacency Matrices
3. Data
   - Preprocessing Methods
   - Effect of Smoothing and Outliers Removal
   - Comparison Between Different Formation Change Point Detection Methods
   - Experimenting with Various Hyperparameter Values
4. Conclusion

Problem :

- Detection of tactical breaking points in football matches $\rightarrow$ essential for understanding how formations and player roles evolve based on strategy

Challenges et état de l'art

Challenges :

- Coaches initially assign a unique role to each player, but they can change their instruction throughout the match.
- Players temporarily switch roles with their teammates.
- Abnormal situations such as set-pieces sometimes occur, in which all the players ignore the team formation.

Litterature :

- Managing to capture temporary role switches between players, but we need to make the asumption that the team formation is consistent throughout half of a match.
- Window approach to estimate formations but there is a trade-off between the reliability of detected change-points and the robustness against the abnormal situations

Input Data

Input Data: The input data consists of spatio-temporal data obtained via a GPS tracking system:

- **Player Positions:** For each time frame t, each player is represented by their coordinates $(x_p(t), y_p(t))$.
- **Sampling Frequency:** The data is collected at 10 Hz, meaning each second contains 10 frames.
- **Number of Players (N):** Generally, $N = 10$, corresponding to 10 field players for a team.
- **Initial Constraints:**
  - Each player must have a unique role assigned at every moment.

Step 1: Initialization and E-Step

Initialization:

- Each player is initialized with a unique role X_p, where $X_p \in \{1, 2, \ldots, N\}$.
- The initial spatial distributions of roles are estimated from the observed average positions.

E-Step (Expectation): Role Assignment

- For each time frame t, roles are assigned to players by minimizing a cost:

$$C_{p,k}(t) = -\log P(X_k \mid (x_p(t), y_p(t))),$$

  where $P(X_k \mid (x_p(t), y_p(t)))$ is the probability that role X_k generates player p's position at time t.
- The problem is solved using the Hungarian algorithm to minimize the total cost.

Step 2: M-Step (Maximization)

Update of Spatial Distributions:

- After roles are assigned, the spatial distributions of roles are updated to maximize their fit with the data.
- For example, if D_k is the set of player positions assigned to role X_k:

$$\mu_k = \frac{1}{|D_k|} \sum_{(x,y)\in D_k} (x, y),$$

where μ_k is the mean (center) of the spatial distribution associated with role X_k.

Iteration: Repeat until convergence.

Step 3: Output (Results)

Final Result:

- Each player p is associated with a role X_k at each time frame t, represented by a function:

$$\beta_t(p) : P \to X, \quad \text{where} \quad \beta_t(p) = X_k.$$

- This function is called the **Player-to-Temporary-Role (P-TR) Map**.

Properties:

- **Uniqueness:** Each player has a unique role at every time frame:

$$\beta_t(p) \neq \beta_t(q), \quad \forall p \neq q.$$

- **Adaptability:** Roles reflect the dynamic movements of players while remaining consistent with overall team formations.

Delaunay Triangulation and Role Adjacency Matrix

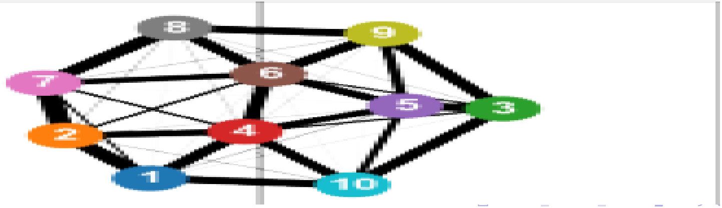
Role Adjacency Matrix $A(t)$ using Delaunay Triangulation:

- Each time frame t produces an adjacency matrix $A(t)$ of size $N \times N$, where N is the number of players.
- The elements of the matrix are defined as:

$$a_{k,l}(t) = \begin{cases} 1, & \text{if roles } X_k \text{ and } X_l \text{ are spatially adjacent} \\ 0, & \text{otherwise.} \end{cases}$$

Sequence of Matrices $\{A(t)\}_{t \in T}$:

- The matrices $A(t)$ are observed over the time interval T, typically corresponding to one half-time period.



Step 5: Matrix Distance Calculation

Matrix Representations:

- The matrices $A(t)$ are binary representations of spatial relationships between roles.

Distance Between Two Matrices:

- The distance between two matrices $A(t)$ and $A(t')$ is defined using the $L_{1,1}$ norm (Manhattan distance):

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

  where:

  - N: Number of players (matrix dimensions are $N \times N$).
  - $a_{k,l}(t)$: Element in row k, column l of matrix $A(t)$.

Purpose:

- This measure captures the structural differences between two formations at different time points.

Step 2: Breakpoint Detection with Discrete Segmentation

Objective: Identify moments when the formation changes using a graph-based segmentation method called **discrete g-segmentation**.

Key Steps:

- **Similarity Graph (MST):**
  - Construct a graph based on the sequence $\{A(t)\}$, where nodes represent matrices.
  - Edges between nodes are weighted by $d_M(A(t), A(t'))$, the distance between matrices.
- **Scan Statistic $R(t)$:**
  - Compute $R(t)$, which measures the imbalance between segments before and after t.
  - A higher $R(t)$ indicates that t is a probable change point.

Scan Statistic $R(t)$

Definition: At a given time t, the scan statistic $R(t)$ evaluates the separation between observations before and after t. It is defined as:

$$R(t) = \frac{1}{|E|} \sum_{e \in E} w(e),$$

where:

- E is the set of edges connecting nodes in the first segment (before t) to nodes in the second segment (after t).
- $w(e)$ is the weight of edge e, corresponding to the distance between the connected observations.

Interpretation:

- A higher $R(t)$ indicates a larger difference between the two segments.

Detection Criteria and Recursive Segmentation

Detection Criteria: A point τ is considered a significant breakpoint if:

- $R(\tau)$ exceeds a statistical threshold (p-value < 0.01).
- Segments before and after τ last at least 5 minutes.
- The average distance between $A(t)$ before and after τ exceeds an empirical threshold (set to 7.0).

Recursive Segmentation:

- If a point τ is detected, the method is recursively applied to the subintervals before and after τ.
- The process stops when no significant points are found.

Output Results

Results Obtained:

- **Periods of Consistent Formation:** The temporal sequence T is segmented into m periods $T_1, T_2, \ldots, T_m$, where each period corresponds to a constant tactical formation.
- **Average Matrices for Each Period:** Each period T_i is represented by an average role adjacency matrix:

$$\bar{A}(T_i) = \frac{1}{|T_i|} \sum_{t \in T_i} A(t),$$

  where $\bar{A}(T_i)$ is the average adjacency matrix for period T_i, and $|T_i|$ is the number of time frames in T_i.
- **Formation Clustering:** The average matrices $\bar{A}(T_i)$ are clustered into groups (e.g., "4-4-2", "3-5-2", etc.) using clustering methods.

Role Change-Point Detection (RoleCPD)

Objective: Extend formation change detection to a more detailed level: the individual roles of players. The goal is to detect tactical changes involving durable role exchanges between players while filtering out temporary permutations.

Input Data:

- **Formation Periods (T_i):** Each period T_i corresponds to a stable formation phase (e.g., "4-4-2", "3-5-2").
- **Temporary Role Assignments ($\beta_t(p)$):** At each time t, a function $\beta_t(p)$ assigns a temporary role to each player p.

Objective for Each Formation Period (T_i):

- Decompose T_i into sub-periods $T_{i,1}, T_{i,2}, \ldots$, where roles are consistent and durable for each player.

Step 1: Role Permutation Representation

Role Representation:

- Each time t is represented by a permutation of roles relative to an initial configuration.
- Initially, each player p is associated with a role X_p.
- At any time t, the temporary assignment $\beta_t(p)$ is expressed as a permutation π_t of the initial roles:

$$\beta_t(p) = \pi_t(X_p).$$

- Here, $\pi_t \in S(X)$, the symmetric group of permutations over the set of roles.

Step 2: Distance Calculation Between Permutations

Hamming Distance:

- To compare two time points t and t', the Hamming distance is used between the permutations π_t and $\pi_{t'}$:

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

- This distance measures the number of roles assigned differently between t and t'.
- Follow same steps as Form CPD.

Final Result: Role Period Decomposition

Decomposition of Role Periods:

- Each formation period T_i is divided into sub-periods $T_{i,1}, T_{i,2}, \ldots,$ where:
  - Each player maintains a constant role during a sub-period ($\pi_{i,j}$).
  - Temporary permutations (σ_t) are treated as minor variations within a sub-period.

Data Description

- Source: GPS data from K League 1 and 2 (2019 and 2020 seasons)
- Sessions: 809 sessions (match halves)
- Data Split:
  - 864 formation periods
  - 2,152 role periods
- Data Components:
  - **ugp**: Player movements and speed
  - **player periods**: Player participation and session transitions
  - **roster**: Player information
  - **role tags true.csv**: Ground truth annotations

Preprocessing Methods

- **Handling Missing Values**:
  - Replace NaNs with valid entries from other dataframes.It ensures the completeness of the data while maintaining consistency , thereby minimizing the impact of missing information on subsequent analyses.
- **Data Smoothing via Moving Average**:
  - Apply moving average with window size 3 to x and y coordinates. This smoothing reduced noise while preserving movement trends, ensuring cleaner positional data for analysis.
- **Detecting and Removing Outliers**:
  - Use z-score method with threshold 3
  - Remove rows exceeding the threshold
  - Compare original vs. cleaned data using boxplots

Preprocessing Methods

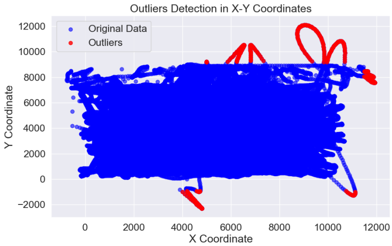


Figure: Outliers in X-Y Coordinates

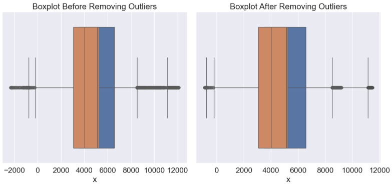


Figure: Outliers Boxplots

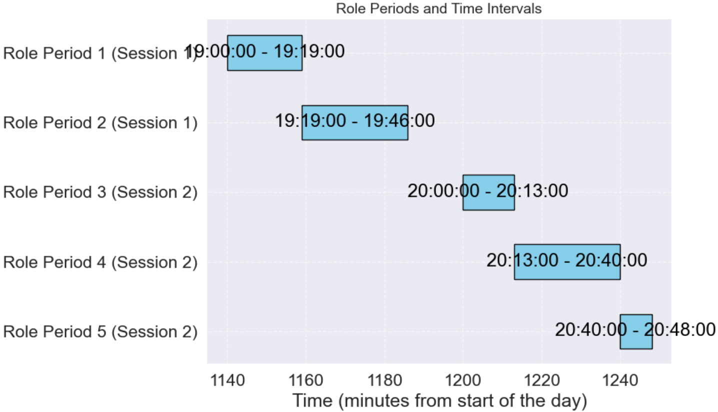
- The blue dots represent the main data points, while the red dots signify the outliers scattered around the edges.
- The boxplots show :
  - Before Removing Outliers: Wide range with high variability.
  - After Removing Outliers: Narrower range with reduced variability.

Naive Experiment : Detection and Visualization of Change-Points using directly X-Y Coordinate Data

- Calculated mean 'x' and 'y' coordinates by 'index' and combined them.
- Applied the PELT algorithm to detect change-points in the combined data.
- It didn't perform really well because by averaging the players position coordinates we don't exploit fully the data.

Effect of Smoothing and Outliers Removal

- Visualization: Gantt charts for ground truth vs. predicted change points
- Observation:
  - Improved detection accuracy with preprocessing
  - Example: Detected change point closer to ground truth



Predicted Change Points with Preprocessing (gSegAvg)

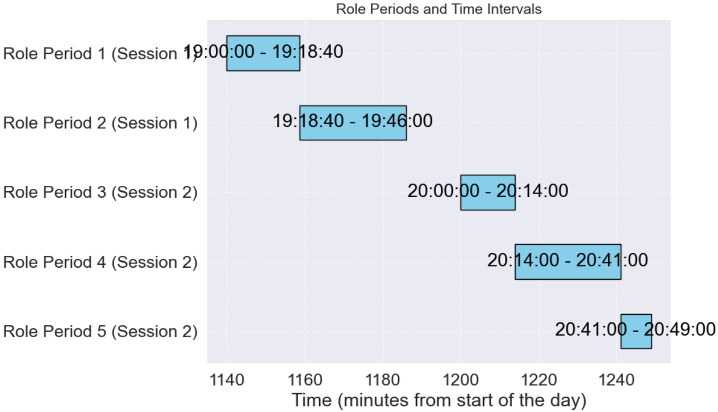


Figure: Predicted change points with preprocessing (gSegAvg)

Predicted Change Points without Preprocessing (gSegAvg)

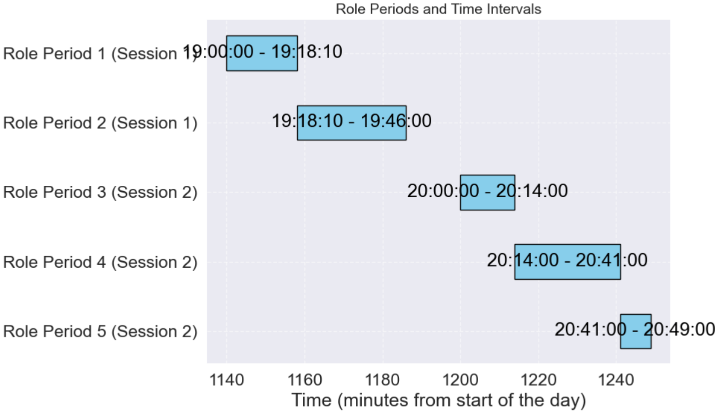
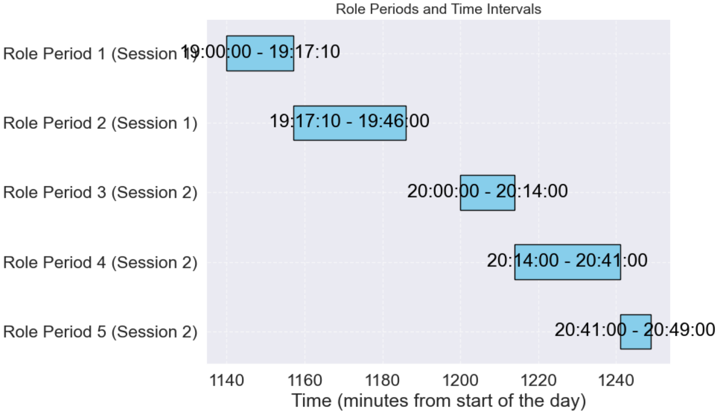


Figure: Predicted change points without preprocessing (gSegAvg)

Comparison of CPD Methods

- Methods Compared:
  - gSeg Union
  - Kernel Linear
- Results:
  - Visualization of predicted change points for each method



Predicted Change Points (Kernel Linear)

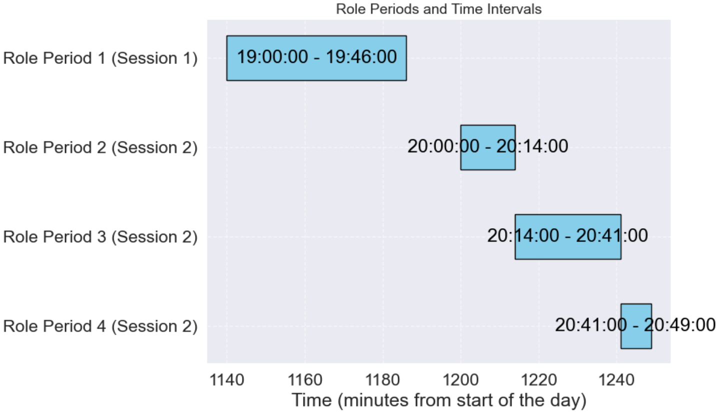
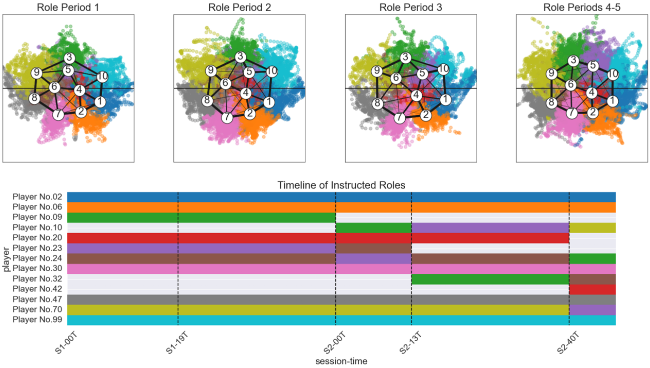


Figure: Predicted change points (Kernel Linear)

Experimenting with Hyperparameters

- Maximum Switch Rate:
  - Original: 0.8 $\rightarrow$ 4 formation periods detected
  - Modified: 0.1 $\rightarrow$ 3 formation periods detected
- Observation:
  - Lower switch rate results in fewer detected formation periods



Detected Formation and Role Periods (Max Switch Rate=0.1)

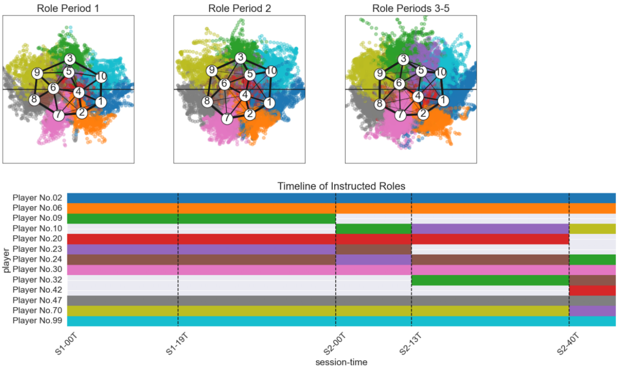


Figure: Detected Formation and Role Periods (Max Switch Rate=0.1)

Other hyperparameters

Modifications:

- We modified the thresholds of the scan statistic:
  - **p-value**,
  - **minimum Manhattan distance**, and
  - **minimum segment duration**.
- However, no significant differences were observed.

Change in Distance Metric:

- For **FormCPD**, we replaced Manhattan distance with Frobenius distance.
- The algorithm detected only 2 formations instead of the previous 4.
- This behavior was expected:
  - **Manhattan distance** is less sensitive to outliers and better suited for sparse data.

Conclusion

- Summary of SoccerCPD framework and its contributions
- Key findings from the results
- Potential future work and improvements

Thank You!

Questions?