University of Nottingham
UK | CHINA | MALAYSIA

Unsupervised Basketball Player-tracking Data Mining with Finite Mixture Models

Submitted 08/05/2018, in partial fulfillment of
the conditions for the award of the degree **BSc Computer Science.**

Shulan TANG
6516373

Supervised by Zheng LU

School of Computer Science
University of Nottingham

I hereby declare that this dissertation is all my own work, except as indicated in the text:

Signature _____

Date _____ / _____ / _____

I hereby declare that I have all necessary rights and consents to publicly distribute this dissertation via the University of Nottingham's e-dissertation archive.

Abstract

One of the major challenges of basketball analytics is a comprehensive analysis of team strategies. While various statistical metrics have been introduced to indicate teams' performance, most of them aim to present an overview of the game from a specific perspective. To get a detailed analysis of a particular offensive possession, manual annotation and observation of video recordings are often incorporated in the analytical process. This process, however, primarily relies on experiences of domain experts, with a risk of human mistake or inaccurate analysis. Additionally, video analysis is time-consuming and tedious work and only can be done by a few numbers of qualified basketball experts. Such unscalable approach motivates researchers to find alternative and more efficient methods for basketball analysis. In the recent years, the availability of player tracking data during games presents great potential for advanced automatic basketball game analysis, without the use of video. Researchers seek for the automatic interpretation and analysis of a team's strategies.

In this dissertation, we present a new machine learning method for organising and exploring basketball player tracking data. Our method describes basketball possessions by a sequence of players' actions. We first build a Gaussian distribution model for extracting representative offensive trajectories of players from the raw spatio-temporal path. We combine basketball knowledge with image processing techniques, representing key basketball event: Screen, as a pair of co-occured player trajectories. We then employ clustering techniques for action extraction. Each action corresponds to an interpretable type of player movement. We demonstrate how these trajectory clusters can be used to describe individual player behaviour by visualising each group. Leveraging this abstraction of actions, we construct a topic mixture model that describes interactions between players, resulting in a description of play structure during each procession with offensive players actively involved in the play. We incorporate basketball domain knowledge to eliminate noise from data and concisely represent team behaviour. Finally, our approach draws on the play comparison, by comparing the low-dimension topic mixtures between each pair of possessions. We evaluate our method by comparing and contrasting the output of different teams. We show that our model can efficiently capture important basketball offensive strategies and can group possessions with similar offensive structure, allowing efficient

exploration and analysis of the player-tracking dataset. By captures similar structures in teams' offence, our model facilitates complicated analysis of basketball offence. The result of the model can be further interpreted by domain experts, while saving much more time and human cost.

Dedication

This dissertation is dedicated to my favourite NBA team, Houston Rockets. I became a fan of Rockets ten years ago when they made 22 wins in a row. During the past ten years, they've been through up and downs but never give up fighting hard. This season, they are more competitive than any of the seasons for the last ten years. The joy, love and passions that I got from watching their games and playing basketball have always cheered me up during the difficult time in my life. They inspired me to pursue what I love, and without this team, I could not imagine to finish or even start this dissertation. They showed incredible motivation, work ethic and desire to win and to be the best, which has been the most significant inspiration for me. Rockets, let's get the Larry O'Brien Trophy.

Acknowledgements

I would like to express my gratitude to my supervisor Dr. Zheng Lu for his unwavering support, guidance, and insight throughout this research project. This work is a result of many discussions with him. He was instrumental in brainstorming new approaches and an integral part of this thesis. He was always there when I hit a roadblock to offer new ideas, challenge my existing methods, and improve my work. This thesis would not have been possible without the help and guidance of him. I am grateful to my family, mother, father, and little brother who have supported me along the way.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

Basketball is a complex team sports involves complicated strategies, especially in high-level professional competitions. These strategies help teams to score or defend their opposing teams efficiently. Therefore, coaches, players and analysts in teams spent much time to watch videos of their own or opposite games to improve themselvesṕlay understands others strategies. Previously, teams have primarily relied on a film analyst to begin their game break-downs. The film analyst would filter through hundreds of hours of game footage, carefully examining each possession. They seek to understand their own or opposite teams' performances to increase their chance of winning. If a player wants to explore or study a specific playerś play, e.g. someone he will defend, the film analyst would have to repeat the entire process, which is a very inefficient approach. With the availability of player tracking data in recent years, more advanced player movement patterns analysis can be conducted. However, the major group of conducted researches on player tracking data focus on discerning pre-labelled data, which means manual labour of domain experts is demanded. Besides, the number of basketball plays and the types of them are usually uncertain, and different teams typically have their unique playbooks. Therefore, it is very time-consuming and unscalable to label all offensive tactics manually. This approach is also not an ideal way to discern new strategies appeared in games.

Therefore, it is very appealing to have an efficient and automatic approach for different discerning tactics, i.e., tactical patterns, of basketball teams. Specifically, here we can define a tactical pattern as a series of frequently appeared trajectories and play events combinations that are characterised by the players who made these moves. The objective of this research is to identify commonly appeared pattern in professional basketball plays that can describe team's tactics and behaviours. The output of this line of research will be a system that can explore the player-tracking dataset and automatically discriminate the similarities and differences in basketball offenses. Two main advantages of automatic analysis motivate us undertook this project: first, powerful computational hardware can speed up analysis process, saving human labour, and achieving real-time analysis. Moreover, a pre-trained model built in the system serves as domain experts which is much more scalable and inexpensive. With both two advantages, analysts and teams can be working with our output to do the more advanced analysis.

1.2 Problem Decomposition

As mentioned before, our project is in the domain of professional basketball analysis with player tracking data. Specifically, the ultimate goal of this line of research is to build a basketball play searching system where similar plays can be automatically found. We define a play as a repeatable sequence of offensive player trajectory pairs that co-occur when they are conducting screen actions. We define the unit of basketball plays we want to explore is a possession. A possession starts when the offensive team controlling the ball and ends when a shot happens or the opposite team get the ball from them. During each possession we want to figure out what kind of play they conducted and how similar or different two possessions are. The basketball plays can also be called the offensive strategies, and they have a number of characteristics that make them difficult to be manually analysed, compared or labelled. First, different Plays have varying levels of complexity: a simple play can be composed of just one single pair of player moments, and a complex play may contain six or more pairs of moments happen simultaneously.

Each offensive possession is composed of one or more strategies. Typically, each team has their unique playbook that defines a number of executive offensive strategies to help them score. Players repeatedly practice them during pieces of training and games. However, basketball is a highly interactive sport and during the game, changes are often made due to different differennce or other factors. Sometimes, players do not follow any pre-defined patterns, and no play in their team's playbook is conducted. Therefore, it is often unclear if a play is conducted and for someone not associated with the team, it is also ambiguous to manually label plays. Given the infeasibility and limited extensibility of manual labelling, we employ an unsupervised approach to clustering similar possessions. In each group, similar movements of players occur in each offensive possession, and the team overall tend to conduct same or similar plays.

1.3 Challenges

Identifying similarity and differences between unlabeled basketball possessions introduces several challenges. We outline each of these challenges below.

1.3.1 Construction of Feature Vector

On thing hard to define is what can differentiate one play from another, as we got the player trajectory data from all across the game, it is essential to extract the most representative features from the data. This requires an understanding of basketball domain knowledge and the ability of transforming it to mathematical algorithm that can computable by computers.

1.3.2 Variable Play Lengths

In a standard NBA game, the maximum duration of a possession is 24 seconds, while a typical offense can last less than five seconds while the complex one can be longer than 20 seconds. We need to develop a method that can handle the large variance of possession times and find a metric that can measure the similarity of plays with different lengths.

1.4 Proposed Solution

Given the challenges above, we propose an unsupervised approach to solve the problem. There are mainly three steps in our approach.

First, we find moments during an offensive possessions where a screen occurs. Screen is a fundamental element of basketball offense. A screen is a blocking move by an offensive player, by standing beside or behind a defender, to free a teammate to shoot, receive a pass, or drive in to score. After setting the screen, the screener is often open to roll to the basket and receive a pass. This tactic is called pick and roll in basketball. Another basketball tactic, called the pick and pop, is for the ball handler to drive to the basket while the screener squares for a jumpshot. More importantly, a screen is illegal if the screener moves in order to make contact, and obtains an advantage; This rule enables us to define a Gaussian model to calculate the probability of a screen being conducted based on players' distances to one another and their speeds.

Next, we locate moments with high screen probabilities and extract two players trajectories that involve in the screen. We use clustering method to determine each trajectory represents which action, as we assume each cluster represent a template of action. Each possession is then represented as a sequence of action.

Finally, we fit these action sequences into latent dirichlet allocation model. This model yields a probability distribution of 'topic' for each possession. This model yields low dimension probability distribution which we can use to compute the similarity between offensive possessions.

1.5 Contributions

We briefly discuss some of the major contributions of our work. A detailed discussion is deferred to subsequent parts of this thesis.

1.5.1 Action representation

We begin by develop a methodology to convert position data into useful basketball-related abstractions. The abstractions can be activities such as a post up, screen, or handoff. The basketball-related abstractions are useful because they can be used as features in play classification. We consider the fundamental component of basketball games can we propose a method to calculate the screen probability using Gaussian distribution, converting basic basketball knowledge into calculable mathematical model. Our model shows well accuracy in capturing screen events in basketball games.

1.5.2 Unsupervised play clustering framework

We implement and evaluate an unsupervised framework to clustering unlabelled plays. This framework develop a bag of word approach used in natural language processing. This framework overcomes the impractical step of manually hand labeling plays, and gives a finer granularity to the structure of a play. It also gives the flexibility to define what a play is, which can often be ambiguous. By applying our model, each possession is then associated with probability mixtures that describe different strategies. Given the large dataset, our model is quick and easy to apply to the existing dataset and can be extended to fit in more data.

1.6 Organization of Thesis

In section 2, we introduce our dataset used, including SportVU and Play by Play dataset. Chapter 3 explains the detailed model and algorithm design of our approach. Chapter 4 presents the step-by-step implementation of our model. In Chapter 4, we evaluate our model performances based on some testing conducted. Chapter 5 concludes with a summary and a reflection of this dissertation. We also discuss a set of future work at the end.

Chapter 2

Background and Related Work

2.1 Dataset

2.1.1 SportVU

In this project, we use SportVU dataset to train our model and conduct experiment. SportVU data STATS SportVU [13] utilizes a six-camera system installed in basketball arenas to track the real-time positions of players and the ball 25 times per second. Each game is associated with one SportVU file. Each file record position, possession, and event data of the game, and the detailed structure of a sportVU file is showed in Figure 2.1. Position - Records the X, Y and Z coordinate of each player and ball on the court, while the Z coordinate of players are always zero.

Moment

The frequency of sampling positions is 25 times per second, each sample of positions is called a moment. A moment contains a 11*5 matrix, recording players and balls positions at that moment.

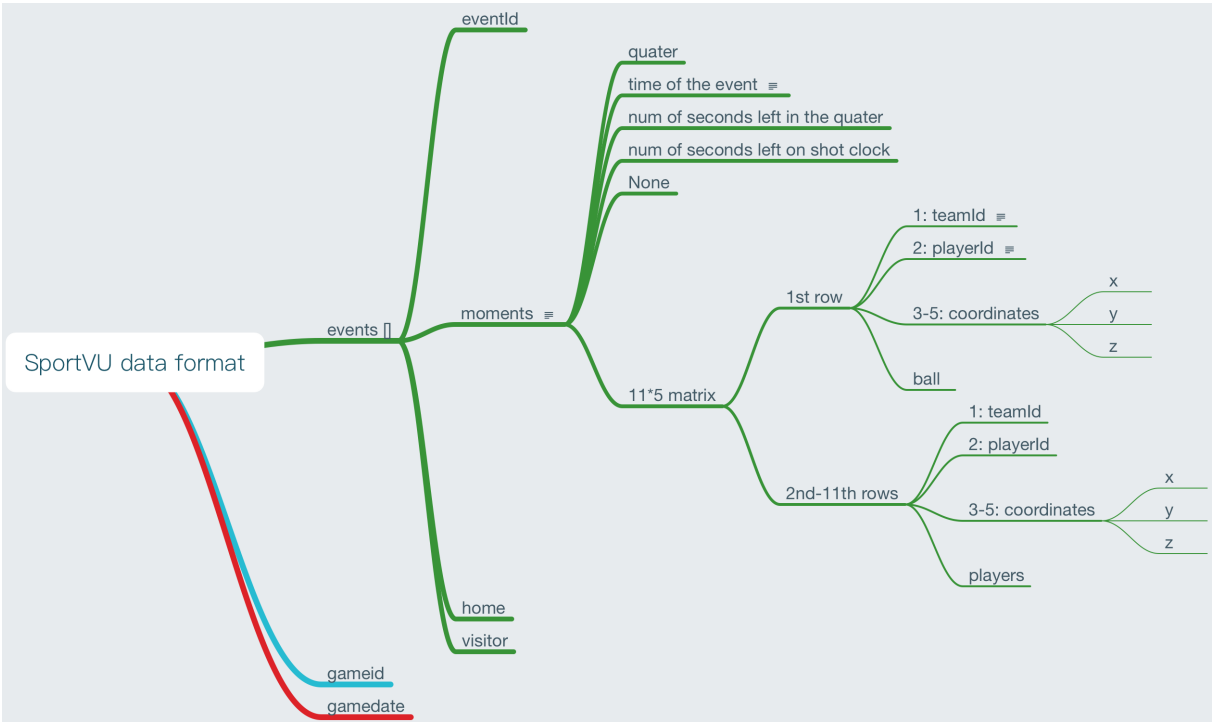


Figure 2.1: SportVU Data Format

Event

Positions are stored in each event happened during the game, each event contains a list moments as well as the player identities on the court. An event can be a free throw, field goal made, field goal missed, offensive rebound, defensive rebound, turnover, foul, dribble, pass, possession, blocked shot, or assist. With each event, SportVU records the game id, quarter, event id, and a single time when the event occurred.

Limitations

The SportVU data does have several limitations. Each position in the SportVU is represented as a point. This means the SportVU data does not capture a players posture or hand position. When a player catches the ball in the post, we cannot tell if the player keeps his back to the basket or faces up.

2.1.2 PlayByPlay Data

BigDataBalls [2] NBA Play-by-Play(PBP) stats include active lineups, shot distances and shot locations in X, Y coordinates. This piece of data describe the event type, starting and end time of the event etc. Merge of PBP and SportVU data Both PBP and SportVU data contain event items, however, since SportVU and PBP data is collected by two separate companies, we checked whether two pieces of data can be perfectly integrated. We built a visualization platform of SportVU data and check whether the eventId is the same with Play-by-play data. For the five games we check, only two of them are same, giving us no guarantee to integrate it directly. Therefore, we need to first extract each offense timestamps from PBP and then find the corresponding moments in SportVU. We also tested the completion of SportVU data, we randomly sampled 5 games and check whether the whole possession of shooting events is recorded. For total 84 shoots, we find 10 are partially missing in SportVU data, therefore we remove the incomplete ones before extracting features. We also evaluated the quality of SportVU data by manually recording every event from the TV footage, and comparing each event to the SportVU event data. In 2 out of 50 instances, we found the SportVU data missed a pass event. In over half the possessions, a dribble was missed or mistimed. This is not surprising, since tracking the ball is much more difficult than tracking players because it is smaller and moves at far greater speeds. Additionally, the exact start and end time of the possession and the coordinates of the ball would typically be off by up to 0.5 seconds. Despite these issues, we concluded that the fidelity of the data was sufficient to distinguish one play from another.

2.2 Related Work

In this section, we present an overview of recent research on player-tracking data. There is a number of researches focus on classifying play calls using pre-defined classes, based on the trajectories of players' movement, e.g. using deep neural network to classify tactics of 11 NBA play calls [14]. The models they used vary from deep neural network to

RNN. With proper data preprocessing, their training performance are generally good, hitting accuracy of more than 70. Another interesting research conducted in trajectory analysis is proposed by Zhong [16], who introduced deep convoluted neural network that use both trajectory data and video. In his work, an 1D convolution is proposed to learn trajectory variation, and he tested the trained model to predict which teams are playing the game, based on their trajectories. Compared to supervised learning, unsupervised learning gained less attention in this area, partially because of the difficulties to capture characteristic traits or lack theoretical guarantees. In 2015, Knauf et. all [7] proposed a novel class of spatio-temporal convolution kernels (STCKs) to capture similarities in multi-scenarios. By separating spatial and temporal kernel, their model can be used in multiple areas, including clustering in professional sports. However, the trajectories they clustered is not interpretable enough. Another interesting research conducted by Kostakis et. all [10] segment the overall activity stream into a sequence of potentially recurrent modes using an approximation algorithm as well as heuristic methods based on iterative and greedy approaches. They represent the data as vertex-edge network and prove it is an NP hard problem. They assume the vertices are constant and conducted experiments using both sports and social media data. In Wang et. alls research [15], they employed unsupervised learning technique to capture soccer tactics, using passes, i.e. the receiver and their positions as input, the model they created T3M can be seen as an extension of classic topic model LDA. Unlike soccer, basketball games have more scores and less passes in one offense. Therefore, passes cannot accurately capture the offense pattern in the play, also, critical moves can happen with players without ball e.g. screen. Combining this with changes in offensive and defensive player positions without ball can help to better describe what is happening. In a recent 2017 paper posted by Andrew Miller and Luke Bornn titled Possession Sketches: Mapping NBA Strategies [12], they take a well-known manifold learning technique called trajectory analysis and develops a methodology of classifying NBA actions through the use of functional mapping of segmented trajectories, called sketches, to build a vocabulary. Similar to Wangs approach, they also employ Latent Dirichlet Allocation (LDA) model.

Chapter 3

Related Algorithms

In this chapter, we describe and discuss three critical algorithms we use in the dissertation, corresponding to different steps in our project. First, we describe the Latent Dirichlet Allocation which is an extensively used algorithm in text mining. In this dissertation, we utilise this algorithm to discover the basketball topics that compose each possession. Then, we present an overview of trajectory clustering algorithm, which can help us to transfer player trajectory into meaning action labels. Finally, we briefly discuss the EM algorithm which is critical for acquiring model parameters.

3.1 Latent Dirichlet allocation

Latent Dirichlet Allocation [3] is a topic model for text structure discovery, which considers each document as a mixture of topics. A topic is represented as a distribution of words in the vocabulary. The definitions of terms used in LDA is listed as follow:

- A word is the basic unit of discrete data, defined to be an item from a vocabulary indexed by $1, \cdots, V$. We represent words using unit-basis vectors that have a single component equal to one and all other components equal to zero. Thus, using superscripts to denote components, the $V_t h$ word in the vocabulary is represented by a V-vector w such that wv = 1 and wu = 0 for u 6= v.

- A document is a sequence of N words denoted by $w = (w_1, w_2, \cdots, w_N)$, where wn

is the nth word in the sequence.

- A corpus is a collection of M documents denoted by $D = w_1, w_2, \cdots, w_M$.

For example, a corpus of Sports articles can find the corresponding topic of Basketball, with words "shoot", "defence" and "rebound", while a corpus of Science articles is more likely to have words "physics", "lab", "research" etc. Each document can be represented as random mixtures over latent topics.

The general idea is to represent each document of a corpus as a random mixture over latent topics. Conceptually, each topic is constructed by a distribution over some words, and each document is associated with a latent distribution of topics. These two distributions determine the probability of any word in a document. In LDA, the generative process for each document w can be summarised as three steps:

- Choose the number of word N for the document.
- Choose the k-dimensional Dirichlet random variable θ describing the distribution of topics where k is the number of topics.
- For each of the N words w_n.
  - Choose a topic according to θ.
  - Choose a word w_n from $p(w_n \mid z_n, \beta)$, a multinomial probability conditioned on the topic z_n.

The joint distribution of a topic mixture θ, a set of M topics z, and a set of N words w is given by $p(\theta, z, w \mid \alpha, \beta) = p(\theta \mid \alpha, \beta) = p(\theta \mid \alpha) \prod_N^{n=1} p(z_n \mid \theta) p(w_n \mid z_n, \beta)$ where α is a k-vector with components $\alpha_i > 0$ which defines the probability of the document has the distribution of θ: $p(\theta \mid \alpha) = \frac{\Gamma(\sum_k^{i=1} \alpha_i)}{\prod_k^{i=1} \Gamma(\alpha_i)} \theta_1^{\alpha_1 - 1} \cdots \theta_k^{\alpha_k - 1}$ and the word probabilities of k topics are parameterized by a k $\times$ V matrix β where $\beta_{i,j} = p(w^j = 1 \mid z^i = 1)$. As shown in the graphic model of LDA in Figure 3.1, the LDA model has three hierarchies. The α and β are corpus level parameters that fixed for generating a corpus and will not change over documents. θ is at the document level which represents the topic distribution over a

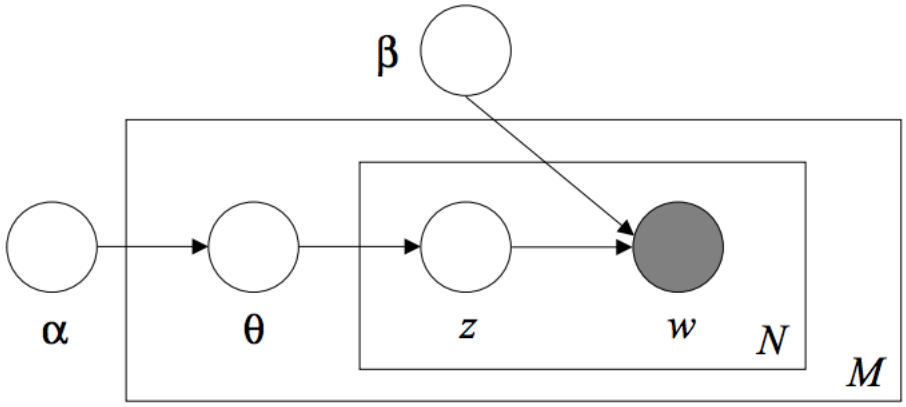


Figure 3.1: Graphic model of LDA [3]

document. Finally, z and w are at word level for generating each word in the document. The key inferential problem is to get the set of topic z and the topic distribution θ given the corpus, α and β.

$$p(\theta, z \mid w, \alpha, \beta) = \frac{p(\theta, z, w \mid \alpha, \beta)}{p(w \mid \alpha, \beta)}$$

3.2 Trajectory Clustering Algorithms

There are four major types of clustering algorithms: hierarchical clustering, distance-based clustering, topographic clustering as well as model-based clustering. Hierarchical clustering [5] is a group of an algorithm which builds a hierarchy of clusters. There are two types of hierarchy building algorithms: ascending (or agglomerative) hierarchical clustering and descending (or splitting) hierarchical clustering. The former one starts with each data in its own and successively merge clusters as the hierarchy moves up. In each merging phase, the similarity between clusters is measured, and clusters with the smallest distance are paired together, forming clusters in the higher hierarchy. The latter method work in the opposite oder. One of the most well-known and extensively used clustering algorithms, K-means, belongs to the second group: prototype-based clustering. K-means is an iterative clustering algorithm with a predefined number of cluster. The algorithm of K-means is easy to understand: it aims at minimising the within-cluster variance criterion

(intra-class inertia). Various distance-based clustering algorithms are proposed based on K-means such as fuzzy K-means [1] and trimmed K-means [6]. Algorithms in the third major group of clustering algorithms, topographic, generalise the competitive learning by updating neighbours of the winner in each iteration[9]. the objective of topographic is to minimise a cost function (distance criterion) that using a neighbourhood kernel to incorporate the topological aspect of the data such as using a Gaussian kernel. Among one of the most known topographic clustering algorithms, one can cite Self Organizing Map [8]. They are so-called deterministic as they do not define a density model on the data, which is different from model-based clustering. Finally, the idea of model-based clustering methods associate each cluster with a mixture density function [11], and each individual's assignment of cluster depends on their probabilities of clusters according to density functions. The parameters of each density function need to be estimated and this is the main process of fitting a mixture model.For example, for a Gaussian mixture model, the parameters need to be estimated is the mean vector and the covariance matrix. The most commonly used way to estimate these parameters is to perform the maximising of log-likelihood, which as we discussed in the next section, can be performed by expectation-maximisation (EM) algorithm [4].

3.2.1 Maximum likelihood via the EM algorithm

The objective of constructing many models, where the model depends on unobserved latent variables, including LDA, includes the estimation and fitting of the parameters α and β of two distributions, given a set of data. EM is an algorithm that is designed to estimate parameters in statistical models by iteratively finding the maximum likelihood or maximum a posteriori (MAP). The EM algorithm involves two-step: expectation and maximization. Taken the EM algorithm of parameter fitting for LDA, at each step, we want to maximize the log-likelihood of the data [4]: $l(\alpha, \beta) = \sum_{M}^{d=1} log\, p(w_d \mid \alpha, \beta)$.
$p(w \mid \alpha, \beta)$ cannot be computed directly. Alternatively, we iteratively find the optimizing values of the variational parameters and maximize the resulting lower bound on the log

likelihood for the model parameters α and β until the lower bound on the log likelihood converges. For the detailed process of EM algorithm for parameter estimation, refer to [4].

Chapter 4

Implementation

4.1 Preprocessing

As discussed in the previous chapter, SportVU data and Play-by-Play data are not perfectly corresponded; therefore, we first extracted the moments from SportVU events that happened during full shooting events in Play-by-Play records.

For each game we explore, we collect the metadata from the Play-by-Play file and this piece of data then help us to locate the moments that are useful in the SportVU file. Specifically, we retrieve the starting point and the ending point of each offence possession from the Play-by-Play file. To accomplish this, we first read in Play-by-Play file of the game to extract offence time periods. The columns we are interested are period, remaining time, *play length*, play id, team abbreviation, and event type. For the column of event type, there are two kinds of event we are interested in: shot and miss. Both types of these events correspond to an offensive possession. Notice that we want to study offensive players' move while they control the ball. Therefore we treated possessions whether they scored or not equally. The column 'team' refers to the player who took the shot, so we can filter by this column to retrieve the particular team we w to study later. The pair of columns *play length* and *remaining time* record length of possession in seconds and minutes: seconds left in the quarter respectively. The length of a quarter, except for overtime, is twelve minutes. For example, a row with *remaining time* equals to 0:11:35

and *play length* equals to 0:00:10 represent the offensive team starting to possess the ball when 11 minutes and 45 seconds left in the quarter and one player shot when 11 minutes and 35 seconds left. The column quarter tells which one of the four quarters this possession belongs. These three columns enable us to retrieve the exact moments of each completed offence from SportVU file so that we can study players' moves before the shot. Processing through the whole Play-by-Play file gives us all the starting and ending moments of the offensive possessions. We also eliminate possessions that shorter than one second, i.e. $play\ length < 1$.

Next, we read in the SportVU file of the same game. The SportVU file is read in as Python DataFrame. In Python, DataFrame is commonly used pandas object for data storing and analysis. It is a 2-dimensional labelled data structure with columns of potentially different types. It can be treated as a spreadsheet or SQL table which we can apply various data selection and conditional filtering efficiently.

As we mentioned before, a complete NBA game is composed of four quarters of twelve minutes, which in total gives us forty-eight minutes. However, the SportVU file is not organised as a complete linear record of 48 minutes. In fact, it breaks the whole game into events. Notice the events in Play-by-play file is not always the same as those in the SportVU file, which means we cannot directly retrieve an offence event record in the Play-by-play file from the SportVU file by event id. Besides, some moments of the game is not recorded in the SportVU file, i.e. time gaps exist between events. One piece of information we can use is that events in SportVU data are organised from the starting of the game to the end, i.e. if event id 1 $<$ event id 2, then start 1 $<=$ start 2. Consequently, we can examine the last moment of one event and the first moment of the following event to see whether there are time gaps in between. If there is, we store the start and the end moments of the time gap in a list. By going through all SportVU events, we know periods of the game that are not recorded.

In the following step, we check whether time gaps in SportVU files belong to an offensive possession in the play by play record. If there is a corresponding offence, we cannot get the full player trajectory data of this possession. Therefore we eliminate all incomplete

offensive records before extract the full offensive possessions. To accomplish this, we go through each offence period(start, end minutes and seconds as well as the quarter number) that we retrieved from the Play-by-play file, compare with the elements in the missing periods list of SportVU data. If the current missing period is intersect with the current offence period, the offence possession is labelled as missing, and we look the next offence period. If the current offence period is before the current missing period, we check the next missing record in the list. This process of checking completes when we reach the end of either the list of missing periods of SportVU or the list of offence of Playbyplay. The detailed implementation of this process is in the List 4.1.

Listing 4.1: Checking missing moments in the SportVU data

```
# find which offense vu_events are missing
for (start, end, quarter, id) in offence_timestamps:

    while missing_id < total_missing:
        cur_missing = missing_duration[missing_id]
        missing_quater = cur_missing[2]
        missing_start = cur_missing[0]
        missing_end = cur_missing[1]
        while missing_quater < quarter and missing_id < total_missing-1:
            missing_id += 1
            # next missing period
            cur_missing = missing_duration[missing_id]
            missing_quater = cur_missing[2]
            missing_start = cur_missing[0]
            missing_end = cur_missing[1]

        while missing_end >= start and missing_quarter == quarter and
            missing_id < total_missing-1:
            missing_id += 1
```

```
            # next missing period
            cur_missing = missing_duration[missing_id]
            missing_quater = cur_missing[2]
            missing_start = cur_missing[0]
            missing_end = cur_missing[1]
        # event not missing
        if missing_start <= end or missing_quarter > quarter or missing_id ==
            total_missing-1:
            break
        else:
            missing_offences.append(id)
            break
```

After removing the offence possessions that are not fully recorded, we then select moments from the SportVU file for each offensive possession. Although both Play-by-play and SportVU have a concept of the event, the whole shooting events from Play-by-play are often not fully recorded in a single SportVU event. Again, we need to go through two lists of all events from SportVU and Play-by-play to find the whole offensive periods. For each fully recorded shooting events, and the current SportVU event, if the shooting start is after the SportVU start and the shooting end before the Sportvu ends, this possession can be retrieved from this single SportVU event. Otherwise, if SportVU ends before possession starts, we check the next SportVU event. If SportVU ends before PBP ends but it starts before PBP starts as well, this SportVU contains the first half of the possession so that we can use it to construct the first half possession moment list with it and then check the next SportVU event until the whole possession is recorded. For a SportVU event, we element moments is where the actual player positions are stored.

Then, we normalised the direction of the offences and cut out the periods when the ball was not in the offensive half court and after shooting. We also normalised offence direction in the same half court. After two-quarters of the game, teams interchange their offence direction, so we also normalise the offence direction when the team shot the basket in the

backcourt, by x = $94 - x$ and $y = 50 - y$. Typically, an offensive possession starts after the opposite team shoot the ball in their half court, so the first thing players do to set up the offence is to move from the defensive half court to the offensive half court. Normally, all offensive strategies are applied after players and balls moved in their offensive court. Therefore, we discard periods when the ball is not in the offensive half court. To do this, we check whether the value of the x-axis of the ball is larger than 47, given the full court length is 94 feet (in the back half court). The end of possession is detected when the ball is near to the basket. Given the basket height and location in the NBA is and the x, y, z coordinate of the ball at each moment, we consider a shooting happened when $ball_x < 6$, $ball_y < 26$ and $ball_z > 10$. By doing this, we narrow down each play to be less than 15 seconds during which players are in their offensive court and conducted some strategies.

4.2 Speed and Distance

The SportVU data is captured by the cameras with 25 frames per second, which gives us players and ball's positions on the court every 0.04 second. Given players' positions on offences, we then calculate their speed and distances between each other and the ball. For each moment, we have a 5×6 matrix storing coordinates of the ball and five offensive players. Also, each moment also records the time in milliseconds and the number of seconds left in the quarter. Ideally, the time differences between two neighbouring moments should be 40 milliseconds since the camera records at 25 frames per second. However, sometimes due to the camera failure some frames are missing. To calculate players' and ball's speeds, we first get the Euclidean distance of their positions between two neighbouring moments and divide the difference of two timestamps. This gives us each player's speed in feet per second. Notice that if the missing moments are too much between two neighbouring records, i.e. more than 1 second, the speeds are not accurate enough. Therefore we also eliminate offence possessions that have large time gaps. Then for each moment, we calculate the Euclidean distances of each pair of players and each player and the ball. After this process, for each moment we have a speed vector with

length six (including the ball and five players) and a six by six distance matrix.

4.3 Key Action: Screen

To distinguish different plays from one another, we started by considering what defines a play from professional coaches and players' perspective. In coaches' playbook, a play is composed of a series of key movements, including screens, passes and moving from a part of the court to another. As we explained, the screen is a basic movement in the basketball game that can create space for shooting, passing and other score opportunities. Therefore, we evaluate the feature of the screen from its basketball definition and construct a Gaussian distribution model to compute and represent it from players' speeds and distances. By definition, a screen is a blocking move by an offensive player, by standing beside or behind a defender, to free a teammate to shoot, receive a pass, or drive in to score. After setting the screen, the screener is often open to roll to the basket and receive a pass. This tactic is called pick and roll in basketball. Another basketball tactic, called the pick and pop, is for the ball handler to drive to the basket while the screener squares for a jump shot. More importantly, a screen is illegal if the screener moves to make contact, and obtains an advantage. This rule forces the player who set the screen to have a very low velocity, moreover, the teammate he free needs to be close otherwise the defender will not be rubbed off. These characteristics of Screen enable us to define a Gaussian model to calculate the probability of a screen being conducted based on players' distances to one another and their speeds.

4.3.1 Screen Score

Typically, a screen is when a player's speed is low, and the other one is moving at high speed, and they need to be close to each other. Therefore, we consider a screen as an optimal one if the screener is standing still, i.e. $v = 0$ and the player being screened is moving with very high speed, i.e. $\frac{1}{v} \to 0$. The last condition of an optimal screen is that two players are close: i.e. $distance \to 0$ we calculate the screen score of the

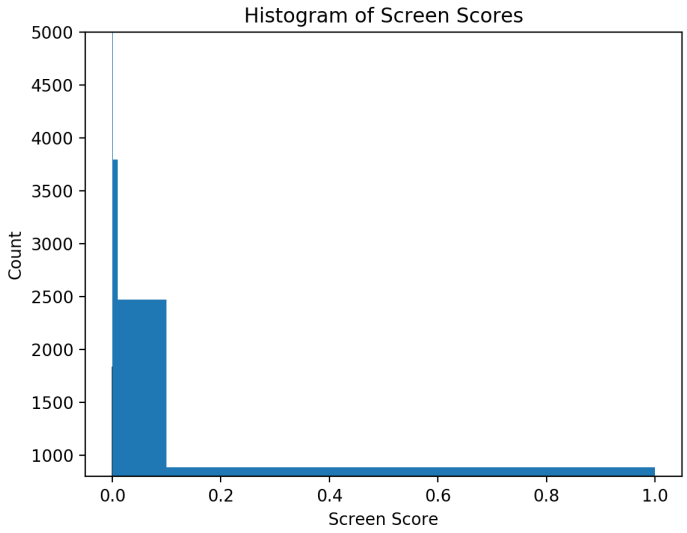


Figure 4.1: Screen Score Histogram

optimal screen by constructing three Gaussian distribution models according to three conditions listed above and multiply them to get a score. Specifically, the screen score is$Screen\,score \sim N(0, \sigma_{distance}) \times N(0, \sigma_{v_l}) \times N(0, \sigma_{\frac{1}{v_h}})$, the score of a perfect screen is $N(0; 0, \sigma_{distance}) \times N(0; 0, \sigma_{v_l}) \times N(0; 0, \sigma_{\frac{1}{v_h}})$ and for each moment the screen score of each pair of player is $\frac{Screen\,score_{pair}}{Screen\,score_{perfect}}$

This gives us a total of 10 pairs, given five offensive players. After getting screen scores at each moment, we then find moments when the score is higher than the threshold. To set this threshold, we first calculate the score for five games and observe the histogram of the score. The histogram of screen scores is shown in Figure 4.1 We looked through the game video manually to get some screen moments and check what the score is they get in our model. We set the threshold to 0.0001, above which a screen is happening.

4.3.2 Screen Abstraction

After getting the screen score for each pair of players at each moment, we then construct the abstract description for each screen. A screen is a continuous action that normally lasts for one to four seconds; therefore, we go through all screen moments and get the continuous moments where the screen scores are above the threshold. We consider two moments belong to the same screen action when their time difference is less than one second, and

the current screen length is shorter than four seconds. Otherwise, two moments belong to two separate screens. We record the start and end moments of each screen action. The detailed implementation of this process is recorded in List 4.2

Listing 4.2: Abstract screen moments to actions

```
def abs_screen(player_record):
    """
        :type player_record: list[tuple] (screen_score, time, clock, player1
            pos, player2 pos, moment_pos)
        :rtype: list [tuple] a list of screen abstraction including
            (start_moment, end_moment) for each screen
    """
    i = 1
    count = 0
    if not player_record:
        return []
    cur_screen ={}
    cur_screen['start'] = player_record[0]
    cur_screen['end'] = player_record[0]
    screen_abs = []
    while i < len(player_record):
        screen_start = cur_screen['start']
        screen_end = cur_screen['end']
        if player_record[i]['time'] - screen_end['time'] < 1000 and
            player_record[i]['time'] -screen_start['time'] < 4000: # screen
            time [1s,4s]
            cur_screen['end'] = player_record[i]# consecutive
        else:
            screen_abs.append(cur_screen)
            cur_screen = {}
            cur_screen['start']= player_record[i]
```

```
            cur_screen['end'] = player_record[i]
        i += 1
    screen_abs.append(cur_screen)
    return screen_abs
```

4.4 Trajectory of Players

After getting the time of screens and the players who conduct them, we then extract players trajectory before, during and after the screen to further distinguish each screen. To separate a whole trajectory through one possession, we first compute the speed of players and cut the trajectory when player's speed is low, i.e. v¡ 1 feet/second.

For two players who actively involved in a screen, their speeds should be different according to the definition of the screen. For the player with higher speed, we started by extracting his trajectory during the screen. We then examine moments before and after the screen to add to his trajectory if he maintains the high speed. For the other player, who's speed is slower, we extract his trajectory before and after within 2 seconds each; this can sufficiently represent his action, e.g. run toward the basket, standing still etc. This process yields two trajectories for each screen: $(Traj_{screener}, Traj_{screenee})$.

4.5 Trajectory Clustering

We assume players' trajectories during their offence can be summarised by a finite number of templates, for example, driving toward the basket, cutting along the 3-point line and executing a curl around a screen. Considering the number of action types is unknown, and labels of each trajectory are difficult to obtain in advance, we analyse them with unsupervised learning techniques. To concisely represent and summarise these action, we clustered extracted player trajectories into n clusters, each of which represents a group of similar action players take. Our objective in this step is to find partitions of player's trajectories such that trajectories in the same group tend to be more similar, indicating

players conducting the same action. To conduct clustering task, we need to first, define the similarity function and choose a clustering algorithm. As discussed in the previous chapter, there are four types of clustering methods for trajectory/curve clustering.

4.5.1 Preprocessing

To calculate the distance between each pair of trajectories, we need to first normalize the length of the player trajectories. We set the length of trajectories to 100, meaning for each trajectory, we have 100 observed sample points on it. We want these points to be smoothly distribute on the trajectory curves. If the number of points on the trajectory is more than 100, we reduce the points evenly by $(length\,of\,trajectory - 100)$. Otherwise, we add points by $(100 - length\,of\,trajectory)$. The implementation of these two methods is in List 4.3

Listing 4.3: Abstract screen moments to actions

```
def add_points(vector, number):
    step = len(vector)/number
    for i in range(number):
        new_value = (vector[i] + vector[i+1])/2
        vector.insert(i+1,new_value)
    return vector

def reduce_points(vector, number):
    step = len(vector)/number
    for i in range(number):
        del vector[i*step-i]
    return vector
```

4.5.2 K Means for Trajectory Clustering

Initialization

We implement a K means classifier to cluster the processed trajectory data. First, given the number of cluster K, we initialise K cluster centroids. Each centroid is represented as a 100×2 matrix. For each one of 100 observations of all trajectories, we calculate the minimum and maximum values of each observation and initialise each observation of K centroids. The detailed implementation of this process in Listing 4.4.

Listing 4.4: K-means: Initialize centroids

```
def _randCent(self, data_X, k):
    """
    pick k random centroids
    output: centroids matrix k*m*n
    """
    m,n = data_X.shape[1],data_X.shape[2] #get feature dimensions
    centroids = np.empty((k,m,n)) #generate a k*m*n store centroids
    minData = data_X.min(0)
    maxData = data_X.max(0)
    random_k = np.random.rand(k)
    for i in range(k):
        centroids[i] = minData+(maxData-minData)*random_k[i]
    return centroid
```

Fitting data

At each iteration, we calculate the distances between each trajectory to all cluster centroids and assign the trajectory to a cluster whose centroid is the nearest. The distance between two trajectories is simply calculated by summing up the Euclidean distances between 100 pairs of points on these trajectories. After assigning trajectories to corresponding clusters, we recalculate each cluster centroid by averaging all trajectories in this

cluster. We repeat this process for a pre-defined number of iterations or when trajectory clusters do not change in two neighbouring iterations.

4.6 Sequence of offence action

The trajectory clustering process enables us to succinctly represent each player trajectory by its cluster label, and a screen action is transformed into a pair of co-occurrence player actions. Then, each offensive possession now becomes a sequence of screen action, for example, $[(6, 9)\,(3, 7)]$ represents a possession during which two screens happened and each tuple stores the trajectory label of two players actively involved in the screen. Now, our problem is transformed to compare these sequences of screen actions.

4.6.1 Edit distance

Noted the length of each sequence is not uninformed. A short sequence may contain only one screen action while a complex offence sequence can have more than 10 actions. A way to measure the similarity between two sequences with different lengths is to compare their Levenshtein distance. Levenshtein distance is first introduced by Soviet mathematician Vladimir Levenshtein in 1965 to measure the difference between two string sequences. Informally, the Levenshtein distance between two words is the minimum number of single-character edits (insertions, deletions or substitutions) required to change one word into the other. For example, the Levenshtein distance between words “apple” and “ample” is one as one substitution can transfer the first world to the second, and the LD between words “off” and “offence” is 4, as for insertion are needed. However, the Levenshtein distance between two short sequences is more likely to be small than two long sequences or two sequences with a large difference in length. Therefore, we normalise Levenshtein distance by giving a penalty of 2 for each insertion, deletion or substitution, and divides the sum of two sequences’ length in the end. By computing the pairwise LD for all possessions, we can find most similar sequences of actions, obtained from the trajectory data. The detailed implementation of this algorithm is shown in List 4.5

Listing 4.5: Levenshtein distance

```
def edited_distance(play1, play2):
    if play1 == []:
       return len(play2)*2
    elif play2 == []:
        return len(play1)*2
    return min(distance(play1[0],play2[0])+
        edited_distance(play1[1:],play2[1:]),
              2+edited_distance(play1,play2[1:]),
               2+edited_distance(play1[1:],play2))
```

The limitation of this method is that to retrieve similar plays of one particular possession, we need to calculate it's LD to every other possession. This process can be time and computational consuming given a large dataset. Also, when both two sequences are long, the worst case time complexity of this algorithm is $O(3^n)$ where n is the length of the shorter sequence. To overcome the limitation of calculating the LD, we consider describing sequences with topics extracted from the actions.

4.6.2 LDA: Mixture of Strategies

As mentioned in the previous chapter, the topic model can be used to discover the distribution of topics in corpus [3]. Each document is described by a mixture of topics, and each topic is generated by a bag of words along with their probabilities. In our application, instead of a mixture of topics, each possession can be considered as a mixture of offensive strategies while each strategy has a bag of screen actions as its words. Each word is a unique pair of players' trajectory labels when they conduct a screen. In our study, we construct the word of our model using a vocabulary of trajectory pairs happened during screens. We then represent each possession as a bag of screen actions constructed by these words. The size of our vocabulary is K^2, where K is the number of trajectory clusters we

get from the previous trajectory clustering algorithm.

The distribution of offensive strategies over each possession $\pi^n \sim Dir_K(\alpha)\ for\ n = 1, \cdots, N$ and the distribution of action for each strategy $\phi_k \sim Dir_V(\alpha_0)\ for\ k = 1, \cdots, K$. For a single game, we set the number of strategies equals to 10, i.e. k=10 for each team. We get a vector of ten probabilities of strategies for each possession, representing the possibilities of the offensive team conducting the corresponding strategy.We can then calculate the distances of each play by calculating the Euclidean distance of the probability vector. The model output aa low-dimensional embedding of every possession so that we can calculate the similarities between them quickly. A simpler way to do this is to label each play with the strategy has the greatest probability.

Python implementation

To apply LDA algorithm on our possession data, we first construct the "corpus" with screen sequences. for each tuple represented action, we construct a "word" and add it to the document. For example, a screen tuple (3,6) is represented as word "3_6". Then, we calculate the frequencies of the words in each document using CountVectorizer module in the sklearn library which yields a word-frequency vector for each document. Then, we employ LatentDirichletAllocation package from sklean library and train the model using the word-frequency vectors of documents. We define the number of topic for the LDA model K and output a $n \times k$ matrix where each row is the probabilities of topics for a single document.

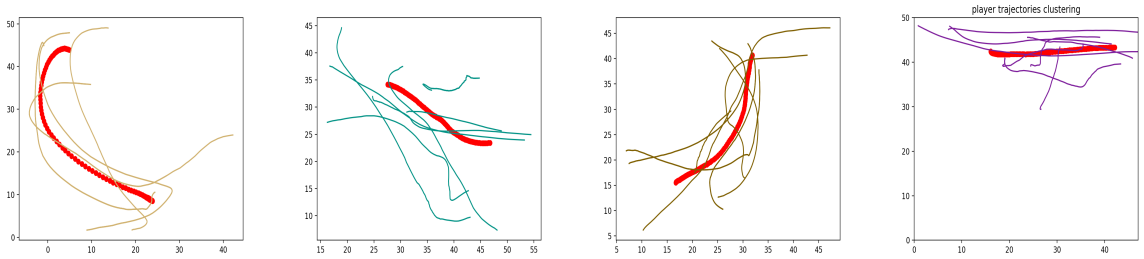
Chapter 5

Evaluation

In the previous chapter, we describe the design and implementation of our possession model, three worth noticing elements are whether the screens are captured accurately, whether the trajectories of players have labels and whether the output distribution of strategies is meaningful. To show our model can extract features describing different plays, we first examine our model's ability to capture screens by manually labelling screens in five games from 15/16 season of Golden State Warriors. Then, we justify the results of our trajectory clustering method by visualising clusters. We also interpret our possession level output of our model to show our model can efficiently catch teams strategies. Specifically, we start by looking at the strategies in different topics of the LDA model, then we compare the results of teams with different offensive styles, and we analyse our model's performance in the context of a particular type of shot: corner three. The evaluation process shows our model can capture similarities and differences in offences.

5.1 Screen Caption

To examine the efficiency of screen capture, we manually labelled all screens of State Golden Warriors in all five games, which gives us a total of 328 screens. Then, we calculate the screen score using our Gaussian distribution model and set the threshold to 0.001. We identify a screen when the screen score is larger than the threshold for more



(a) Cluster 3 (b) Cluster 4 (c) Cluster 15 (d) Cluster 19

Figure 5.1: Trajectory Clusters

than ten frames, i.e. 0.4 seconds. We show that we capture 334 screen actions, of which 298 are correct. Table 5.1 shows the performance of our Gaussian model. The precision of our model is 90.9, and the recall is 89.2. The miss-recognition tend to happen when two players are close with medium speeds.

Table 5.1: Gaussian model for Screen

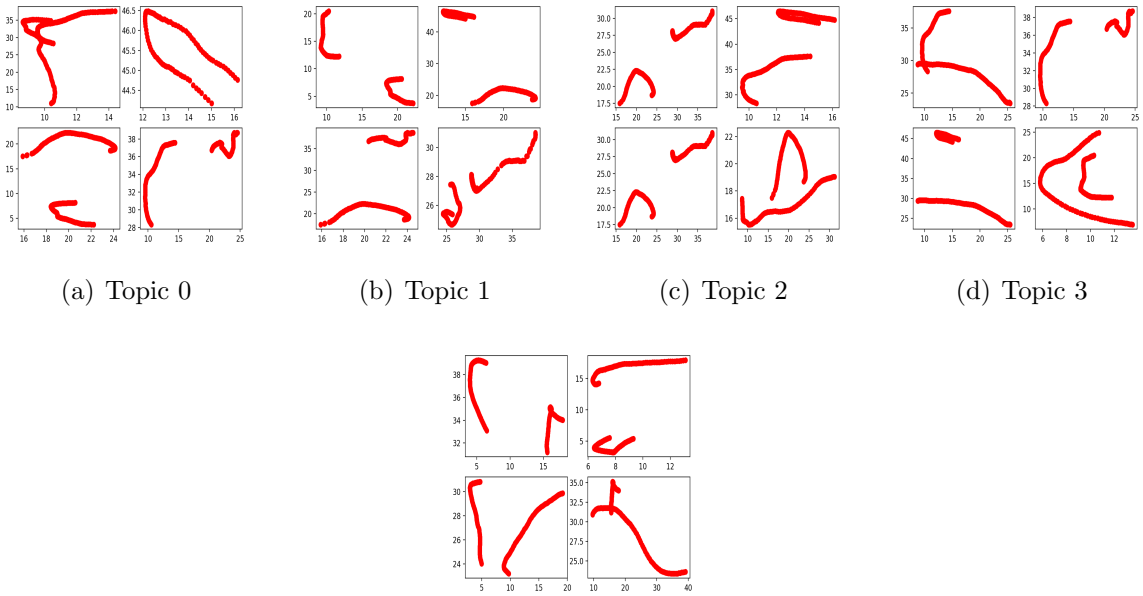
Total positive	**True positive**	**False positive**
328	298	36

5.2 Trajectory Clusters

We implement a k-means algorithm to form trajectory clusters and this enable us to represent each trajectory as a single integer. This integer can be considered as the label we assigned to each trajectory. Therefore, we need to evaluate whether each cluster include similar trajectories. We visualize the trajectory clusters to observe all segments in each cluster an the cluster centroids. Figure 5.1 shows four clusters of our trajectories. It shows the clustering algorithm is sufficiently good to group similar player action together.

5.3 Possession level output

In this section, we focus on the output of the possession level model output to demonstrate our model provide a way for basketball analysis. Our LDA model discovers the latent



(a) Topic 0 (b) Topic 1 (c) Topic 2 (d) Topic 3

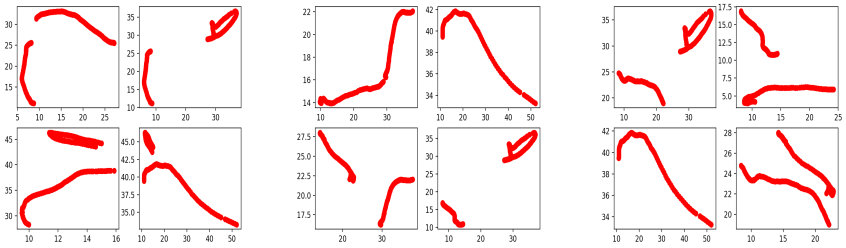
(e) Topic 4

Figure 5.2: Five Selected Topics

topics from possessions, and each topic contains screen action that conducted by offensive players. The different actions conducted on different topics represent a change in offensive strategies. In Figure 5.2, we show top four screen actions that appear on different topics, and we can see strategic patterns from different topics. For example, if possession has a high probability of topic four, it is more likely to include the screen actions in this topic: one player cut to the basket while another player moves from the basket to corner three-point line. Another concrete example can be possessions involving topic 2; offensive players tend to use curl cuts to receive an open layup. The curl cut refers to a player executing a curl around a screen. Moreover, topic two also involves V-cut, where the cutter walking the defender a couple of feet inside the 3 point line, then exploding out to receive the ball. Note that we only demonstrate four screen actions for each topic, while there are many more actions than we show.

5.3.1 Team Differences

We then apply our model for four NBA teams, to show that our model can also capture the strategy differences between teams. We select one game from Houston Rockets, Golden



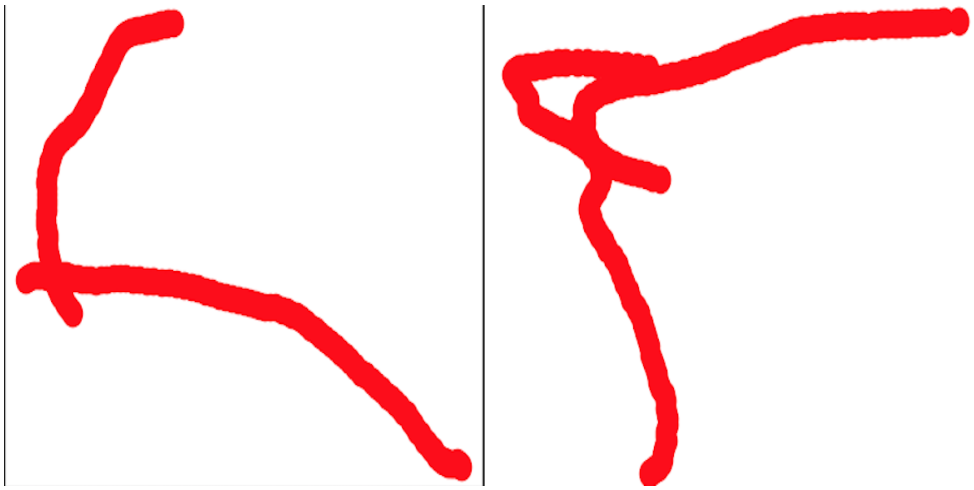
(a) Topic 0 (b) Topic 1 (c) Topic 2

Figure 5.3: Topics Shared by Houston Rockets and Golden State Warriors

State Warriors, Cleveland Cavaliers and San Antonio Spurs in 2015/16 for a total of four games. These four teams have different offensive styles in general. One interesting discovery is that Houston Rockets and Golden State Warriors share three topics that have highest probabilities in these games, Figure 5.3 show four of the actions that most likely appear in these three topics. On the other hand, Cleveland Cavaliers and San Antonio Spurs's offence are less alike there is no topic that appears in both of their games. The experiment result is not surprising as the Spurs has very different strategies compared to Cleveland Cavaliers, and Houston Rockets is often compared to Golden State Warrior as both two teams use screens heavily to create opportunities to shoot three points. From the three shared topics, we can see all three of them involves screen actions where one player cutting to the corner three points, which is one of both teams favourite shooting spots.

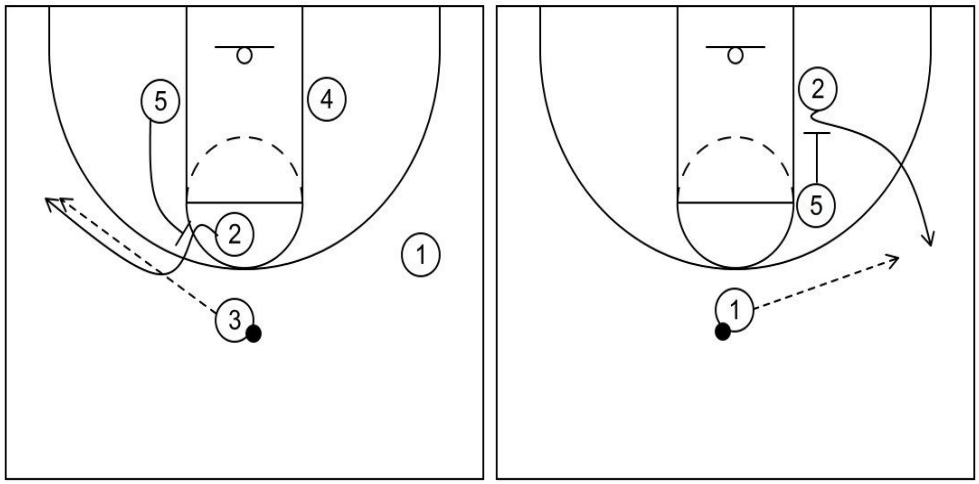
5.3.2 Three Points

In the last section, we find a tendency of Houston Rockets and Golden State Warriors' offences to screen for shooting three points. Now, we look closer to these two teams strategies of creating spaces for three-point shooting to verify that our model can identify different play structures. We use the two games of two teams that we use in the last section and manually select possessions that ended by three-point shooting. We notice that some of these possessions have similar screen actions. For example, Figure 5.4 shows two screen actions that appear in both teams' possessions. In both action, one player



(a) Action 1 (b) Action 2

Figure 5.4: Two common screen action conducted by Houston Rockets and Golden State Warriors



(a) Flare Screen (b) Down Screen

Figure 5.5: Two different screen actions

setting a screen for another player to free him from his defence and shoot a three-point in the corner. Meanwhile, we also find three-point shooting possessions with very different topic mixtures. For example, one play in Houston Rockets possessions includes a flare screen, where one player cutting along the three-point line where his teammate set a screen for him. Another play found in Golden State Warrior's possessions includes a down screen where a player sets a screen for another player near the basket, and the other player uses this screen comes from the paint area and get the space for shooting. Figure 5.5 shows the differences between these two actions.

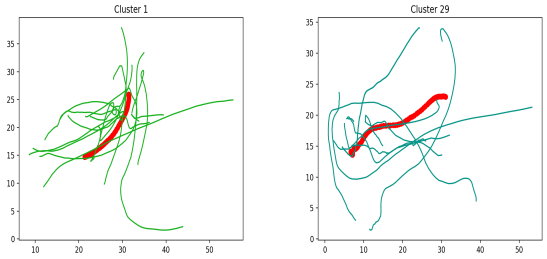
Chapter 6

Future Work

In the last chapter, we evaluate the performances of our model in different ways and demonstrate its ability to capture the similarities and differences of basketball possessions. In this chapter, we discuss how our model can be improved and further extended.

6.1 Curve modelling with Model-Based Curve Clustering Method

In our present implementation, we get the label of each player trajectory data using the cluster index it belongs to. We implemented a Kmeans clustering algorithm to accomplish this. Although this algorithm yields an overall good result for labelling trajectories, it sometimes can be affected by noise. Besides, because the speeds of players change all the time, the sampling points on the curve is the perfect even. Figure **??** shows a couple of trajectory clusters that need to be further divided and the cluster centroids can be smoothed. Therefore, we propose a model-based clustering algorithm which can generate smooth clustering centroids. Moreover, this approach is more global and data-driven and therefore need more computational power compared to our current method. The general idea of this approach is to construct a probabilistic clustering algorithm in which each cluster center is represented as a curve function. Model-based cluster with Gaussian mixture can present more accurate result by handling noise and capture unlike



(a) Cluster 1 (b) Cluster 29

Figure 6.1: Trajectory Clusters

but logically possible events.

6.1.1 Bezier Curves

Bezier curves are named after P. Bezier and are used extensively in computer-aided geometric design. Players trajectories can be considered as curves of their movements and positions at each moment is the observed point on the curve. Each curve model is represented as $B_0^n(t) = B(t) \times P$ where matrix P contains the control points, and the matrix B(t) is a Bernstein matrix of degree n. A Bernstein matrix is a generalized Vandermonde matrix which, for a vector t = [ti,t2, ., tm], where m is the number of observations we have on this curve. ti -¿ [0,1] and a given degree n, (n decides the number of control points of the curve, the large n is, the more flexible our curve model will be) has the form:

$$B(t) = \begin{bmatrix} B_0^n(t_1) & B_1^n(t_1) & \cdots & B_n^n(t_1) \\ B_0^n(t_2) & B_1^n(t_2) & \cdots & B_n^n(t_2) \\ \vdots & \vdots & \vdots & \vdots \\ B_0^n(t_m) & B_1^n(t_m) & \cdots & B_n^n(t_m) \end{bmatrix} \text{ where } B_i(t) = \begin{pmatrix} n \\ i \end{pmatrix} t^i(1-t)^{(n-i)}$$ For each cluster, our aim is to find an B(t) such that the Frobenius norm$error = \|B(t) \times P - D\|$ is minimized. The matrix D contains the data points in player's trajectory. Each row gives coefficient in a given moment t and each column is coefficient for a given control point. Therefore, we see that for a vector $t \in R^m$ and a given degree n we have B(t) as a $m \times (n+1)$. In 2-dimension, the $(n+1) \times 2$ matrix P contains the x and y coordinates

of the control points and has the following form $P = \begin{bmatrix} x_0 & y_0 \\ \vdots & \vdots \\ x_n & y_n \end{bmatrix}$

$b(t) \times P$ results to a $M \times 2$ matrix which contains m points on the curve.

6.1.2 Finite Mixture Model

Finite mixture models [11] is a latent variable model that extensively used in probabilistic modeling, machine learning and pattern recognition areas. They assumes that each class is composed of a number of sub-classes hence are very useful to model heterogeneous classes.

Definition

Let z demotes a discrete random variable in the finite set $Z = \{1, \cdots, K\}$. The density of x is represented as a mixture of K component densities: $f(x) = \sum_{k=1}^{K} p(z = k)p(x \mid z = k)$ where $p(z = k)$ is the mixing proportion and $p(x \mid z = k)$ is the component density. Each of the component density is represented as a parametric model.

6.1.3 Mixture Models for multivariate data with Gaussian

In this section, we present background knowledge on modelbased clustering of multivariate data using Gaussian mixtures. Gaussian mixtures are widely used to solve multivariate data clustering problems []. Multivariate data clustering refers to the group of clustering problems where each situation, product, or decision need to be clustered involves more than a single variable. Let us denote by $(x_1, \cdots, x_n)$ an observed i.i.d(independent and identical distributed) dataset, each observation x_i is represented as multidimensional vector in R^d. We also let $z = (z_1, \cdots, z_n)$ denotes the corresponding unobserved (missing) labels where the class label z_i takes its values in the finite set $\{1, \cdots, K\}$, K being the number of clusters. In the finite mixture model, the data probability density function is assumed to be a Gaussian mixture density defined as $f(x_i; \Psi) = \sum_{k=1}^{K} \pi_k N(x_i; \mu_k, \sum_k)$

each component Gaussian density being associated with a cluster $N(\cdot; \mu, \sum)$ denotes the multivariate Gaussian density with mean vector μ and covariance matrix $\sum$. In this mixture density, the π_ks are the non-negative mixing proportions that sum to 1 and μ_k and $\sum_k$ are respectively the mean vector and the covariance matrix for each mixture component density. The problem of clustering therefore becomes the one of estimating the parameters of the Gaussian mixture model $\Psi = (\pi_1, \cdots, \pi_k, \Psi_1, \cdots, \Psi_k)$, where $\Psi_k = (\mu_k, \sum_k)$. This can be performed by maximizing the following observed-data log-likelihood of $L(\Psi; X) = log \prod_{i=1}^{n} p(x_i; \Psi)$ with the EM algorithm.

6.1.4 Model-based curve clustering

In our case, the Mixture-model-based clustering approaches is introduced generalize the standard multivariate mixture model to the partition of curves where the individuals are presented as curves rather than a vector of a reduced dimension. We assume that each curve is drawn from one of R clusters of curves which are mixed at random in proportion to the relative cluster sizes $(\pi_1, . \cdots, \pi_R)$. Each cluster of curves is supposed to be a set of homogeneous curves modeled by a Bezier curve model. The mixture density of a curve$y_i (i = 1, \cdots, n)$ can be written as $p(y_i \mid t; \Psi) = \sum_{r=1}^{R} \pi_r \prod_{j=1}^{m} N(y_{ij}; \beta_r^T t_j, \sigma^2) = \sum_{r=1}^{R} \pi_r N(y_i; X\beta_r, \sigma^2 I_m)$ where X is the $m \times n+1$ Vandermonde matrix, β_r is the n+1*2 control point matrix of the cluster r $(r = 1, \cdots, R)$, the π_r are the non-negative mixing proportions that sum to 1 and $\Psi = (\alpha_1, \cdots, \alpha_r, \theta_1, \cdots, \theta_r)$ with $\theta_r = (\beta_r, \sigma_r^2)$. σ_r^2 being the noise variance for the cluster r, which can be represented as the covariance matrix of cluster r. The unknown parameter vector Ψ is estimated by maximum likelihood via the EM algorithm.

6.1.5 Bayes Rule and GMM

If we get the parameter for each Gaussian mixture model, we can compute the probability of curve x to be found in the cluster, but what we want is the probability of cluster v, given x. According to Bayes Rule, this can be computed by $\frac{p(x|v) \times p(v)}{p(x)}$. p(x) is computable from

Gaussian model, p(v) is p_i and p(x) is mixture density of x. when Gaussian parameter and p_i are fixed, we can determine which cluster each curve belongs to by assigning it to the cluster with the largest value of $P(v \mid x)$. now the problem becomes find the p_i and $\mu, \sum$ for each Gaussian.

6.1.6 The EM algorithm

After fixing the number of clusters, we then need to find the parameters for each Gaussian model. The aim of EM clustering in the case of regression mixtures is to cluster n iid curves $((x_1, y_1), \cdots, (x_n, y_n))$into K clusters. We assume that each curve consists of m observations $y_i = (y_{i1}, \cdots, y_{im})$ regularly observed at the inputs $xi = (x_{i1}, \cdots, x_{im})$ for all $i = 1, \cdots, n$ (e.g. x may represent the sampling time in a temporal context). Let z = $(z_1, \cdots, z_n)$ be the unknown cluster labels associated with the set of curves (time series) $(y_1, \cdots, y_n)$, with $z_i \in 1, \cdots, K$, K being the number of clusters. The way to achieve this by EM is to maximize the likelihood function, and the EM algorithm for regression mixtures starts with an initial model parameters $\Psi_{(0)}$ and alternates between the two following steps until convergence.

Expectation

Given the current guess of parameter, compute which cluster each curve belongs to. Compute the expected complete-data log-likelihood given the curves Y, the time vector t and the current value of the parameter Ψ denoted by $\Psi_{(q)}$. the complete-data log-likelihood is given by: $L_c(\Psi; Y, h) = \sum_{i=1}^{n} \sum_{r=1}^{R} h_{ir} log \alpha_r + \sum_{i=1}^{n} \sum_{r=1}^{R} h_{ir} logN(y_i; \beta^T t_j, \sigma_r^2 I_m)$ where h = $(h_1, \cdots, h_n)$ is the vector of cluster labels for the n curves and h_{ir} is an indicator binary-valued variable such that h_{ir} = 1if $h_i = r$ (i.e., if y_i is generated by the cluster r). 2) Maximization Compute the update $\Psi^{(q+1)}4$ for Ψ by maximizing the Q-function with respect to Ψ. The two terms of the Q-function are maximized separately. The first term is maximized with respected to π subject to the constraint $\sum_{r=1}^{R\pi_r} = 1$ using Lagrange multipliers which gives the following updates: $\alpha_r^(q+1) = \frac{1}{n} \sum_{i=1}^{n} \tau_{ir}^{(q)} (r = 1, \cdots, R)$.

The second term can also be decomposed independently as a sum of R functions of (β_r, σ_r^2) to perform R separate maximizations. Each maximization corresponds therefore to solving a weighted least-squares problem.

6.2 Incorporating more features

So far we segment player trajectory with there speed and use their trajectories to construct screen actions. We can construct a more complex model by incorporating more information from the game.

6.2.1 Player Role

In basketball games, offensive players have different roles and conduct different actions. For example, a screen is often happened between a point guard and the centre, as the centre is normally the tallest player on the court and can efficiently block the defender. There are, however, sometimes screens happen between two guards, which are players good at shooting three points so that both of them may get shooting opportunities. Therefore, we can further distinguish different actions based on the roles of players who conduct it.

6.2.2 Passes

Now we compute scores of the screen in plays as the screen is a fundamental element in the basketball tactics. Another important action is the passing of the ball between players. Each passing strategy contains the trajectories of passer and receiver before and after pass, and position of two players We represent the passing strategy as $((R_p, R_r), Tr_{1b}, Tr_{1a}, Tr_{2b}, Tr_{2a})$. One challenge of detecting passing event is that it may generate the error when the ball is moving fast or when the ball is passing above players head. To get accurate detections of passes, we can incorporate supervised machine learning techniques.

6.2.3 Action Order

In our current model, we employ LDA model where the order of actions is ignored. However, the order of actions does matter in many cases as possessions consists same actions with different orders can have different results. As a concrete example, a possession that starts with a post up pass might be Early Post Up, but a possession with a post up pass after two block screens is Floppy. Therefore, enforcing order of actions gives more accurate descriptions.

Chapter 7

Summary and Reflections

In this chapter, we summarize our project's design and outcomes. Based on our evaluation result, we reflect on our outcomes. We also reflect on the overall project and resource management.

7.1 Summary

In this dissertation we design and implement a unsupervised machine learning system to effectively explore basketball player-tracking data. We present a new way to extract feature that distinguish different basketball plays from one another. We incorporated LDA model which discover the underlying structure of basketball possessions. We show that our system can find the similarity of different plays and providing an alternative way for basketball professions to analyze professional games. Our model is scalable and quick than manually going through game videos.

7.2 Reflection on Time Management

I started this dissertation with planning ahead and a analysis of potential risks. The Gantt Chart is shown in Appendix C. The schedule, in reality, is slightly different due to the uncontrollable factors such as the date of a demonstration performance. In particular, as this is a relatively new area of study, I spent more time to figure out the better way

to construct feature vector for model training, as they are essential to the final model performances. While sometimes being advanced and sometimes being logged, all the important dates defined by the school were met as expected. Hence I believe the time was well-managed. In my interim report, the overall design of the system is presented and I managed to implement the design. There are some adjustments made after the interim report are written such as the representation of each screen action. Also, we choose to not add the pass action in the feature due to the difficulties mentioned in Chapter Six. This, however can be further considered in the future work.

7.3 Reflection on Resource Management

During the development of the project, all the source codes are stored using a private Github repository. The repository is updated regularly for effective version control. This meant that the loss of code will not an issue since codes can be easily rewound.
The dataset used is stored locally as well as on Github, and the processed data file is stored in separate folders, enabling the access of different versions of data file and efficient data retrieval.

7.4 Reflection on Evaluation results

In the evaluation chapter, our model shows the ability to capture the differences between different plays and can be easily interpreted by the experts or even amateur basketball fans. There are, however some ways that the model can be improved in the future. First, currently we construct our trajectory clustering method based on the point-wise similarity of each trajectory, which is not very robust to noise. Therefore, we propose a more robust model-based data-driven approach in Chapter 6. This approach can generate more accurate label for player trajectory when employing more powerful hardware. Also, more feature can be incorporated in the future to discriminate plays in more details, for example, plays conducted by different players, plays start same but with different actions.

Bibliography

[1] Bezdek, J. C. Numerical taxonomy with fuzzy sets. *Journal of Mathematical Biology 1*, 1 (1974), 57–71.

[2] BigDataBall. Nba play-by-play stats 2004 to 2017.

[3] Blei, D. M., Ng, A. Y., and Jordan, M. I. Latent dirichlet allocation. *Journal of machine Learning research 3*, Jan (2003), 993–1022.

[4] Dempster, A. P., Laird, N. M., and Rubin, D. B. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)* (1977), 1–38.

[5] Friedman, J., Hastie, T., and Tibshirani, R. *The elements of statistical learning*, vol. 1. Springer series in statistics New York, 2001.

[6] García-Escudero, L. Á., and Gordaliza, A. Robustness properties of k means and trimmed k means. *Journal of the American Statistical Association 94*, 447 (1999), 956–969.

[7] Knauf, K., Memmert, D., and Brefeld, U. Spatio-temporal convolution kernels. *Machine Learning 102*, 2 (2016), 247–273.

[8] Kohonen, T. Essentials of the self-organizing map. *Neural networks 37* (2013), 52–65.

[9] Kong, S.-G., and Kosko, B. Differential competitive learning for centroid estimation and phoneme recognition. *IEEE Transactions on Neural Networks 2*, 1 (1991), 118–124.

[10] Kostakis, O., Tatti, N., and Gionis, A. Discovering recurring activity in temporal networks. *Data Mining and Knowledge Discovery* (2017), 1–32.

[11] McLachlan, G., and Peel, D. *Finite mixture models*. John Wiley & Sons, 2004.

[12] Miller, A. C., and Bornn, L. Possession sketches: Mapping nba strategies, 2016.

[13] STATS. Sportvu optical player tracking system.

[14] Wang, K.-C., and Zemel, R. classifying nba offensive plays using neural networks.

[15] Wang, Q., Zhu, H., Hu, W., Shen, Z., and Yao, Y. Discerning tactical patterns for professional soccer teams: an enhanced topic model with applications. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2015), ACM, pp. 2197–2206.

[16] Zhong, Y. *Learning Person Trajectory Features for Sports Video Analysis*. PhD thesis, Applied Sciences: School of Computing Science, 2017.

Appendix A

Interim Report

Appendix B

Ethics Checklist

Appendix C

Project Gantt Chart