Player Motion Analysis: Automatically Classifying NBA Plays

by

Mitchell Kates

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2014

Author .... Signature redacted ....
Department of Electrical Engineering and Computer Science
September 5th, 2014

Certified by. Signature redacted ....
John Guttag
Professor, Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by ..... Signature redacted ....
Prof. Albert R. Meyer
Chairman, Masters of Engineering Thesis Committee

Player Motion Analysis: Automatically Classifying NBA Plays

by

Mitchell Kates

Submitted to the Department of Electrical Engineering and Computer Science
on September 5th, 2014, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Computer Science and Engineering

Abstract

Basketball is a team game, and an important task for coaches is analyzing the effectiveness of various offensive plays. Currently, teams spend a great deal of time examining video of past games. If teams could automatically classify plays, they could more effectively analyze their own plays and scout their opponents.

In this thesis, we develop a methodology to help automatically classify a set of NBA plays using data from the SportVU optical tracking system, which tracks the position of each player and the ball 25 times per second. The problem is made challenging by the variations in how a play is run, the high proportion of possessions where no set play is run, the variance in length of plays, and the difficulty of acquiring a large number of labeled plays.

We develop a framework for classifying plays using supervised machine learning. In our approach, we incorporate a novel sliding block algorithm that improves our classifier by accounting for the difference in play lengths. We also use a variant of the traditional one vs. all multi-class SVM. This approach is well suited to distinguish labeled plays from free motion and unlabeled plays.

This thesis demonstrates that we can use SportVU data to automatically differentiate plays. We selected a total of six plays to classify, where each play had at least 20 labeled instances. We also added a large selection of plays that were not one of these six and labeled them as Other. Our framework correctly predicted the play with an accuracy of 72.6% and an F-score of .727.

We also propose a framework, based on our engineered features, to extend our research to unlabeled plays.

Thesis Supervisor: John Guttag
Title: Professor, Electrical Engineering and Computer Science

Acknowledgments

This thesis would not have been possible without the help and guidance of many. In particular, this work is a result of many discussions with my thesis advisor, John Guttag. John was instrumental is brainstorming new approaches and an integral part of this thesis. John was always there when I hit a roadblock to offer new ideas, challenge my existing methods, and improve my work.

This thesis would also not have been possible without the expert knowledge of Jenna Wiens and Garthée Ganeshapillai. Jenna and Garthee both have extensive knowledge in machine learning, and they continually gave me great ideas and guidance.

I'd also like to thank my lab mates Guha Balakrishnan, Joel Brooks, Armand McQueen, Amy Zhao, Davis Blalock and Yun Liu, who offered ideas and feedback during group meetings.

Finally, I am grateful to my family, mother, father, and siblings who have supported me along the way. They continue to push me to be the best I can be.

Contents

1 Introduction **11**

1.1 Motivation . . . 11

1.2 Problem Decomposition . . . 12

1.3 Challenges . . . 13

1.4 Proposed Solution . . . 14

1.5 Contributions . . . 15

1.6 Organization of Thesis . . . 16

2 Data **19**

2.1 SportVU Data . . . 19

2.2 Play Data . . . 20

3 Supervised Play Classification **23**

3.1 Overview . . . 23

3.1.1 Framework Outline . . . 23

3.1.2 Fixed Length Segmentation vs. Variable Length Segmentation 24

3.2 Preprocessing and Canocalization . . . 25

3.2.1 The Makeup of a Possession . . . 25

3.2.2 Canocalization of the Possession . . . 27

3.3 Possession Segmentation . . . 27

3.4 Feature Generation . . . 28

3.4.1 General Feature Properties . . . 29

3.4.2 Positions . . . 31

3.4.3 Closeness 31
3.4.4 Passes 32
3.4.5 Ball Position 32
3.4.6 Dribbling 33
3.5 Play Classification 34
3.5.1 One vs. All SVMs 34
3.5.2 Training Play Specific SVMs 35
3.6 Probabilities to Prediction 35

4 Experimental Results 37

4.1 Optimizing Parameters 37
4.2 5-Stage Framework Performance 38
4.3 Multiple Classifiers vs. Standard Multi-Class 39
4.4 Variable Length Segmentation vs. Fixed Length Segmentation 40
4.5 Ranked Predictions 42

5 Bottom-Up: A New Language for Unsupervised Classification 45

5.1 Defining a Language 45
5.2 From Language to Classifiers 47
5.3 Defining a Play 47
5.4 Applications and Conclusions 49

6 Summary and Conclusion 51

6.1 Summary 51
6.2 Future Work 52
6.2.1 Play Options 52
6.2.2 Machine Learning Features 52
6.2.3 Implementing Bottom-Up Classification 53
6.3 Conclusion 53

List of Figures

1-1 The location of the 6 SportVU cameras 12

2-1 The Courtlytics visualization tool, inspired by Kowshik's research [5] 21

2-2 A diagram of each play 22

3-1 The 5-stage framework flowchart 24

3-2 The flow of an NBA possession 26

3-3 The algorithm used to define the start of the possession in Fixed Length Segmentation 26

3-4 Segmentation using a standard split 28

3-5 Segmentation using a sliding window 28

3-6 Grid Based Positioning and Degree Distance Positioning 29

3-7 Dribbling court segmentation 34

5-1 Degree Distance Positioning for *Perimeter* and *Blocks* 46

5-2 A Series Action for *Floppy* 48

List of Tables

2.1 Each play's run time and number of instances 21

3.1 The degree and distance boundaries for Degree Distance Positioning . 30

3.2 Types of passes 33

3.3 A Sample Output from the Framework SVMs 36

4.1 5-stage framework confusion matrix 38

4.2 The number of labeled examples and run time for each play 39

4.3 Variable Length Segmentation (VLS) vs. Fixed Length Segmentation (FLS) 40

4.4 Floppy vs. All, Variable Length Segmentation 41

4.5 Elbow vs. All, Variable Length Segmentation 41

4.6 Invert vs. All, Variable Length Segmentation 41

4.7 Delay vs. All, Variable Length Segmentation 41

4.8 Early Post Up (E.P.U.) vs. All, Variable Length Segmentation 42

4.9 Horns vs. All, Variable Length Segmentation 42

4.10 Accuracy with number of predictions 43

Chapter 1

Introduction

1.1 Motivation

In the past decade, the focus on sports analytics has grown tremendously. Analytics may not dictate a team's decisions, but as one National Basketball Association (NBA) general manager said, "analytics are always part of the conversation and internal debate" [12]. NBA 'analytics groups' are growing from one statistician to full-fledged analytics teams, and startups are emerging to help teams slice and dice data in new and innovative ways [6]. With a growing importance on sports analytics, technologies have similarly advanced to capture more data.

Historically, NBA teams have relied on a film analyst to begin their game breakdowns. The film analyst would filter through hundreds of hours of game footage, carefully examining each possession. When a coach wanted to look at something new, the film analyst would have to repeat the entire process. In 2011, SportVU fundamentally changed the way games could be analyzed. SportVU adapted a missile tracking technology to capture player and ball movements using six digital cameras placed around the catwalk of basketball arenas, as seen in Figure 1-1 [3]. This system has opened the door for a whole new era of data analysis in basketball.

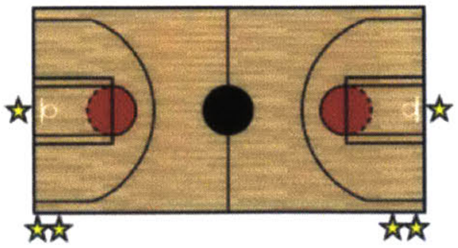


Figure 1-1: The location of the 6 SportVU cameras

1.2 Problem Decomposition

The ultimate goal of this line of research is to classify plays. We define a play as a repeatable series of player and ball movements. Plays have varying levels of complexity: a play can be as simple as a single post entry pass or as complex as a series of six or more cuts and screens. The same possession can also include pieces of multiple plays. For instance, a play can breakdown before it is finished, and a second play can be called. Similarly, if a play breaks down sufficiently quickly, it is unclear if a play was even run. For someone not associated with the team, determining their playbook and manually labeling each possession is ambiguous.

We look at six specific plays: *Elbow*, *Floppy*, *Horns*, *Early Post Up*, *Delay*, and *Invert*. We also have plays labeled as *Other*. A play is labeled as *Other* if there was no play run during the possession or the play was not one of the six plays. These plays are described in more detail in Section 2.2.

There are two main approaches to classifying plays, a top-down approach and a bottom-up approach. In the top-down approach, we start with labeled plays, and subsequently build features and classifiers that can differentiate the plays. We rely on expert knowledge from the team's analyst to label a number of plays in our dataset. Some of our features are based on abstractions of basketball activities while others use the position data in novel ways.

In the bottom-up approach, we build a complete language of basketball activities and construct our own definition of plays from the language. Some of our most effective features in the top down approach are very good at differentiating labeled plays, but they cannot be described in any basketball language. A bottom up approach relies exclusively on generating plays from basketball activities. Once a complete language of basketball activities is defined, a system can be created that converts basketball activities into plays. For example, a play in the system can be defined as any elbow pick and roll, or a play can be defined as any elbow pick and roll where there are no other ball-side players. Since these two plays may be the same or different depending on whom you ask, this approach gives you the flexibility to define plays differently.

In this thesis, we implement the top-down approach by building a 5-stage supervised learning classification framework. The implementation details can be found in Chapter 3 and its performance results can be found in Chapter 4. We also outline a proposal for a bottom-up approach in Chapter 5.

1.3 Challenges

Identifying new, unlabeled plays from existing labeled plays introduces several classification challenges. We outline each of these challenges below.

Variable Play Lengths - Plays run for a variable amount of time. Some plays are defined by a series of movements that occur over 1.5 seconds, where others can take as long as 6 seconds. We develop strategies to handle the large difference in these time frames.

Classifying Other Plays - For many NBA possessions, a pre-defined play is never run. Players are in free motion, where they can move in unpredictable ways. To complicate the problem, most of these 'random' possessions include actions (such as screens) that occur in plays. Although the base problem is to separate one play from another, we need to determine if the team even ran one of the plays for which we have training examples. In our data set, 35% of possessions will fall into the category of *Other*, which means they are either in free motion or running a play that is not in our playbook.

When we classify possessions with play labels, we expect similar plays would generate similar feature vectors. Plays labeled as *Other* can look radically different from one another. This means that two plays labeled as *Other* can generate very different feature vectors. Traditional machine learning approaches assume vectors with the same label will cluster together, but *Other* plays are scattered throughout our feature space.

Play Breakdown - Although a play may closely resemble its intended design as it starts, players quickly change and react to the defense. Play deterioration creates randomness and noise as the play progresses. Additionally, we find noise at the beginning of the play when players are setting up to run the play.

Training Set Size - Manually labeling plays is a very tedious task. NBA teams have large playbooks that are continuously changing. This makes it difficult to get a large number of labeled instances of individual plays.

1.4 Proposed Solution

The goal of our framework is to predict what play is run during a possession. A possession is defined by the SportVU data [15]. The SportVU possession data includes the start and end time for each possession during a game. SportVU also provides the X and Y positions for each player and the ball, as well as any basketball events, such as a pass or dribble.

For each play in our dataset, we separately determine if our test possession can be that play, and combine our results at the end. For concreteness, we will use the play *Early Post Up* as a running example in discussing our framework.

Our first step is to divide the possession into time blocks. If we were looking to see if our test possession was *Early Post Up* (which runs for $\sim$2 seconds), we want to check if there is a two second block during the test possession that looks like *Early Post Up*. The size of the time blocks corresponds to the length of the play we are looking at. In the case of *Early Post Up*, we would slice our test possession into time blocks that are two seconds long using a sliding window algorithm, which is described in section 3.3.

We now have a group of blocks that are subsets of our test possession, and we want to check if any of these blocks look similar to *Early Post Up.* For each of our blocks, we convert the raw position data into features that can be productively used in a machine learning classifier. We use a combination of positions, events, and interactions between players to generate a feature vector. We convert each time block into a feature vector, and pass it through an SVM that determines if the time block is *Early Post Up* or not. In addition, our SVM generates a probability that the time block is *Early Post Up.* Running each time block through our SVM gives us a binary decision and a probability that the play is *Early Post Up.*

We repeat this process with all 6 plays. We now have a binary decision and a probability for each of our 6 plays that our test possession could be that play. If the test possession is not any of the 6 plays, we predict the test possession is *Other.* If the test possession could be at least one play, we predict the play with the highest probability.

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.

$\star$ We begin to 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.

$\star$ We propose a method, **Variable Length Segmentation**, that generates time blocks that account for variable play lengths. A possession includes the transition time into the frontcourt, the play, and free motion after the play, but we only are interested in the piece of the possession where the play is run. However, each of the six plays runs for a different amount of time. This makes it difficult to choose a single length of time for our test possession. A time block that is too short would cut off a valuable piece of the play and a time block that is too long would include too much noise. The key idea is that our training vectors are generated from time blocks that match the length of individual plays. We achieve this by using a sliding window that captures the time blocks. If we simply divided the possession into blocks, the important piece of the possession could be split between two blocks. This method ensures that at least one of our blocks will contain the majority of the possession.

⋆ We implement and evaluate a **5-stage classification framework** for classifying unlabeled plays. This framework takes a top-down approach using supervised learning. This framework recognizes and makes use of the variable lengths of time that a play can run. It also handles the case of possessions that fall into the category of Other, which generate very different feature vectors. We correctly predicted the play with an accuracy of 72.6% and an F-score of .727.

⋆ We propose an **unsupervised classification framework** that uses a bottom-up approach. This framework develops a language to describe the domain of basketball activities. It then builds classifiers to recognize each basketball activity in the domain. Finally, a rule-based system defines what constitutes a unique play. 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.

1.6 Organization of Thesis

The remainder of this thesis is organized as follows. Chapter 2 introduces the SportVU and NBA hand labeled data. Chapter 3 explains our 5-stage supervised classification framework. Chapter 4 presents our experimental results. Chapter 5 proposes a framework for a new basketball language and unsupervised play classification. Chapter 6 concludes with a summary and discussion of future work.

Chapter 2

Data

This chapter describes the data we were provided, and the techniques we used to develop our feature vectors.

2.1 SportVU Data

SportVU provides a tremendous amount of data per game. For each game, SportVU collects position, possession, and event data.

Position - Records the X and Y coordinate of each player on the court, as well as the X, Y, and Z coordinates of the ball. These positions are recorded 25 times per second.

Possession - Records every possession during the game. Each possession includes the start time, the end time, the possession outcome (field goal made, field goal missed, foul, etc.) as well as the number of touches, dribbles, and passes during the possession.

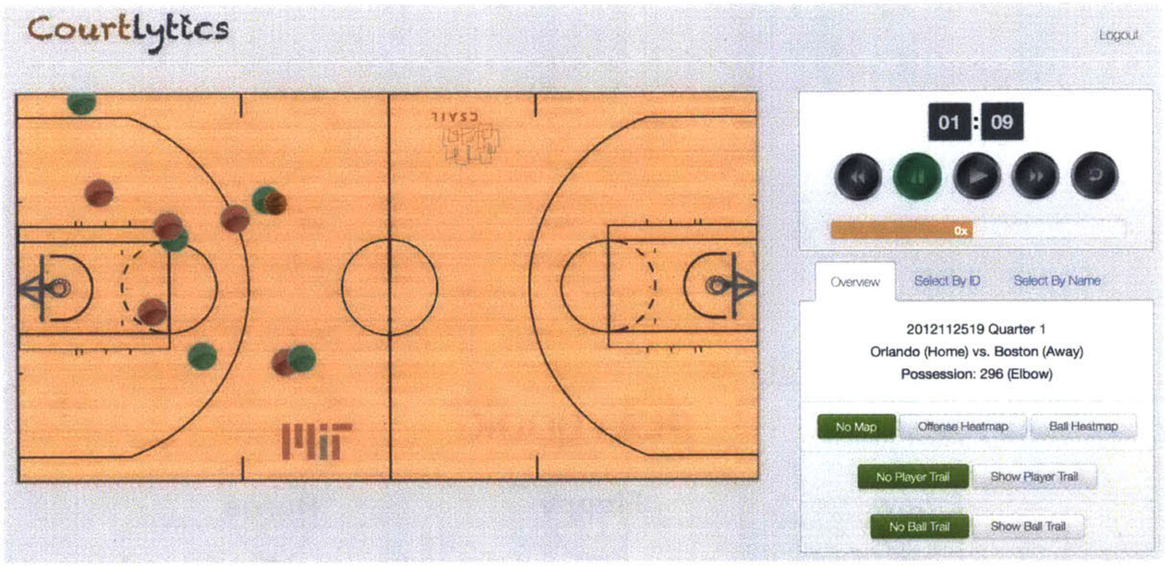
Event - Records events during the game. An event can be a free throw, field goal made, field goal missed, offensive rebound, defensive rebound, turnover, foul, dribble, pass, possession, blocked shot, or assist. With each event, SportVU records the game id, quarter, event id, and a single time when the event occurred. Even in situations where an event occurs over a period of time, such as a pass, there is only one time associated with the event. The single time of the event is the start of that action. For example, the time associated with a pass is the time when the ball leaves the passer's hands. SportVU also gives the player who completed the event, and if applicable, the defensive player associated with event.

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 player's 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.

To test the quality of the data, we randomly sampled 82 possessions from the SportVU data. We built a visualization tool, as seen in Figure 2-1, that uses the position data to recreate the play in a web based, HTML5 application [8]. We evaluated the position data by watching the data run in the visualization and eyeballing the associated position of the players with the TV footage. We evaluated the event data by manually recording every event from the TV footage, and comparing each event to the SportVU event data. In 6 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 Play Data

To determine which play was run during a SportVU possession, we relied on hand labels from an NBA team. An NBA team labeled offensive possessions from 49 games during the 2012-13 regular season. Each label included the start and end time of the possession, the game id, quarter, offensive team, defensive team, and the name of the play. Each possession was associated with a single play labeling. This is important



2012112519 Quarter 1
Orlando (Home) vs. Boston (Away)
Possession: 296 (Elbow)

Figure 2-1: The Courtlytics visualization tool, inspired by Kowshik's research [5]

because teams can theoretically run multiple plays during the course of the possession; the possession was labeled with the first play run.

We removed any possessions that had fewer than 20 instances and all possessions labeled as *transition*. In table 2.1, we list the number of instances and the run time for each play. We also focused exclusively on plays for which we had diagrams (see figure 2-2).

Play	**Instances**	**Run time**
Elbow	66	3 sec.
Floppy	70	5 sec.
Horns	21	3 sec.
Early Post Up	67	2 sec.
Delay	92	4 sec.
Invert	35	2.5 sec.
Other	186	—

Table 2.1: Each play's run time and number of instances

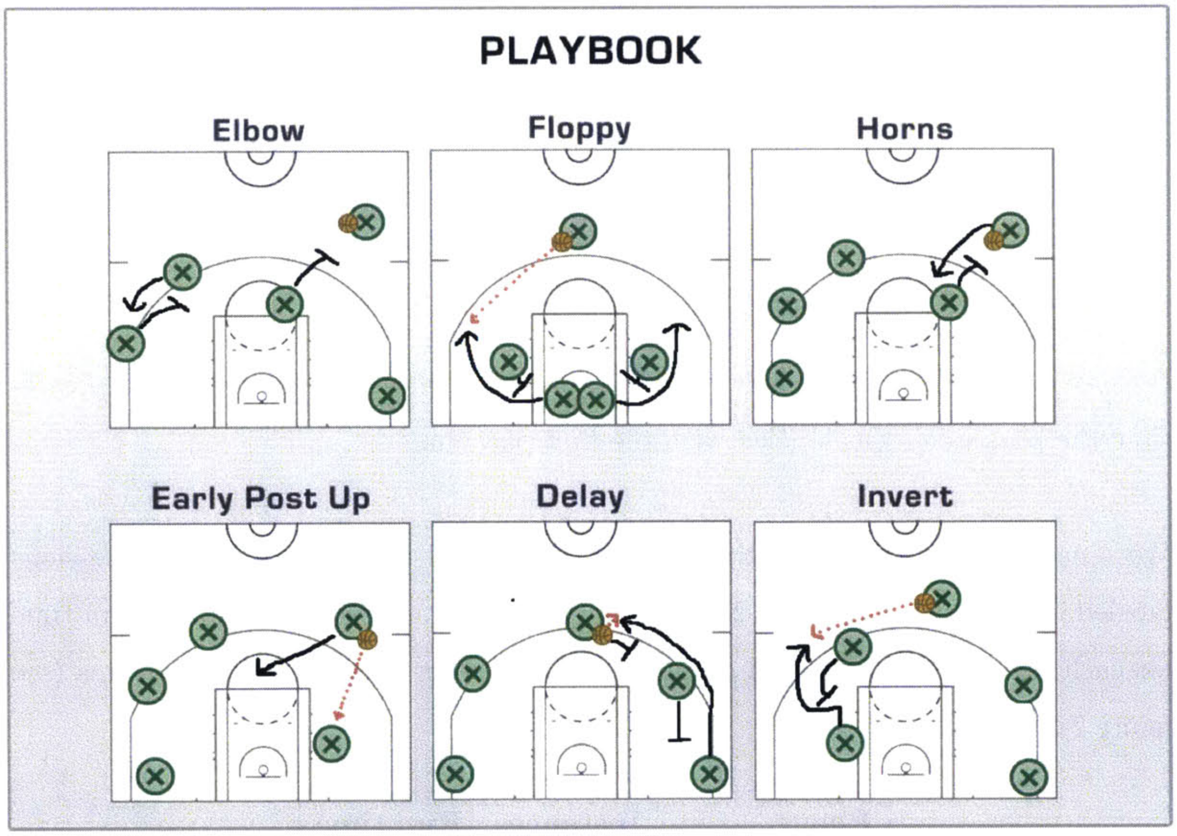


Figure 2-2: A diagram of each play

Chapter 3

Supervised Play Classification

Our goal is to create a classifier that will correctly determine what play was run during a possession. We propose a 5-stage classification framework using supervised learning. We separate our framework into the following components: pre-preprocessing the data, segmenting the possession, converting possession data into feature vectors, building play specific classifiers, and making a prediction using probability outputs.

3.1 Overview

3.1.1 Framework Outline

Figure 3-1 depicts our 5-stage framework. In step A, we preprocess and canocalize (standardize) a possession. We perform steps B-D for each of the 6 plays. In step B, we use Variable Length Segmentation to segment the possession into blocks that match the length of the play. In step C, we convert the raw position data from the time block into feature vectors for the machine learning classifier. In step D, we determine if the test possession might be the play in question by running each time block through an SVM, which gives us a binary decision and a probability. In step E, we make our final prediction based on the probabilities and the binary decision variables. Each step is explained in detail in the remainder of this chapter.

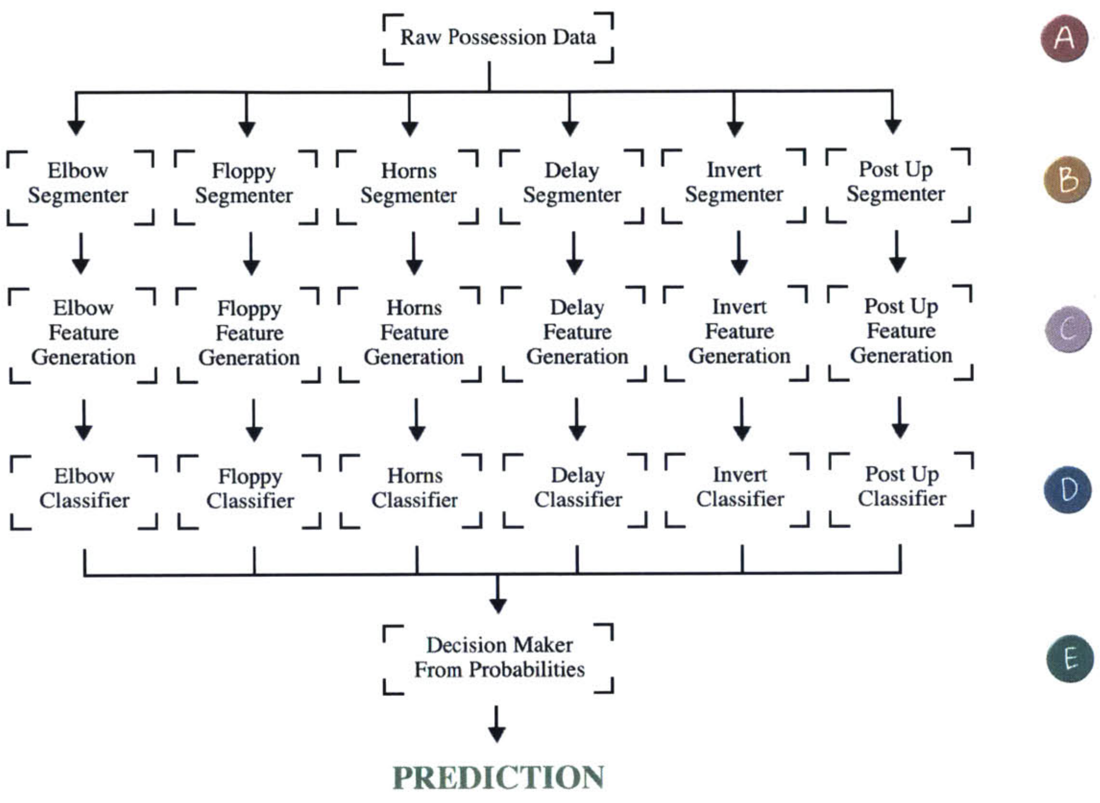


Figure 3-1: The 5-stage framework flowchart

3.1.2 Fixed Length Segmentation vs. Variable Length Segmentation

We handle the challenge of isolating the piece of the possession where the play was run using Variable Length Segmentation. In Variable Length Segmentation, we segment the possession into multiple blocks, where the length of the block matches the length of the play. Implementation of Fixed Length Segmentation is described in section 3.3.

We also built and tested a more standard method, Fixed Length Segmentation. In Fixed Length Segmentation, we generate one block per play, and we use a constant time block. For example, let's say we chose a time block of 6 seconds. In our training examples, we did not know when the play started or how long it ran for. We predict a 6 second block that starts where we believe the play began. In our test possession, we similarly found this 6-second block.

Fixed Length Segmentation is advantageous because it does not require the knowledge of each play's length. It also may be advantageous if player positions before and after the play are important in determining what play was run.

Fixed Length Segmentation requires us to optimize the time block length that we choose. A short play such as *Early Post Up* may only run for 2 seconds and a longer play such as *Floppy* will run for 5 seconds. With *Early Post Up*, the play would break down after the initial post up pass at second 2. If we used the maximum length of the two plays, the last 3 seconds in generating the feature would be unrelated to the play; they would contribute only noise. On the other hand, if we chose a length that is too short, we would cut off valuable information for the play *Floppy*.

Fixed Length Segmentation must handle the case where the possession ends before the time block ends. Many of our features rely on the time from the start of the play. In these cases, we repeat the last positions for each block. For instance, assume we have a time block of 6 seconds and our possession ends 3 seconds into the block. If one of our features takes position snapshots every second during the block, the positions would be identical at seconds 3, 4, 5 and 6.

We also needed a methodology to determine when the play began. In the next section, we outline our approach to determining when a play starts. We compare the performance of these two segmentation methods in Chapter 4.

3.2 Preprocessing and Canocalization

In this section, we describe the makeup of a possession, our algorithm for determining when a play starts (for Fixed Length Segmentation), and how we canonicalize our data.

3.2.1 The Makeup of a Possession

Each team has at most 24 seconds from the time they secure possession of the ball to shoot. Typically, as seen in Figure 3-2, a possession will follow the general flow of 5-7 seconds to advance the ball up to the frontcourt, 8-10 seconds to run a play in the half court, and 6-8 seconds where an end-of-the-shot-clock offense is run. Sometimes, a team may not run any plays, or they do not need an end-of-the-shot-clock offense.

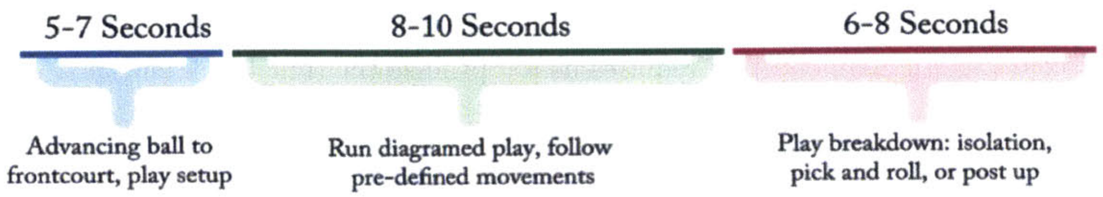
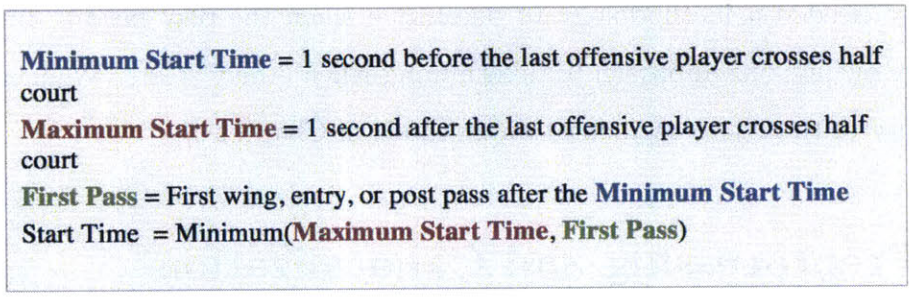

5-7 Seconds
8-10 Seconds
6-8 Seconds
Advancing ball to frontcourt, play setup
Run diagramed play, follow pre-defined movements
Play breakdown: isolation, pick and roll, or post up

Figure 3-2: The flow of an NBA possession

We outline our approach to determine the start of the play. To begin, we define the time block where the play could begin. We assume that a play can start between 1 second before the last offensive player crosses half court and 1 second after the last offensive player crosses half court. In our base case, we mark the start of the possession as 1 second after the last player crosses half court. However, if there is a post pass, entry pass, or wing pass during the this time range, we mark the event as the start of the possession.



Minimum Start Time = **1 second before the last offensive player crosses half court**
Maximum Start Time = **1 second after the last offensive player crosses half court**
First Pass = **First wing, entry, or post pass after the Minimum Start Time**
Start Time = Minimum(Maximum Start Time, First Pass)

Figure 3-3: The algorithm used to define the start of the possession in Fixed Length Segmentation

In Variable Length Segmentation, we are not concerned with finding the exact start time of the possession. However, if we preprocess our data to remove the transition time into the frontcourt, we can speed up the training of our classifiers. We simply use 1 second before the last offensive player crosses half court as the start of the possession. In both segmentation method, we remove the last second of the possession where the shot itself occurs.

3.2.2 Canocalization of the Possession

It is important that all training examples for a given play look similar to one another. However, plays may take place on either the near or far half of the court, and left or right side of the half. To achieve uniformity, we flip any possessions on the far half of the court across to the near side and any on the left side across to the right side. We determine whether a position took place on the left or right side based on the average position of the ball. These preprocessing steps ensure that every play is run on the near right side of the court.

3.3 Possession Segmentation

Our preprocessing removes the beginning of the possession where the ball is advanced to the front court and the tail of the possession. This leaves the general time window when the play was run. However, as discussed above, each play can run for a variable amount of time.

We know the maximum run time for each kind of play in our set of labeled data. We also have the time when the play started during the possession. Based on these two pieces of information, we can block the possession into different time blocks with lengths corresponding to the maximum runtime of the play we are testing. In other words, the classifier that determines if the play is *Early Post Up* or not would break the possession into time blocks of 2 seconds, and the classifier that determines if the play is *Floppy* would break the possession into time blocks of 5 seconds. With *Early Post Up*, if our total possession length is 8 seconds, we can separate the position into 4 blocks that are each 2 seconds long.

This type of segmentation can be problematic if the 2-second block when *Early Post Up* occurs falls exactly between two blocks. Figure 3-4 depicts a possession that runs for 8 seconds. The play *Early Post Up* occurs from seconds 1-3. This means our play is split equally between two blocks, and neither block may look like *Early Post Up*.

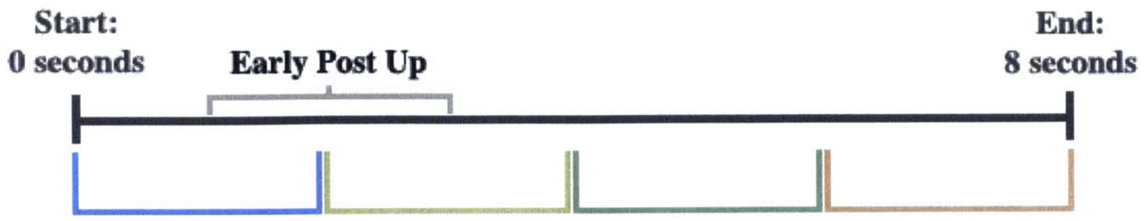


Figure 3-4: Segmentation using a standard split

In our framework, we use a sliding window to determine the subsequent block's start time. As seen in Figure 3-5, if we use a sliding window of 1 second, the first block starts at second 0, and the second block starts at second 1. This ensures that the majority of *Early Post Up* will occur in a single time block.

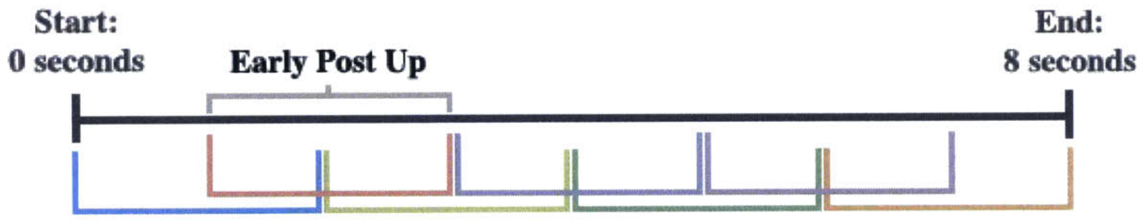


Figure 3-5: Segmentation using a sliding window

3.4 Feature Generation

To generate features, we consider what differentiates one play from all others. When teams run plays during practice, each player closely follows their ideal position. In games, there is a larger variance between where players are supposed to be and where they actually end up. Basketball is a read and react game, and the defense often dictates what the offense does. Despite the large variance in positions, there are often signature events that can be used to classify the play. For example, a signature event may be a double screen or a backdoor cut. We generate a feature set that uses a combination of general player positions along with signature events.

3.4.1 General Feature Properties

Our features rely heavily on position and time. We explore several ways to break up the court, as well as several ways to capture the time when an event occurs.

Segmenting the Court

We considered two main types of court segmentation, Grid Based Positioning and Degree Distance Positioning. In Grid Based Positioning, the court is divided into a grid, and locations are recorded by X and Y coordinates. In Degree Distance Positioning, locations are recorded by the distance and angle from the basket (i.e. top of the key is 90°, right corner is 0°)

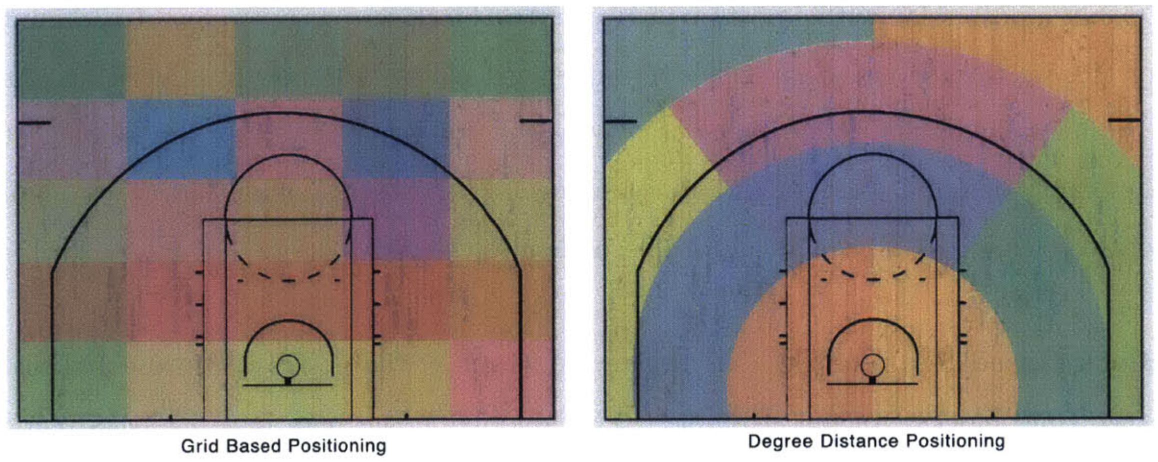


Figure 3-6: Grid Based Positioning and Degree Distance Positioning

We use Degree Distance Positioning in our framework because it is a more natural way to split the court. The court is divided into areas that coaches and players can refer to. For example, capturing locations such as *top of the key* is easier to define with Degree Distance Positioning (ex. Distance >25 feet and 45° <Degrees <135°). In Grid Based Positioning, the *top of the key* would include pieces of several different areas. Table 3.1 outlines the cutoffs for different distances and radians. We use expert knowledge to manually determine the inside, mid-range, perimeter, and outside.

Distance	Degree Cutoffs
<10 feet	90°
10-18 feet	45°, 90°, 135°
18-25 feet	45°, 90°, 135°
>25 feet	90°

Table 3.1: The degree and distance boundaries for Degree Distance Positioning

Sequencing

It is important to capture the time and order in which events occur during the possession. For 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.* To account for this we take several approaches to sequencing in our classification algorithm. As a running example, assume that our play length is 6 seconds and we are looking at the binary decision, "did an on-ball screen occur?" Further assume that two on-ball screens occurred 1.2 seconds and 4.1 seconds into the play.

Block Based Sequencing - We divide the length of the possession into equal distance time blocks. In our example, we might divide the play into 6 blocks (each representing one second) and create 6 features, in which case our resulting 6 features would be [0,1,0,0,1,0], each indicating whether a screen occurred during that block.

Multiple Block Based Sequencing - With Block Based Sequencing, we drew hard cutoffs at specific times during the possession. This meant that a screen occurring at 3.99 seconds and 4.01 seconds would create different feature vectors. With our small number of examples, we may not have the benefit of having enough instances of each situation. We can create time blocks of 1 and 2 seconds, to create 6 and 3 features respectively. Our final feature vector would be [0,1,0,0,1,0]+[1,0,1] = [0,1,0,0,1,0,1,0,1].

No Sequencing - We look at the entirety of the possession all at once. Although this method misses some important information about when events occur, it significantly reduces the number of features. With our limited number of examples, we must weigh the benefits of a smaller feature space against the limitations of not capturing when events occur during the possession. In our example, we would create a single variable that simply counts the number of on-ball screens during the possession. In this case, our single feature would be [2].

In our features, we used a combination of block based sequencing and no sequencing. Given the small number of training examples, Multiple Block Based Sequencing created vectors that were too large for our dataset. In future work with more labeled data, Multiple Block Based Sequencing may be a more advantageous approach. No sequencing was a more logical choice when the occurrence of the event rarely happened more than once, such as an on-ball screen.

3.4.2 Positions

Each player's positions provide critical information about the play being run. We look at the player's position midway through the time block and use Block Based Sequencing to generate the features.

3.4.3 Closeness

A play is often defined by its interactions between players, and more importantly, how they interact with one another. Even in plays such as isolations, the play is recognized by the lack of interaction between the player with the ball and the remaining 4 players. Interactions can happen between players on the ball and off the ball. For example, the play *Floppy* is defined as two post players who set screens on each block that two other players use.

Measuring when players get close is a good metric to capture screens. Basketball players are taught to go shoulder-to-shoulder on screens, which means that the two offensive players should be so close they touch shoulders (this prevents the defense from getting between the two offensive players). The shoulder-to-shoulder advice is similarly applied to handoffs, which are screens with a change of possession from one player to another. We cannot see if players gets shoulder-to-shoulder in the data, but we can see when they get very close to one another using the X and Y data.

We capture 3 types of closeness during a possession: an off-ball screen, an on-ball screen, and a handoff. An instance of closeness is defined as two players who come within 5 feet of one another for at least 3 frames of SportVU data (0.12 seconds). Although in most cases the players are much closer than 5 feet, there were a significant number of examples where players miss screens, or screeners slip before setting the screen. The larger threshold helps capture these instances, while still largely avoiding false positives.

3.4.4 Passes

Passes represent an important component of a play. To determine when a pass occurs, we look at all the associated event data of a possession. Since the event data only has a single time for a pass, we need to calculate it's start and end time. We look at the events prior to the pass and after the pass. Events such as *dribble* or *possession* signify the pass has not yet begun or the pass is already complete.

Bringing in expert knowledge about pass types helps reduce the number of features related to passes. We break down passing into 6 basic pass types, and sequentially check if a pass qualifies, as seen in Figure 3.2

3.4.5 Ball Position

The path of the ball and the ball's position in relation to players can expose some interesting details about each play. The most effective feature to separate the similar plays *Elbow* and *Horns* is the number of ball side players throughout the possession.

Pass Type	Pass Qualifications
Interior Pass	Passes that occur within 12 feet of the basket. This type of pass is usually seen on drives to the basket or basket cutters after a post up.
Handoff Pass	Passes that start and end within 5 feet of one another.
Corner Pass	A pass that ends in the 20x10 foot box in each corner of the court.
Post Pass	A pass to an interior player. The player can be as far out as 19 feet (some post ups can occur this far from the rim). The receiver of the post pass must also not have any other offensive players within 5 feet, and no offensive players on the same side of the court that are closer to the basket. These last two metrics are important to differentiate a post up pass from a player curling off a screen on the block.
Wing Pass	A pass that ends on the outer third of the court.
Perimeter Pass	All other passes.

Table 3.2: Types of passes

Similarly to how we generate features for player's positions, we also use Degree Distance Positioning to extract features that will describe the ball's position throughout the possession.

3.4.6 Dribbling

We are interested in capturing when a player dribbles from one region of the court to another. If a player is dribbling, but remains in the same general spot on the court, they are not advancing the play. We ignore instances where players dribble but do not move anywhere.

We break down the court into areas, and record every time a dribbler crosses one of the boundaries and dribbles at least 8 feet in total distance. If a single dribbler crosses more than one boundary, we record each boundary crossing as a separate event. In our feature vector, each region had one feature per adjacent boundary. In our diagram, the purple region would have 3 features associated with it because there are three adjacent areas.

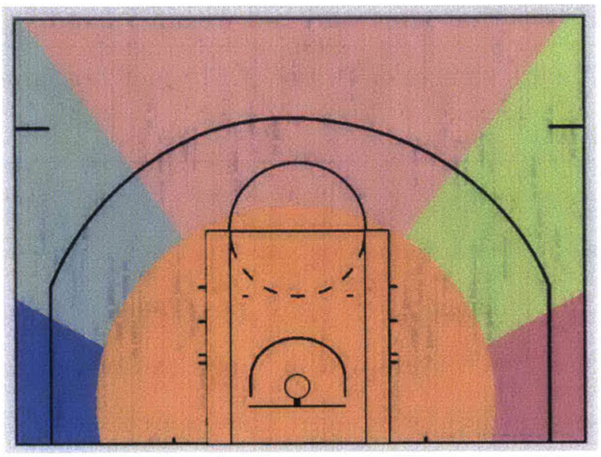


Figure 3-7: Dribbling court segmentation

3.5 Play Classification

3.5.1 One vs. All SVMs

We have a supervised learning multi-class classification problem [2]. We considered both One vs. All SVM classifiers and Pairwise SVM classifiers. In One vs. All SVM classification, there is a binary SVM for each class that separates members of that class from members of all the other classes. In Pairwise SVM Classification, there is a binary SVM for each class that separates members of that class from members of another class [9]. With both approaches, there is a voting scheme to determine the final prediction. A comparison between these two methods shows that both approaches typically outperform a single multi-class ranking SVM, which uses one SVM to classify all the classes [1].

We also had *Other* plays, which have feature vectors that can look quite different from one another. This means we could not simply add *Other* as another class in our SVM. Rather, we took the approach that if an unlabeled possession is not one of the plays we test for, then it must be *Other*. To implement this approach, we built a classifier for every play label except *Other*. All our classifiers in the 5-stage framework use an SVM with a radial basis function kernel [13, 16]. The radial basis function kernel outperformed both a linear kernel and a distance-weighted K Nearest Neighbors classifier [4].

For each classifier, we generate a probability along with a binary decision. A probability can help us compare outputs from the different SVMs. To generate a probability from our SVMs, we use a modified version of Platt Scaling using Newton's method with backtracking [14, 17].

3.5.2 Training Play Specific SVMs

As described above, each play class requires its own SVM. We start with the maximum length of the play, which is used as the length of our possession's blocks. For each negative training possession, we split the possession into blocks using our Variable Length Segmentation. Each block becomes one negative feature vector in our classifier. For each positive training possession, we have the labeled start time when the play starts. We generate one time block that starts at the labeled start time. A positive training possession produces exactly one positive feature vector in our classifier.

3.6 Probabilities to Prediction

Once we run our time blocks through each classifier, we need to generate a result from the list of probabilities. In some cases, we will be very confident in our prediction, while in others we may be unsure about our prediction. During the course of a possession, players move quite a bit. It is very likely that the motion of one play can closely mimic the motion of another play. For example, *Elbow* and *Horns* are very similar. The only difference is that *Elbow* has 3 ball side players and *Horns* only has

2 ball side players. If one player who is not ball side starts to wander to the ball side, it becomes unclear which play was run.

Correctly predicting the play on the first try is the ultimate goal, but having multiple guesses provides valuable information. For example, a coach may want to look at all possessions that resemble *Elbow.*

For a given play, each time block is run through the SVM. The SVM outputs a binary decision and a probability associated with that decision. In certain cases, the output can have a negative prediction, yet the positive probability is greater than 50%. We look at the binary decision first, then the probability. The final probability associated with a play is the maximum probability from all the time blocks.

At this point, we will have a binary decision and a final probability for each play. If no plays have a positive result, we predict the play is *Other.* Otherwise, we look at all the plays with a positive result, and predict the play with the highest probability.

We also generate a list of predictions. First, we take all the plays with a positive result and predict them based on their probability. Second, we predict *Other.* Finally, we take all the plays with a negative result and predict them based on their probability. Table 3.3 illustrates an example output with two positive results. This table would output the predictions 1. *Elbow*, 2. *Horns*, 3. *Other*, 4. *Invert*, 5. *Early Post Up*, 6. *Floppy*, 7. *Delay*

Play	**Binary Decision**	**Positive Probability**
Floppy	0	3%
Elbow	1	83%
Horns	1	51%
Invert	0	52%
Delay	0	2%
Early Post Up	0	3%

Table 3.3: A Sample Output from the Framework SVMs

Chapter 4

Experimental Results

In this section, we present the experimental results for our 5-stage framework. In section 4.1, we evaluate the performance of our 5-stage framework. In section 4.2, we compare the strategy of using multiple classifiers vs. using a standard multi-class approach. In section 4.3, we compare Variable Length Segmentation vs. Fixed Length Segmentation. In section 4.4, we evaluate our performance with multiple predictions per possession.

4.1 Optimizing Parameters

We had several parameters that we wanted to optimize. Our framework is built from six individual One vs. All classifiers. Since it was too computationally expensive to optimize each One vs. All classifier concurrently, we optimized the parameters for each classifier separately.

We could not simply perform one split of the data into training and test data sets where we optimize on the training data and evaluate on the test data. We had as few as 4 examples in the test set for the play *Horns.* The results would be too unreliable with such a small number of test points. Instead, we ran 10 iterations where we split the data into different training and test sets using an 80%/20% stratified split. We optimized the set of parameters on the training data by averaging the results from 3 iterations of a 5-fold cross validation. This gave us 10 different sets of parameters for each One vs. All classifier.

Next, we needed to convert the 10 sets of parameters into a single set of parameters. The standard deviation of the parameters indicated that their value did not differ greatly across splits, so we averaged the 10 sets of parameters to get a single set of parameters.

We compared the performance of using the individually optimized parameters vs. the average of the parameters and saw negligible differences in performance. This gave us confidence that the average parameters were a realistic estimate for how the classifier would perform on new data.

4.2 5-Stage Framework Performance

Our 5-stage framework contains multiple One vs. All classifiers. We evaluated the performance of each One vs. All classifier using an 80%/20% stratified split for our training and test data. We repeated this process 10 times and averaged the results.

We correctly classified 72.6% of the plays with a standard deviation of 3.15 and an f-score of .727. Table 4.1 illustrates the confusion matrix of our results.

		Prediction						
		E.P.U.	Elbow	Floppy	Horns	Invert	Delay	Other
True Label	E.P.U.	42	3	2	1	63	1	14
	Elbow	1	50	1	0	0	5	9
	Floppy	0	0	61	1	0	0	8
	Horns	1	0	0	15	1	0	4
	Invert	3	1	0	0	18	2	10
	Delay	1	5	0	1	0	81	4
	Other	24	18	4	5	8	6	123

Table 4.1: 5-stage framework confusion matrix

The majority of our misclassifications occur with *Other*. The 5-stage framework both predicts *Other* when the label is a play and predicts a play when the label is

Other. This shows that it is much easier to differentiate one play from another play than one play from *Other.* We also notice the large number of misclassifications with *Early Post Up* and *Invert.* These are the two shortest plays, which have few very movements. This is consistent with the results because these plays are most likely to be confused with random motion.

4.3 Multiple Classifiers vs. Standard Multi-Class

We compared the performance of our 5-stage framework with a standard multi-class approach. In the standard multi-class approach, we treat *Other* the same as every other label. We use a One vs. One approach that builds a classifier for each pair of plays, and the play with the most positive labels is the final prediction. This One vs. One approach performs reasonable well across a variety of data sets [7].

To standardize the two approaches, we use Fixed Length Segmentation for each play. A multi-class classifier contains multiple One vs. One classifiers. We optimized the parameters for these classifiers in the same way that we optimized the parameters in the final 5-stage framework, as described in section 6.1. Table 4.2 shows the accuracy, standard deviation, and f-score of these two approaches.

Approach	**Accuracy**	**Standard Deviation**	**F-score**
5-stage Framework	68.6%	2.96	.670
Standard Multi-class	62.6%	2.82	.602

Table 4.2: The number of labeled examples and run time for each play

The 5-stage framework outperforms the standard multi-class approach at a statistically significant level (p-value of 0.9635). As expected, the difference largely comes from its ability to classify *Other.*

4.4 Variable Length Segmentation vs. Fixed Length Segmentation

We compared the performance of using Variable Length Segmentation vs. Fixed Length Segmentation. We can evaluate the performance of each approach by comparing their accuracy, standard deviation, and f-score. Table 4.3 outlines each method's performance on each of the 6 plays.

Play	**Length**	**Instances**	**VLS F-score**	**FLS F-score**
Horns	3 sec.	21	.823	.691
Invert	2.5 sec.	35	.547	.350
Elbow	3 sec.	66	.572	.661
Early Post Up	2 sec.	67	.535	.483
Floppy	5 sec.	70	.901	.895
Delay	4 sec.	92	.860	.836

Table 4.3: Variable Length Segmentation (VLS) vs. Fixed Length Segmentation (FLS)

In all plays except *Elbow*, Variable Length Segmentation outperforms Fixed Length Segmentation. We see the largest difference in performance with *Invert* and *Horns*. These plays had the fewest number of instances in our dataset. This suggests that Variable Length Segmentation outperforms Fixed Length Segmentation when the sample sizes are small. In Fixed Length Segmentation, the play occurs in a small window during the time block. The feature vector looks very different if the play occurred in the beginning of the time block rather than the end of the time block. Since we have a limited training set, we do not have examples of the play at every point in the time block.

We also looked at the individual confusion matrices for each One vs. All classifier using Variable Length Segmentation. We averaged the results of 10 iterations.

		Prediction	
		Floppy	Not Floppy
True Label	Floppy	59	11
	Not Floppy	2	466
CCR: 97.6%, F-score: .901			

Table 4.4: Floppy vs. All, Variable Length Segmentation

		Prediction	
		Elbow	Not Elbow
True Label	Elbow	48	18
	Not Elbow	55	416
CCR: 86.5%, F-score: .572			

Table 4.5: Elbow vs. All, Variable Length Segmentation

		Prediction	
		Invert	Not Invert
True Label	Invert	20	14
	Not Invert	19	484
CCR: 93.8%, F-score: .547			

Table 4.6: Invert vs. All, Variable Length Segmentation

		Prediction	
		Delay	Not Delay
True Label	Delay	76	16
	Not Delay	10	436
CCR: 95.3%, F-score: .860			

Table 4.7: Delay vs. All, Variable Length Segmentation

		Prediction	
		E.P.U.	Not E.P.U.
True	E.P.U.	39	27
Label	Not E.P.U.	40	432
CCR: 87.5%, F-score: .535			

Table 4.8: Early Post Up (E.P.U.) vs. All, Variable Length Segmentation

		Prediction	
		Horns	Not Horns
True	Horns	17	4
Label	Not Horns	6	510
CCR: 98.6%, F-score: .823			

Table 4.9: Horns vs. All, Variable Length Segmentation

Longer plays, such as *Floppy* and *Delay*, are easiest to classify. These plays have many movements, so they are less likely to be confused with *Other*. On the other hand, *Invert*, which is comprised of a single screen and pass, had the lowest f-score at .547.

4.5 Ranked Predictions

An alternative approach to evaluating the 5-stage framework is to look at the accuracy if we are given more than one prediction. One advantage of the 5-stage framework is the ability to easily generate a list of predictions. Figure 4.10 shows that, given 2 predictions, we can reach an accuracy of 91.4%.

Number of Guesses	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Accuracy	72.6%	91.4%	97.2%	98.6%	99.0%	99.2%	100%

Table 4.10: Accuracy with number of predictions

Chapter 5

Bottom-Up: A New Language for Unsupervised Classification

Our 5-stage framework is based on supervised learning to classify plays. The 5-stage framework takes a top-down approach. We start with a list of play labels, and work backwards by building features that differentiate one play from another. This approach is limited by the fact that it requires human supplied labels for each play. Our supervised 5-stage framework is impractical if we want to generate the plays run by every team in the NBA. A better solution for play classification would be able to run without the time consuming task of manually labeling plays. In this section, we propose a bottom-up approach that can define plays without labels. This framework is broken down into three components: defining a language to represent the data, building classifiers to identify each element in the language, and developing a system to define plays from the language.

5.1 Defining a Language

For each possession, we start with the player and ball position data. Since we cannot define plays exclusively in terms of X and Y coordinates, we need a layer of abstraction to define plays in basketball-specific terminology. Previous work has proposed using acceleration as the basis for unsupervised play classification [10] We propose building a pre-defined language to describe plays.

Our language will need to describe locations on the court. In section 3.4.1, we described Degree Distance Positioning, which is a method to divide the court into blocks using a combination of distance and degrees from the basket. This positioning is advantageous because it can naturally be mapped to basketball terminology such as wing or corner. We break the court into many blocks, and basketball positions are defined by group of blocks in our graph. As seen in Figure 5-1, we can define the basketball locations *perimeter* and *blocks* by choosing a group of blocks from our base Degree Distance Positioning. The location *perimeter*, diagrammed in Figure 5-1B, is made from 8 blocks around the 3-point line. The location *blocks*, diagrammed in Figure 5-1C, is made from 6 blocks which encapsulate each block.

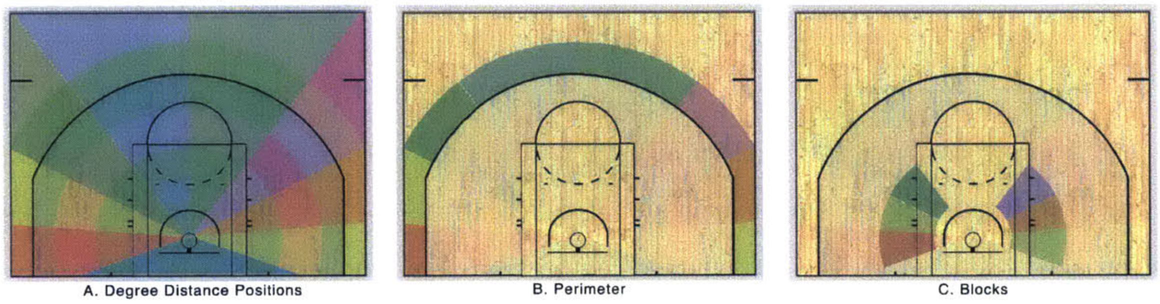


Figure 5-1: Degree Distance Positioning for *Perimeter* and *Blocks*

This method for describing locations on the court gives the flexibility to include basketball terminology that overlaps on the court. For example, the perimeter includes the corner, the wing, and the top of the key. This method also gives us an easy way to add more position definitions as the language progresses.

We will also need a methodology to build a domain of basketball activities. Since it is difficult to include every possible basketball action, it is important that our methodology is very flexible to handle new basketball actions. Each action can have either a single location, or a start and end location. For example, a corner backdoor cut would involve a player cutting from the corner area to the paint area.

To build our list of classifiers, we start by labeling the most commonly seen basketball actions, such as *on-ball screens*. Although we start with general case of an on-ball screen, we create separate terminology for the variants of an on-ball screen. In this framework, a player can perform multiple actions simultaneously. For instance, in the case of an on-ball screen, a player can perform both an *on-ball back screen* and an *on-ball rescreen.*

Eventually, we will need to create new terminology not currently present in the current domain of basketball vocabulary. In this framework, we can create new terms to describe complex basketball actions. Over time, we will build up a powerful language database that describes many of the possible basketball actions.

5.2 From Language to Classifiers

We need a way to recognize when each basketball action occurs in our language. There are two approaches, each of which is better suited for different actions: rule based classification and machine learning based classification. In simple actions such as cuts from point A to point B, a rule-based classification is a simple and effective method to identify the cut. For more complicated actions, such as types of screens, machine learning will be more effective because it can handle many of the edge cases that are hard to define with rules.

In our 5-stage supervised learning, we began the process of building classifiers for basketball actions. We implemented various rule-based systems to identify basketball actions. Our closeness metric looks at times when two players get close to one another. This helps us find on-ball and off-ball screens. We also created six actions for passes using rule-based classification. We can build off of the classifiers in our 5-stage framework to build classifiers for our language.

5.3 Defining a Play

The ultimate goal of this framework is to classify plays, but the term 'play' can be ambiguous. If we gave two coaches a sufficiently large set of possessions and asked them to diagram what plays were run, the coaches would almost certainly produce different diagrams and differ on the cumulative number of plays run. This occurs because it is unclear when two possessions are sufficiently different to consider them separate plays.

In our framework, plays can be defined dynamically as a sequence of actions. A play can be fully described by it's actions, which we will define as an Action Series. For example, the play *Floppy* is defined by the following series of actions: (1) Point guard dribbles from half court to the top of the key, (2) Two post players sets a screen at the block, (3) Two wing players cross under the basket, (4) Each wing players uses a block screen, and (5) The point guard passes to one of the two wing players. Plays can also have multiple definitions. In the case of Floppy, even if the point guard drives instead of passing to a wing player, we might still say the play is *Floppy*. As seen in Figure 5-2, we can add an additional action (5) where the point guard drives to the basket.

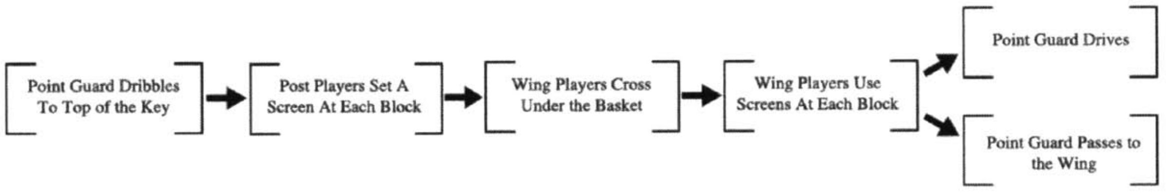

Point Guard Dribbles To Top of the Key
Post Players Set A Screen At Each Block
Wing Players Cross Under the Basket
Wing Players Use Screens At Each Block
Point Guard Drives
Point Guard Passes to the Wing

Figure 5-2: A Series Action for *Floppy*

We can also use this approach to perform unsupervised classification on a large set of possessions. For example, we can classify a large set of possessions for a given team. First, we watch a game or two and create a set of Action Series that define each play. We then run each possession through every Action Series to determine if the possession contains that Action Series. With this method, we can create an Action Series from as small as a single instance of a play. This approach might be superior to our 5-stage supervised classification framework, where we required a large number of labeled possessions to perform the classification.

5.4 Applications and Conclusions

The bottom-up unsupervised approach has a number of potential advantages over the top-down 5-stage framework. There is a significant amount of analysis that occurs within a play. For example, a team may not care if the play is *Elbow* or *Horns* (both pick and roll plays). They may only be interested in all possessions where there is a pick and roll and the picker rolls to the basket. The bottom-up approach gives the flexibility to find all instances of this series of actions, which is only a piece of a defined play.

This approach also gives a quicker way to classify plays for a given team. In our experiments with the 5-stage framework, we used 537 labeled examples, and our results would have improved if we had more training examples. With the bottom-up approach, we could perform the same classification with a much smaller number of examples. Since the number of labeled instances is drastically reduced, it becomes feasible to create classifiers for opponents.

The bottom-up approach provides a general method to build plays from the ground up. Creating a language to describe the position data in more abstract terms also has benefits beyond play classification. However, implementation is significantly more difficult than the top-down approach. We are tasked with developing a large set of basketball actions and building classifiers to recognize these actions, which is an enormous task.

Chapter 6

Summary and Conclusion

We summarize our work in section 6.1, review our major concepts and frameworks in section 6.2, and discuss our future work in section 6.3.

6.1 Summary

In this thesis, we explored the potential to automatically classify NBA plays. An NBA franchise hand labeled possessions from 49 games during the 2012-13 season. We looked at 7 play labels: *Elbow*, *Floppy*, *Horns*, *Early Post Up*, *Invert*, *Delay* and *Other*.

Classifying these plays had four main challenges. First, plays are defined by a small subset of the movements during a possession, so we needed a way to identify the important piece of the possession. Second, each of our labeled plays runs for a different amount of time. Third, we had the "play" *Other*, which were possessions for which we didn't have a label and possessions where no play was run. *Other* generates feature vectors that look different from one another. Fourth, we had a small number of labeled plays for training our classifiers.

We built a 5-stage framework that takes a top-down approach to classify plays. We started with labeled plays and used supervised learning to build the classifier. This framework is different from a traditional multiclass classifier because it made use of the variable lengths of time that a play can run and it handled the case of plays that fall into *Other*. The 5-stage framework had an accuracy of 72.6% with a standard deviation of 3.15.

We then proposed an unsupervised classification framework that is based on a bottom-up approach. This framework starts by building a language to describe the domain of basketball activities. The language contains basketball activities such as cuts, screens, post ups, backdoors, etc. Each basketball activity has an associated classifier that is used to recognize when the basketball activity occurs. The language is then used a basis for a rule based system to identify plays.

6.2 Future Work

In the near future, we hope to improve our top-down 5-stage framework. We can improve our framework by developing new ways to capture events, incorporating play options into our classification, and using machine learning to build our features. We can also begin to implement the bottom-up approach.

6.2.1 Play Options

A play can have multiple options. For instance, in the play Elbow, the screener can choose to roll to the basket or pop out for a three pointer. We can begin to identify options within plays using a hierarchical classification framework. In the first stage, we can identify the play itself, and in the second stage, we can identify what option of the play was run.

6.2.2 Machine Learning Features

Our 5-stage framework uses a rule-based system to generate all of our feature vectors, but certain basketball actions can be identified more accurately with machine learning classifiers. As seen in the paper, Automatically Recognizing On-Ball Screens [11], machine-learning classifiers were better suited to identify on-ball screens than a simple rule based system. In future work, we will compare the performance of rule based vs.

machine learning based classifiers for more complex basketball actions.

6.2.3 Implementing Bottom-Up Classification

The bottom up approach introduces the possibility of unsupervised play classification. In future work, we will begin to solve this problem by creating a language to describe basketball actions. We will also concurrently build classifiers to identify when these basketball actions occur. Finally, we will develop a system to convert basketball actions into plays.

6.3 Conclusion

Automatically classifying plays would provide tremendous benefits to NBA franchises, and this thesis shows that play classification from position data is possible. We demonstrated that our 5-stage framework is an effective approach to classifying plays. With a larger dataset and more feature engineering, we believe a classifier can be built to reliably predict the correct play. We are also hopeful that a bottom-up approach will allow more generalized play classification.

Bibliography

[1] Ben A. Aisen. A comparison of multiclass svm methods, 2006.

[2] Christopher Bishop. *Pattern Recognition and Machine Learning.* Springer, 2006.

[3] Alex Conover. Sportvu: An inside look. *Stats In the News*, 2012.

[4] S. Dudani. The distance-weighted k-nearest-neighbor rule. *IEEE Transactions on Systems, Man and Cybernetics*, (6):325–327, 1976.

[5] Srikanth Nori Gautam Kowshik, Yu-Han Chang, and Rajiv Maheswaran. Visualization of event-based motion-tracking sports data. Technical report, University of Southern California, 2012.

[6] Kirk Goldsberry and Eric Weiss. The dwight effect: A new ensemble of interior defense analytics for the nba. *MIT Sloan Sports Analytics Conference*, 2013.

[7] Chih-Wei Hsu. A comparison of methods for multiclass support vector machines. Technical report, National Taiwan University, 2002.

[8] Mitchell Kates and John Guttag. www.courtlytics.com, 2014.

[9] Christopher D. Manning. *Introduction to Information Retrieval.* Cambridge University Press, 2008.

[10] Philip Maymin. Acceleration in the nba: Towards an algorithmic taxonomy of basketball plays. Technical report, NYU-Polytechnic Institute, 2012.

[11] Armand McQueen. *Automatically Recognizing On-Ball Screens.* Massachusetts Institute of Technology, 2013.

[12] USA TODAY Sports Network. Will analytics overtake the old school?, mar 2014.

[13] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[14] J. Platt. Probabilistic outputs for support vector machines and comparison to regularized likelihood methods. *Advances in Large Margin Classifiers*, 2000.

[15] STATS. Sportvu optical player tracking system.

[16] S. Chris Colbert Stfan van der Walt and Gal Varoquaux. The numpy array: A structure for efficient numerical computation. *Computing in Science & Engineering*, 13(22-30), 2011.

[17] C.-J. Lin T.-F. Wu and R. C. Weng. A note on platt's probabilistic outputs for support vector machines. Technical report, National Taiwan University, 2004.