8TH ANNUAL
MIT SLOAN
SPORTS ANALYTICS CONFERENCE
FEBRUARY 28 - MARCH 1, 2014 HYNES CONVENTION CENTER PRESENTED BY ESPN

Automatically Recognizing On-Ball Screens

Armand McQueen, Jenna Wiens, and John Guttag
Massachusetts Institute of Technology
Cambridge, MA, USA, 02139
Email: amcqueen@mit.edu, jwiens@mit.edu, guttag@mit.edu

Abstract

The pick and roll is a powerful tool; as former coach Stan Van Gundy once said of his Magic team, "[The pick and roll is] what we're going to be in when the game's on the line. [...] I don't care how good you are, you can't take away everything" [1]. In today's perimeter oriented NBA, the pick and roll is more important than ever before. The player tracking data that is now being collected across all arenas in the NBA holds out the promise of deepening our understanding of offensive strategies. In this paper we approach part of that problem by introducing a pattern recognition framework for identifying on-ball screens. We use a machine learning classifier on top of a rule-based algorithm to recognize on-ball screens. Tested on 21 quarters from 14 NBA games from last season our algorithm achieved a sensitivity of 82% and positive predictive value of 80%

1 Introduction

The pick and roll is a fundamental part of offense in the NBA; thus, much effort is invested in determining the most effective way to run it. Until now, this required a significant amount of manual labor for video/play review (or just *a lot* of coaching experience). With the player tracking data provided by the STATS SportVu system and machine learning techniques, we can automate much of this effort. In this paper, we describe the development of a system for automatically recognizing on-ball screens. Our results suggest that such a system could be useful for identifying similarities between teams in the NBA, and how they run their offenses. Moreover, the techniques and framework described here can be applied to other pattern recognition tasks, such as identifying offensive plays or defensive schemes.

2 Data Segmentation

For our analysis we use player-tracking data collected by the STATS SportVu system from the 2012-2013 season. These data form a Position table and an Event table:

Position table - contains the {x,y} position of every player on the court and the {x,y,z} coordinates of the ball, at 25 frames a second.

Event table - contains play-by-play data, i.e., a chronological list of events that occur in the game (e.g. fouls, passes, dribbles, shots), and when each event occurred.

```
IF [
        there is a player who is dribbling the ball (i.e., the ball-handler)
    AND there is an offensive player within 10 feet of the ball-handler (i.e.,
        the screener)
    AND the screener is not in or near the paint (see in-set figure for
        definition of "near")
    AND the screener is no more than 2 ft. further from the basket than the
        ball-handler is
    AND there is a defensive player <= 12 ft. from the ball-handler (i.e., the
        on-ball defender)
    AND the basketball is not in the paint
] for >=13 consecutive frames
AND the ball-handler is the same for every frame
THEN
  identify as action
```

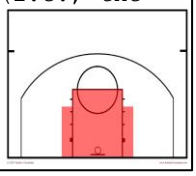


Figure 1: A rule-based algorithm to segment the data into actions

To identify on-ball screens, we begin by segmenting the data from the Position table into short periods of time, called *actions*. We do this using a rule-based algorithm that examines each frame against a set of criteria described in Figure 1. We developed these criteria to include any period of time when an on-ball screen was possible and to eliminate situations when an on-ball screen could not be occurring (e.g., when there is no one dribbling or the ball-handler is in isolation). An action represents a continuous period of time during which every frame fulfills the criteria. In our data, the average length of an action was just over 1.5 seconds long.

Segmenting the data into actions serves two purposes: (1) it reduces the size of the data by eliminating irrelevant data, and (2) it creates a set of discrete examples that can be annotated. This first step alone significantly reduces the amount of labor required to identify on-ball screens. When we applied the criteria in Figure 1 to 21 randomly selected quarters from 14 games (see Appendix), we achieved a sensitivity of approximately 98% and a positive predictive value (PPV) of approximately 52%. I.e., approximately 98% of all the on-ball screens were part of actions and approximately 52% of all actions contained an on-ball screen. Of course, a PPV of 52% is not nearly good enough.

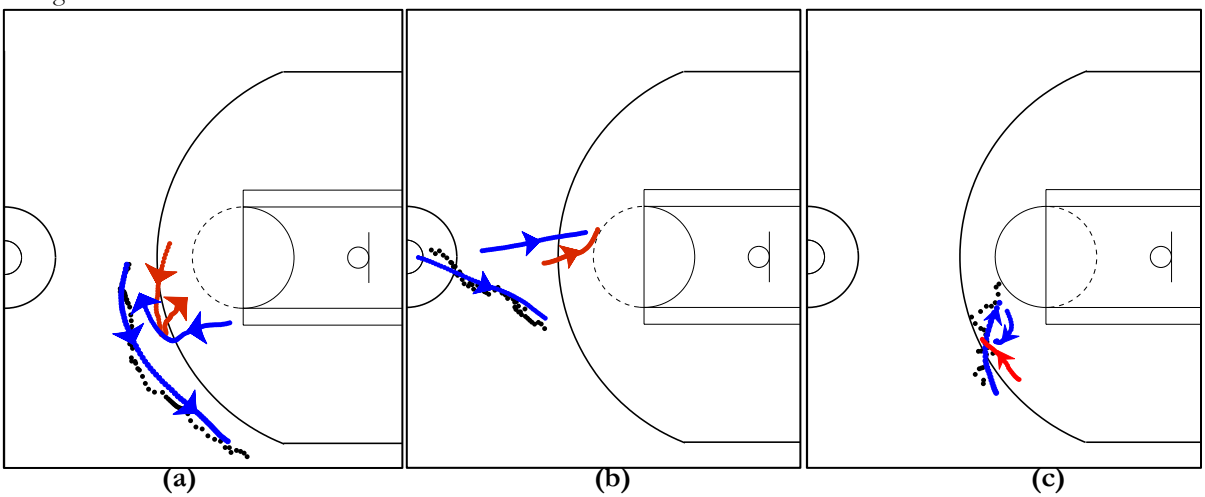


Figure 2: These court mockups illustrate the three major categories of actions we identified.

- **(a) Type 1, a classic on-ball screen. The screener starts close to the paint and moves away from the basket towards the ball-handler to set a screen.**
- **(b) Type 2, establishing half-court positions. The paths of ball-handler and screener come close as they move to their positions in the half-court offense. There is no screen in this type of action.**
- **(c) Type 3, a perimeter movement. The ball-handler moves along the perimeter, while another offensive player moves in the opposite direction along the perimeter. That second offensive player may or may not set a screen.**

When we inspected the actions by eye, we identified three major categories of actions (see Figure 2). Based on these observations, we hypothesized that the large number of Type 2 and Type 3 non-screen actions causes the poor PPV. In the next section, we describe the development of a machine learning classifier to automatically discriminate the on-ball screen actions from the non-screen actions.

3 Machine Learning Classifier

Learning and testing a supervised classifier can be split into several major steps as shown in Figure 3. In the sections that follow we describe each step in building our classifier in detail.

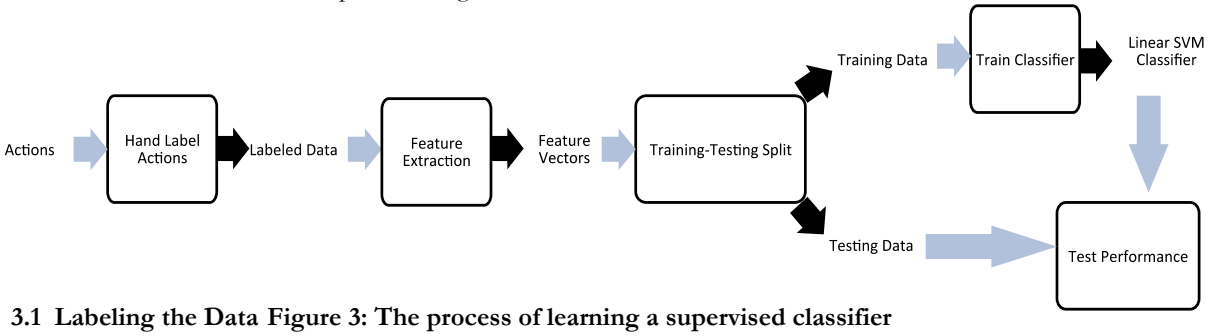


Figure 3: The process of learning a supervised classifier

3.1 Labeling the Data

We assign each action a value of either +1 or -1, indicating whether the action is an on-ball screen or not. As mentioned above, we chose 21 randomly selected quarters from 14 games (Appendix). Ten teams were represented in the 14 games, of which nine teams had a non-trivial (greater than 10) number of screens. To label the data, we watched the quarter and recorded the time each on-ball screen occurred. If we had any doubt that an on-ball screen occurred, we did not include it in our analysis. We also did not consider handoffs to be on-ball screens. If an action contained an on-ball screen annotation between the start and end times, it was labeled as positive, otherwise it was labeled as negative.

3.2 Feature Extraction

For each action, we extract 30 continuous features. We begin by determining the ball-handler, the screener and the on-ball defender using the Event and Position tables.

Ball-Handler: We examine the Event table and if the most recent event was a dribble, the player that performed the dribble is identified as the ball-handler. Otherwise there is no ball-handler and the preprocessing algorithm rejects the frame.

Screener: We determine the player that sets the screen by weighing three factors: how long the player is stationary, whether he approaches the ball-handler early in the action and how close the player gets to the ball-handler.

On-Ball Defender: The on-ball defender is identified as the defensive player who is closest to the ball-handler during the first 11 frames of the action. To account for the possibility of the on-ball defender being caught on the screen, we weight the distance between the player and the ball-handler more heavily in the earlier frames of the action.

Next, we quantify the pairwise interactions between the ball-handler, the screener, the on-ball defender and the basket. Let p_t for $t = 1 \ldots n$ represent the {x,y} position of a player p during an action. We measure the pairwise distance between two players, player a and player b, at time t, as $d_{t,a,b} = \|a_t - b_t\|$, and between player a, and the basket as $d_{t,a,basket} = \|a_t - p_{basket}\|$

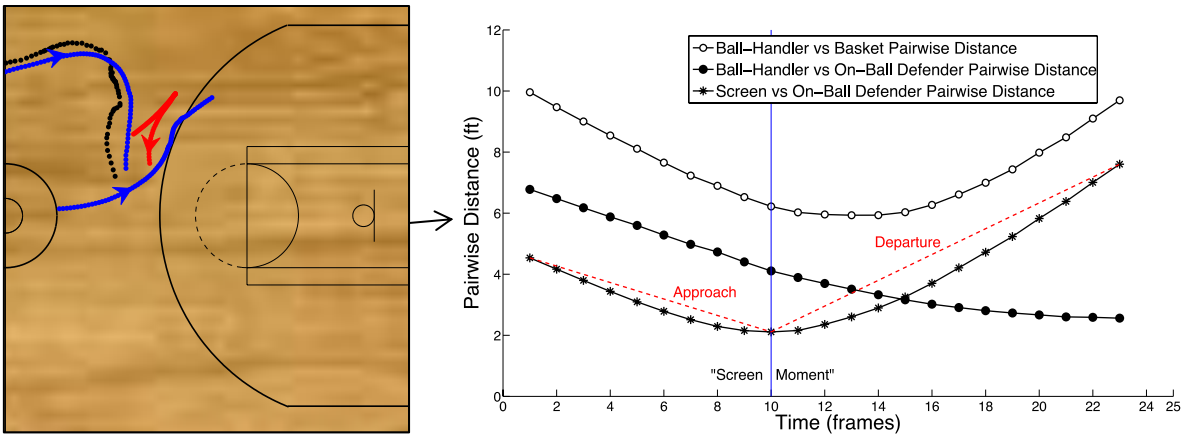


Figure 4: The court mock-up on the left illustrates actual position data for an action between Portland and OKC. This data is transformed into the pairwise distance time series plotted on the right in which we identify the "screen moment". This index event allows us to extract relevant features pertaining to *before* and *after* the screen. For clarity, we only show 3 of the 6 pairwise distance time series.

We define the moment of the screen, s, as the time when the pairwise distance between the ball-handler p_b and the screener p_s is at a minimum.

$$s = \underset{t}{\operatorname{argmin}}\ d_{t,p_b,p_s}$$

This *screen moment*, divides the action into two halves: the approach $t \leq s$ and the departure $t > s$. Given the pairwise distance between player a and player b, and s we summarize each time series using 5 continuous values:

Table 1: The five continuous features extracted for each action

1	2	3	4	5
$\min_t d_{t,a,b}$	$\frac{(d_{s,a,b} - d_{1,a,b})}{s}$	$\frac{1}{s}\sum_{t=1}^{s} d_{t,a,b}$	$\frac{(d_{n,a,b} - d_{s,a,b})}{(n-s)}$	$\frac{1}{(n-s)}\sum_{t=s}^{n} d_{t,a,b}$

We repeat the above to characterize the distance between each player and the basket, substituting p_{basket} for player b. Extracting the features listed above for each set of pairwise distance results in 30 continuous features (five for each pair). Next, we discretize each continuous feature into five binary features based on quintiles. This results in a 150-dimensional feature vector for every action.

3.3 Splitting Data into Training and Testing Sets

In order for each team to factor equally into our analysis, we needed to create a dataset that included the same number of data points from each team. We randomly selected 27 positive and 27 negative data points from each team and combined that into a global dataset. We then partitioned that dataset into a training set of 252 points (14 positive and 14 negative randomly chosen from each team) and a test set of 234 points (13 positive and 13 negative from each team).

3.4 Training the Classifier

Our dataset is defined as follows:

$$\mathcal{D} = \{(\mathbf{x}_i, y_i) | \, \mathbf{x}_i \in \mathcal{X}, y_i \in \{-1, 1\}\}_{i=1}^{n}$$

where n represents the number of labeled examples in the training data, and $\mathcal{X}$ represents a 150-dimensional feature space in which each feature vector, $\mathbf{x}_i$, lies. Using support vector machine (SVMs) we learn a linear classifier $f(\mathbf{x}) : \mathcal{X} \to \mathbb{R}$ where

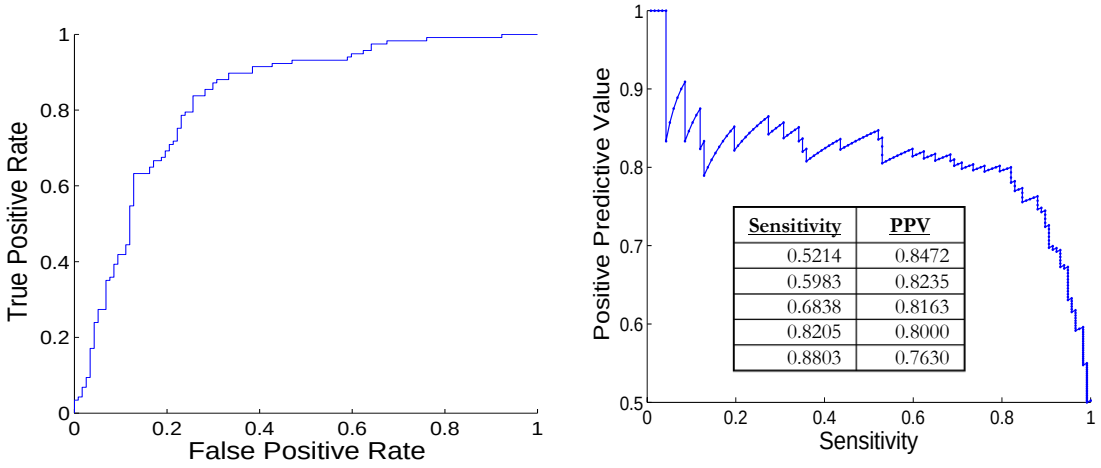
$$f(x) = \mathbf{w}^T\mathbf{x} - b$$

$\mathbf{w}$ is a weight vector $\in \mathbb{R}^{150}$ and b is a scalar. The optimal settings of $\mathbf{w}$ and b are learned using LIBSVM [3]. When learning the SVM we select the cost-parameter using 5-fold cross validation on the training set[1]. Given a new point x, it is labeled according to $sign(f(x))$.

3.5 Performance

We applied the classifier to our test set and generated predictions according to $\hat{y} = sign(f(x) - d)$ for each example. To measure performance on the test set, we considered the true positive rate and the false positive rate at each value of the decision boundary, d. This creates the Receiver Operator Characteristic (ROC) curve (Figure 5a). We used the area under the ROC curve (AUROC)[2] as a measure of performance. We repeated the entire process, shown in Figure 3, 200 times for different random splits of data. The mean AUROC across all splits was 0.8309. As shown in the Figure 5b, there exists a trade-off between sensitivity and positive predictive value. If we set the threshold so that we have 80% confidence that a positive label is correct (80% PPV), we can identify 82% of the positively labeled actions (82% sensitivity).

[1] For an in-depth review of SVMs please refer to [2].
[2] The AUROC is a summary statistic representing the probability that, given two random examples of opposing labels, the classifier will rank the positive example higher than the negative example. An AUROC of 0.5 is no better than random and an AUROC of 1.0 is perfect.

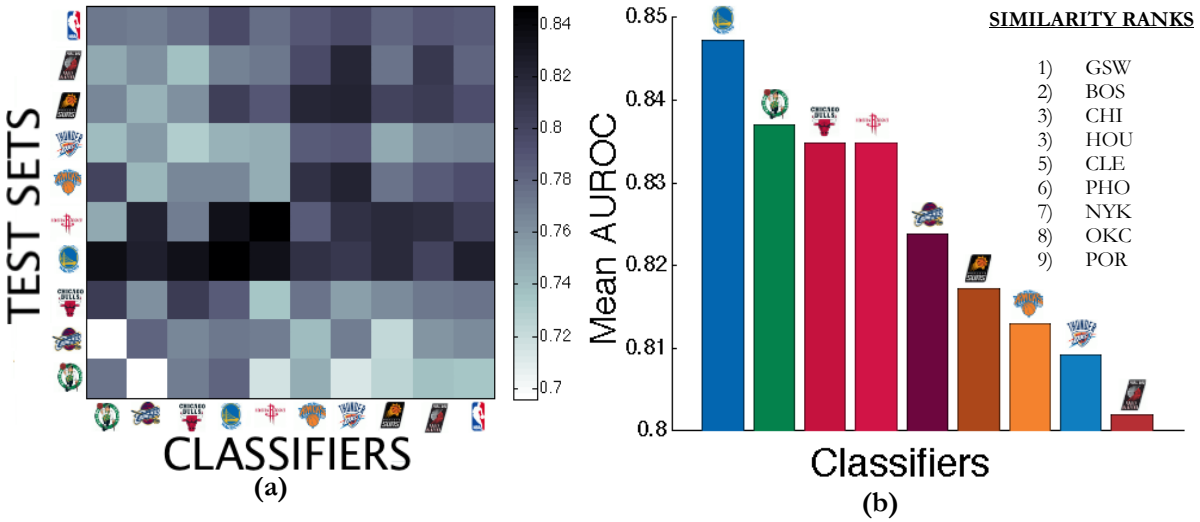


(a) (b)

Figure 5: (a) The ROC curve for the global classifier with the median AUROC
(b) The Recall-Precision tradeoff (right) for that global classifier with some Sensitivity-PPV value pairs.

4 Improving The Classifier By Identifying Similar Offenses

Different teams run different offenses and this limits the global classifier's ability to identify on-ball screens. We hypothesized that we could achieve even better performance by learning classifiers trained on groups of teams that run similar offenses. To test this hypothesis, we first estimated the similarity between pairs of teams. If a classifier trained on data from Team A performs well when applied to data from Team B, we consider Team A to be similar to Team B (in terms of on-ball screens). (Note: this notion of similarity is not symmetric.) In order to control for the number of training examples (e.g., for some teams we had more data than for others), each team-specific classifier was trained on the same number of randomly selected examples. We applied each classifier to each team's test data. The resulting AUROCs of each experiment are shown in Figure 6a (Full results in Appendix). Figure 6b gives the similarity of each team with respect to the Golden State Warriors.



(a) (b)

Figure 6: (a) The performance (AUROC) when testing all team-specific classifiers and the global classifier on each team's test set.
(b) Creating similarity ranks for GSW by sorting single-team classifier performance on the GSW test set.

4.1 Augmented Team-Specific Classifiers

To test our hypothesis, we carried out an experiment comparing the value of **more** training data from a **variety** of teams versus the value of using **less** training data from a subset of **similar** teams. We started by creating a 9-team group classifier trained on 252 actions, 14 positive and 14 negative from all nine teams. Then, for each team, we created an 8-team group classifier where we trained a classifier on 224 actions, 28 from each of the 8 most similar teams. We continued to reduce the size of the group classifier using the similarity rankings (as defined earlier) until we were left with a classifier trained on the single most similar team.

As shown in Figure 7, the size of the optimal classifier varied between teams. In almost every case the optimal classifier was trained on 56% or less of the available training data. This is somewhat surprising, since in general classifier performance improves with more data. These results suggest that inter-team differences in the way on-ball screens are run are significant, and simply pooling the data together to learn one classifier can be detrimental to performance. By intelligently selecting a subset of teams to include in the training set, we can drastically reduce the amount of labeled data required without sacrificing team-specific performance.

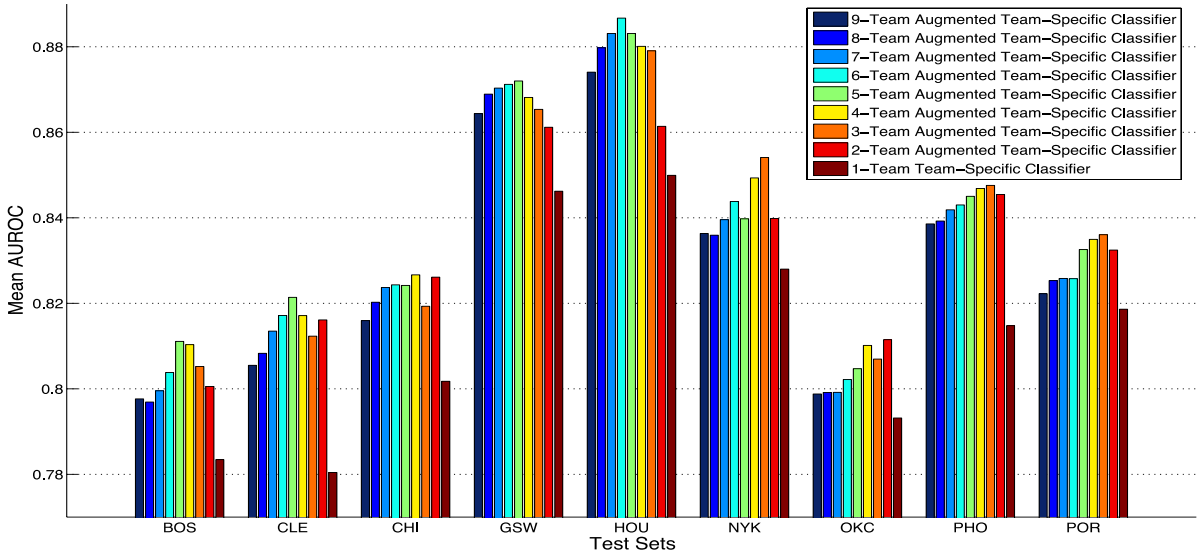


Figure 7: For each team's test set, this graph shows how the performance of the Augmented Team-Specific classifier changes as we repeatedly remove the least similar team from the training set. For each team, the training size of the Augmented Team-Specific Classifier decreases from left to right.

It appears that on-ball screens run by Golden State and Houston are both much easier to classify than those run by any of the other teams. We looked over film from the games we labeled and found that one possible explanation is that the Warriors and the Rockets place a strong emphasis on spacing and so, unlike many other teams, they rarely have their perimeter players approach each other. Therefore they have very few Type 3 actions. In contrast, Oklahoma City, a team for which it is more difficult to identify on-ball screens, focuses on forcing switches by running Westbrook-Durant brush screens on the perimeter. This means many of the Thunder's actions are Type 3. The classifier is much better at differentiating Type 1, which are always screens, from Type 2 actions, which are never screens, than it is at differentiating Type 3 actions that are screens from Type 3 actions that are not screens. A secondary classifier focusing exclusively on Type 3 actions might be able to improve the performance of our system.

6 Limitations, Future Work, and Conclusions

Our ability to evaluate our method was limited by the fact that we did our own labeling of the data. Since our classifiers are learned from the data, the problem is not as severe as it would have been if we were evaluating a hand-coded classifier. However, going forward we would like to conduct a study in which an outside expert supplies the labels.

As part of this work we built team-specific classifiers. However, teams change personnel throughout the course of a game and over the course of a season. To the extent that offenses are dominated by coaching philosophies, our approach makes sense. Going forward, it would be interesting to build personnel-specific classifiers. Unfortunately, at the time of this writing, we did not have enough data to do this.

Despite these limitations, we believe that the line of work reported here shows great promise. Despite the limited number of examples, we were able to use machine learning to derive accurate classifiers. Furthermore, we showed that learning team-specific classifiers from data rather than hand coding a global classifier sheds light upon how certain teams run their pick-and-roll offenses similarly. We believe that we can use the same approach (though with different features and segmentation criteria) to recognize both other offenses and various defensive schemes.

7 References

[1] Abrams, Jonathon. "The Pick-and-Roll Is the N.B.A.'s Old Reliable." *The New York Times*, 19 Nov. 2009. Web. 6 Jan. 2014.

[2] Cortes, Corinna, and Vladimir N. Vapnik. "Support Vector Networks." *Machine Learning* 20 (1995), Print.

[3] Chang, Chih-Chung, and Chih-Jen Lin. "LIBSVM: A Library for Support Vector Machines." *ACM Transaction on Intelligent Systems and Technology* (2011). Print.

8 Appendix

Labeled Games

Away Team	Home Team	Date	Quarter(s)
POR	OKC	2-Nov-12	4
CHI	PHO	14-Nov-12	1
MIN	GSW	24-Nov-12	1
CLE	BOS	19-Dec-12	1, 2, 3 & 4
NYK	OKC	7-Apr-13	1, 2, 3 & 4
HOU	BOS	11-Jan-13	2
POR	HOU	8-Feb-13	1
CHI	HOU	21-Nov-12	3
POR	PHO	21-Nov-12	4
NYK	GSW	11-Mar-13	3
CHI	GSW	15-Mar-13	1
PHO	GSW	20-Feb-13	2
NYK	HOU	23-Nov-12	4
POR	HOU	8-Feb-13	3
CHI	HOU	21-Nov-12	1

Labeled Data Size By Team

LABELED DATA		
TEAM	# of positive actions	# of negative actions
BOS	50	43
CLE	37	51
CHI	71	54
GSW	35	42
HOU	43	81
NYK	62	43
OKC	39	33
PHO	31	34
POR	50	28
GLOBAL	418	409

Performance (AUROC) of Team-Specific and Global Classifiers on Each Team

		CLASSIFIERS									
		BOS	CLE	CHI	GSW	HOU	NYK	OKC	PHO	POR	GLOBAL
TEST SETS	BOS	0.7744	0.6958	0.7706	0.7820	0.7169	0.7471	0.7128	0.7246	0.7380	0.7347
	CLE	0.6978	0.7810	0.7667	0.7718	0.7704	0.7389	0.7702	0.7236	0.7593	0.7631
	CHI	0.8066	0.7601	0.8062	0.7857	0.7349	0.7746	0.7532	0.7641	0.7718	0.7761
	GSW	0.8370	0.8238	0.8348	0.8472	0.8348	0.8129	0.8092	0.8172	0.8019	0.8250
	HOU	0.7487	0.8233	0.7703	0.8348	0.8467	0.7879	0.8138	0.8165	0.8150	0.8039
	NYK	0.7999	0.7408	0.7663	0.7654	0.7469	0.8122	0.8232	0.7741	0.7863	0.7928
	OKC	0.7385	0.7556	0.7303	0.7441	0.7469	0.7871	0.7890	0.7485	0.7666	0.7678
	PHO	0.7662	0.7447	0.7603	0.8024	0.7901	0.8191	0.8222	0.8012	0.8056	0.7932
	POR	0.7488	0.7602	0.7379	0.7671	0.7716	0.7968	0.8203	0.7761	0.8091	0.7830
	GLOBAL	0.7717	0.7714	0.7772	0.7966	0.7769	0.7898	0.7959	0.7814	0.7888	0.7859