Improving Offensive Performance Through Opponent Modeling

Kennard Laviers
School of EECS
U. of Central Florida
Orlando, FL
klaviers@eecs.ucf.edu

Gita Sukthankar
School of EECS
U. of Central Florida
Orlando, FL
gitars@eecs.ucf.edu

Matthew Molineaux
Knexus Research
Springfield, VA
matthew.molineaux@knexusresearch.com

David W. Aha
NCARAI
Naval Research Lab
Washington, DC
david.aha@nrl.navy.mil

Abstract

Although in theory opponent modeling can be useful in any adversarial domain, in practice it is both difficult to do accurately and to use effectively to improve game play. In this paper, we present an approach for online opponent modeling and illustrate how it can be used to improve offensive performance in the Rush 2008 football game. In football, team behaviors have an observable spatio-temporal structure, defined by the relative physical positions of team members over time; we demonstrate that this structure can be exploited to recognize football plays at a very early stage of the play using a supervised learning method. Based on the teams' play history, our system evaluates the competitive advantage of executing a *play switch* based on the potential of other plays to increase the yardage gained and the similarity of the candidate plays to the current play. In this paper, we investigate two types of play switches: 1) whole team and 2) subgroup. Both types of play switches improve offensive performance, but modifying the behavior of only a key subgroup of offensive players yields greater improvements in yardage gained.

1. Introduction

By accessing the play history of your opponent, it is possible to glean critical insights about future plays. This was recently demonstrated at a soccer match by an innovative, well-prepared goalkeeper who used his iPod to review a video play history of the player taking a penalty kick; identifying the player's tendency to kick to the left allowed the goalkeeper to successfully block the shot (Bennett 2009). Although play history can be a useful source of information, it is difficult to utilize effectively in a situation with a large number of multi-agent interactions. Opponent modeling can be divided into three categories: 1) online tracking, 2) online strategy recognition and 3) off-line review. In online tracking, immediate future actions of individual players (passes, feints) are predicted, whereas in online strategy recognition, the observer attempts to recognize the high-level strategy used by the entire team. In offline review, general strengths, weaknesses, and tendencies are identified in an offline setting and used as part of the training/learning regimen.

This paper addresses the problem of online strategy recognition in adversarial team games. In physical domains (military or athletic), team behaviors often have an observable spatio-temporal structure, defined by the relative physical positions of team members. This structure can be exploited to perform behavior recognition on traces of agent activity over time. This paper describes a method for recognizing defensive plays from spatio-temporal traces of player movement in the Rush 2008 football game (Figure 1) and using this information to improve offensive play. To succeed at American football, a team must be able to successfully execute closely-coordinated physical behavior. To achieve this tight physical coordination, teams rely upon a pre-existing playbook of offensive maneuvers to move the ball down the field and defensive strategies to counter the opposing team's attempts to make yardage gains. Rush 2008 simulates a modified version of American football and was developed from the open source Rush 2005 game, which is similar in spirit to Tecmo Bowl and NFL Blitz (Rush 2005).

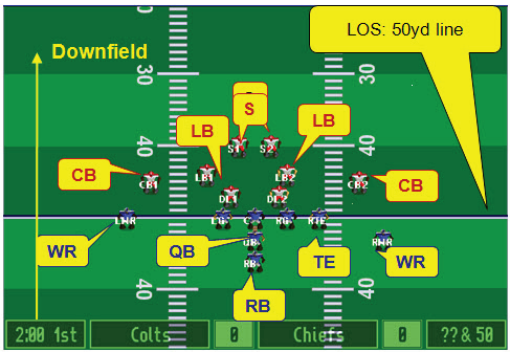


Figure 1: Rush 2008 football simulator the blue players are the offense (Pro formation) and the red the defense (2222 formation).

Although there have been other studies examining the problem of recognizing completed football plays, we present results on recognizing football plays online at an early stage of play, and demonstrate a mechanism for exploiting this knowledge to improve a team's offense. Our system evaluates the competitive advantage of executing a *play switch* based on the potential of other plays to improve the yardage gained and the similarity of the candidate plays to the current play. Our play switch selection mechanism outperforms

both the built-in offense and a greedy yardage-based switching strategy. Calculating the relative similarity of the current play compared to the proposed play is shown to be a necessary step to reduce confusion on the field and effectively boost performance. Additionally we investigated the utility of limiting the play switch to a subgroup of players; by only modifying the actions of small subgroup of key players, we can improve on the total team switch.

2. Related Work

Previous work on team behavior recognition has been evaluated within the following domains: American football (Intille and Bobick 1999), basketball (Jug *et al.* 2003), Robocup soccer simulations (Riley and Veloso 2002; Kuhlmann *et al.* 2006), and tabletop games (Sukthankar and Sycara 2007). In Robocup, most of the research on team intent recognition is geared towards designing agents for the coach competition. Techniques have been developed to extract specific information, such as home areas (Riley *et al.* 2002), opponent positions during set-plays (Riley and Veloso 2002), and adversarial models (Kuhlmann *et al.* 2006), from logs of Robocup simulation league games. This information can be utilized by the coach agent to improve the team's scoring performance. For instance, information about opponent agent home areas can be used as triggers for coaching advice and for doing "formation-based marking", in which different team members are assigned to track members of the opposing team. However, the focus of the coaching agents is to improve performance of teams in future games; our system immediately takes action on the recognized play to evaluate possible play switches and perform play adaptations.

Rush 2008 was developed as a platform for evaluating game-playing agents and has been used to study the problem of learning strategies by observation (Li *et al.* 2009). Intention recognition has been used within Rush 2008 as part of a reinforcement learning method for controlling a single quarterback agent (Molineaux *et al.* 2009); in this paper, we present an approach for optimizing policies across multiple agents.

3. Rush Football

Football is a contest of two teams played on a rectangular field bordered on lengthwise sides by an end zone. Unlike American football, Rush teams only have 8 players on the field at a time out of a roster of 18 players, and the field is 100×63 yards. The game's objective is to out-score the opponent, where the offense (i.e., the team with possession of the ball), attempts to advance the ball from the line of scrimmage into their opponent's end zone. In a full game, the offensive team has four attempts to get a *first down* by moving the ball 10 yards down the field. If the ball is intercepted or fumbled and claimed by the defense, ball possession transfers to the defensive team. The Rush 2008 simulator only runs one play with the line of scrimmage set to the center of the field. Stochasticity exists in catching (i.e., whether a catch is successful), fumbling, tackles, distance/location of a thrown ball, and the selection of who to throw to if no receiver is "open" when the QB is forced to throw the ball. To simplify the dimensionality of the game we do not take into account differences between downs or use field goals.

The offensive lineup contains the following positions:

Quarterback (QB): given the ball at the start of each play. The QB hands the ball off or passes it to another player.

Running back (RB): begins in the backfield, behind the line of scrimmage where the ball is placed, with the quarterback and fullback.

Full back (FB): serves largely the same function as the RB.

Wide receiver (WR): primary receiver for pass plays.

Tight end (TE): begins on the line of scrimmage immediately to the outside of the offensive lineman and can receive passes.

Offensive linemen (OL): begin on the line of scrimmage and are primarily responsible for preventing the defense from reaching the ball carrier.

A Rush play is composed of (1) a starting formation and (2) instructions for each player in that formation. A formation is a set of (x,y) offsets from the center of the line of scrimmage. By default, instructions for each player consist of (a) an offset/destination point on the field to run to, and (b) a behavior to execute when they get there. Play instructions are similar to a conditional plan and include choice points where the players can make individual decisions as well as pre-defined behaviors that the player executes to the best of their physical capability. Rush includes three offensive formations (**power**, **pro**, and **split**) and four defensive ones (**23**, **31**, **2222**, **2231**). Each formation has eight different plays (numbered 1-8) that can be executed from that formation. Offensive plays typically include a handoff to the running back/fullback or a pass executed by the quarterback to one of the receivers, along with instructions for a running pattern to be followed by all the receivers. An example play from the **split** formation is given below:

- the quarterback will pass to an open receiver;
- the running back and fullback will run hook routes;
- the left wide receiver will run a corner right route;
- the right wide receiver will run a hook route;
- the other players will block for the ball holder.

4. Play Recognition using SVMs

In this paper we focus on intent recognition from the viewpoint of the offense: given a series of observations, our goal is to recognize the defensive play as quickly as possible in order to maximize our team's ability to intelligently respond with the best offense. Thus, the observation sequence grows with time unlike in standard offline activity recognition where the entire set of observations is available. We approach the problem by training a series of multi-class discriminative classifiers, each of which is designed to handle observation sequences of a particular length. In general, we expect that the early classifiers should be less accurate since

they are operating with a shorter observation vector and because the positions of the players have deviated little from the initial formation.

We perform this classification using support vector machines (Vapnik 1998). Support vector machines (SVM) are a supervised algorithm that can be used to learn a binary classifier; they have been demonstrated to perform well on a variety of pattern classification tasks, particularly when the dimensionality of the data is high (as in our case). Intuitively an SVM projects data points into a higher dimensional space, specified by a kernel function, and computes a maximum-margin hyperplane decision surface that separates the two classes. Support vectors are those data points that lie closest to this decision surface; if these data points were removed from the training data, the decision surface would change. More formally, given a labeled training set $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_l, y_l)\}$, where $\mathbf{x}_i \in \Re^N$ is a feature vector and $y_i \in \{-1, +1\}$ is its binary class label, an SVM requires solving the following optimization problem:

$$\min_{\mathbf{w},b,\xi} \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{l}\xi_i$$

constrained by:

$$\begin{aligned} y_i(\mathbf{w}^T\phi(\mathbf{x}_i) + \mathbf{b}) &\geq 1 - \xi_i, \\ \xi_i &\geq 0. \end{aligned}$$

The function $\phi(.)$ that maps data points into the higher dimensional space is not explicitly represented; rather, a *kernel* function, $K(\mathbf{x}_i, \mathbf{x}_j) \equiv \phi(\mathbf{x}_i)\phi(\mathbf{x}_j)$, is used to implicitly specify this mapping. In our application, we use the popular radial basis function (RBF) kernel:

$$K(x_i, x_j) = \exp(-\gamma||x_i - x_j||^2), \gamma > 0.$$

Several extensions have been proposed to enable SVMs to operate on multi-class problems (with k rather than 2 classes), such as one-vs-all, one-vs-one, and error-correcting output codes. We employ a standard one-vs-one voting scheme where all pairwise binary classifiers, $k(k-1)/2 = 28$ for every multi-class problem in our case, are trained and the most popular class is selected. Many efficient implementations of SVMs are publicly available; we use LIBSVM (Chang and Lin 2001). It is also possible to use the training paradigm described below to learn simpler supervised classifiers (e.g. kNN) in cases where there is no perceptual noise in the simulation data.

We train our classifiers using a collection of simulated games in Rush collected under controlled conditions: 40 instances of every possible combination of offense (8) and defense plays (8), from each of the 12 starting formation configurations. Since the starting configuration is known, each series of SVMs is only trained with data that could be observed starting from its given configuration. For each configuration, we create a series of training sequences that accumulates spatio-temporal traces from $t = 0$ up to $t \in \{2, \ldots, 10\}$ time steps. A multiclass SVM (i.e., a collection of 28 binary SVMs) is trained for each of these training sequence lengths. Although the aggregate number of binary classifiers is large, each classifier only employs a small fraction of the dataset and is therefore efficient (and highly paralellizable). Cross-validation on a training set was used to tune the SVM parameters (C and σ) for all of the SVMs.

Table 1: Play recognition results (all play combinations)

$t = 2$	3	4	6	8	10
12.50	96.88	96.87	96.84	96.89	96.81

Classification at testing time is very fast and proceeds as follows. We select the multiclass SVM that is relevant to the current starting configuration and time step. An observation vector of the correct length is generated (this can be done incrementally during game play) and fed to the multi-class SVM. The output of the intent recognizer is the system's best guess (at the current time step) about the opponent's choice of defensive play and can help us to select the most appropriate offense, as discussed below.

Table 1 summarizes the experimental results for different lengths of the observation vector (time from start of play), averaging classification accuracy across all starting formation choices and defense choices. We see that at the earliest timestep, our classification accuracy is at the baseline but jumps sharply near perfect levels at $t = 3$. This strongly confirms the feasibility of accurate intent recognition in Rush, even during very early stages of a play. At $t = 2$, there is insufficient information to discriminate between offense plays (perceptual aliasing), however by $t = 3$, the positions of the offensive team are distinctive enough to be reliably recognized.

5. Offensive Play Switches

To improve offensive performance, our system evaluates the competitive advantage of executing a *play switch* based on 1) the potential of other plays to improve the yardage gained and 2) the similarity of the candidate plays to the current play. To start, we train a set of SVM models to recognize defensive plays at a particular time horizon as described in the previous section; this training data is then used to identify promising play switches. A play switch is executed:

1. after the defensive play has been identified by the SVM classifier;
2. if there is a stronger alternate play based on the yardage history of that play vs. the defense;
3. if the candidate play is sufficiently similar to the current play to be feasible for immediate execution.

To determine whether to execute the play switch for a particular combination of plays, the agent considers N, the set of all offensive plays shown to gain more than a threshold ϵ value. The agent then selects $Min(n \in N)$, the play in the list most like the current play for each play configuration and caches the preferred play in a lookup table.

When a play is executed, the agent will use the first three observations to determine what play the defense is executing before performing a lookup to determine the play switch to make. The process is ended with execution of a change

order to members of the offensive team. Calculating the feasibility of the play switch based on play similarity is a crucial part of improving the team's performance; in the evaluation section, we evaluate our similarity-based play switch mechanism vs. a greedy play switching algorithm that focuses solely on the potential for yardage gained.

5.1 Play Similarity Metric

To calculate play similarities, we create a feature matrix for all offensive formation/play combinations based on the training data. Features collected for each athlete A include max, min, mean, and median over X and Y in addition to the following special features:

FirstToLastAngle: Angle from starting point (x_0, y_0), to ending point (x_n, y_n), is defined as $atan\left(\frac{\triangle y}{\triangle x}\right)$

Start_Angle: Angle from the starting point (x_0, y_0) to (x_1, y_1), defined as $atan\left(\frac{y_1 - y_0}{x_1 - x_0}\right)$

End_Angle: Angle from the starting point (x_{n-1}, y_{n-1}) to (x_n, y_n), defined as $atan\left(\frac{\triangle y}{\triangle x}\right)$

Total_Angle: $= \sum_{i=0}^{N-1} atan\left(\frac{y_{i+1} - y_i}{x_{i+1} - x_i}\right)$

Total_Path_Dist: $= \sum_{i=1}^{N-1} \sqrt[2]{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2}$

Feature set F for a given play c, ($c = 1...8$, represents possible play matches per formation) contains all features for each offensive player in the play and is described as

$$\overrightarrow{F_c} = \{A_{c1} \cup A_{c2} \cup \cdots \cup A_{c8}\}$$

These features are similar to the ones used in (Rubine 1991) and more recently, by (Wobbrock *et al.* 2007) to match pen trajectories in sketch-based recognition tasks, but generalized to handle multi-player trajectories. To compare plays we use the sum of the absolute value of the differences (L_1 norm) between features F_{c_i} and F_{c_j}. This information is used to build a similarity matrix M_{ij} for each possible offensive play combination as defined below.

$$M_{ij} = \sum_{c=1}^{\left\|\overrightarrow{F_c}\right\|} \Delta\overrightarrow{F_c}$$
$$i, j = 1 \ldots 8$$

There is one matrix M for each offensive formation O_β, where $\beta \in \{\text{pro, power, split}\}$ are the offensive formations. Defensive formation/play combinations are indicated by $D_{\alpha p}$, where $\alpha \in \{23,\ 31,\ 2222,\ 2231\}$ and p represents plays 1..8. *M* for a specific play configuration is expressed as $O_\beta D_{\alpha p} M_i$, given *i (1... 8)* is our current offensive play. The purpose of this algorithm is to find a value *j* (play) most similar to *i* (our current play), with a history (based on earlier observation) of scoring the most yardage. This process is accomplished for every offensive play formation against every defensive play formation and play combination. When the agent is constructing the lookup table and needs to determine the most similar play from a list, given current play i, it calls the method, $Min(O_\beta D_{\alpha p} M_i)$ which returns the most similar play.

6. Improving the Offense

Our algorithm for improving Rush offensive play has two main phases, a preprocess stage which yields a play switch lookup table and an execution stage where the defensive play is recognized and the offense responds with an appropriate play switch for that defensive play. We train a set of SVM classifiers using 40 instances of every possible combination of offense (8) and defense plays (8), from each of the 12 starting formation configurations. This stage yields a set of models used for play recognition during the game. Next, we calculate and cache play switches using the following procedure:

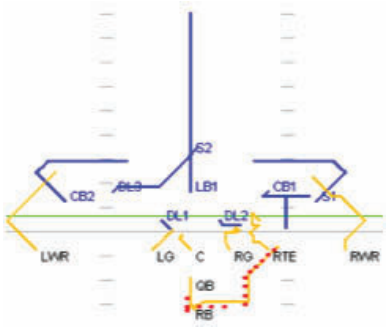
1. Collect data by running RUSH 2008 50 times for every play combination.
2. Create yardage lookup tables for each play combination. This information alone is insufficient to determine how good a potential play is to perform the play switch action on. The transition play must resemble our current offensive play or the offensive team will spend too much time retracing steps and perform very poorly.
3. Create feature matrix for all offensive formation/play combinations.
4. Create the final play switch lookup table based on both the yardage information and the play similarity.

To create the play switch lookup table, the agent first extracts a list of offensive plays *L* given the requirement $yards\,(L_i) > \epsilon$ where ϵ is the least amount of yardage gained before the agent changes the current offensive play to another. We used $\epsilon = 1.95$ based on a quadratic polynomial fit of total yardage gained in 6 tests with $\epsilon = \{MIN, 1.1, 1.6, 2.1, 2.6, MAX\}$ where MIN is small enough no plays are selected to change and MAX where all plays are selected for change to the highest yardage play with no similarity comparison. Second, from the list L find the play most similar (smallest value in the matrix) to our current play *i* using $Min(O_\beta D_{\alpha p} M_i)$ and add it to the lookup table.

During execution, the offense uses the following procedure:

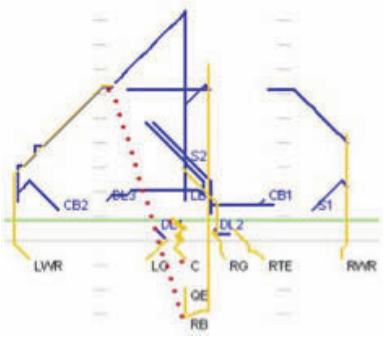
1. At each observation less than 4, collect movement traces for each play.
2. At observation 3, use LIBSVM with the collected movement traces and previously trained SVM models to identify the defensive play.
3. Access the lookup table to find $best(i)$ for the current play *i*.
4. If $best(i) \neq i$, send a change order command to the offensive team to change to play $best(i)$.

However, it's not necessary (or always desirable) to change all players to the new play. We also evaluated the performance of subgroup switching; modifying the actions of a small group of key players, while leaving the remaining players alone. By segmenting the team we are able to combine two plays previously identified as alike to each other with regard to spatio-temporal data, but different in regards



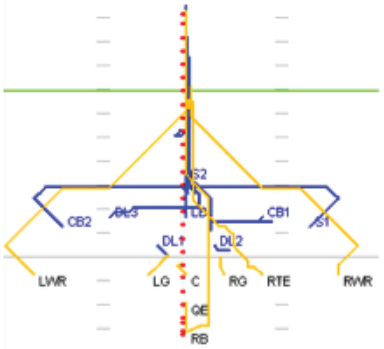
Starting play

```
QB: 0 30 HANDOFFRB
RB: 70 -15 SWEEP
LWR:-30 -30 RUNBLOCK
{RWR,RTE,LG,C,RG}:0 0 RUNBLOCK
```



Commands for all players changed to optimal play

```
QB: 0 30 PASS
RB: -70 -40 PASSROUTE_BOMB
LWR:0 -50 PASSROUTE_CORNERRIGHT
RWR:0 -50 PASSROUTE_HOOK
{RTE,LG,C,RG}:0 0 PASSBLOCK
```



Adapted play with only key subgroup players changing commands

```
QB: 0 30 PASS
RB: -70 -40 PASSROUTE_BOMB
LWR:-30 -30 RUNBLOCK
{RWR,RTE,LG,C,RG}:0 0 RUNBLOCK
```

Figure 2: The play in the left figure is adapted using the middle to produce the adapted play shown on the right. The green line indicates average yardage gained.

to yards gained. Based on our domain knowledge of football, we selected three subgroups for evaluation: 1) QB, RB, and FB; 2) LG, C, and RG; 3) LWR, RWR, RTE, LTE.

Figure 2 is a good example of a very successful merge of two plays which produced a superior play with subgroup switching. The green line represents the average yardage gained. The left image is the most likely path of the baseline case (a running play which yields little yardage on average). The middle image is the most likely execution trace produced by the total play switch method. The play produced by the total play switch was not much more successful than the baseline case. However, when only Group 1 (QB, RB, FB) is modified, the success of the play increases greatly and the new play is shown to be very coordinated and effective.

7. Empirical Evaluation

Our switching algorithm was tested using the RUSH 2008 simulator for ten iterations of each possible play configuration in three separate trials. We compared our play switch model (using the yardage threshold $\epsilon = 1.95$ as determined by the quadratic fit) to the baseline Rush offense and to a greedy play switch strategy ($\epsilon = MAX$) based solely on the yardage (Figure 3).

Overall, the average performance of the offense went from 2.82 yards per play to 3.65 yards per play ($\epsilon = 1.95$) with a total gain of 29%, $\pm 1.5\%$ based on sampling of 3 sets of 10 trials. An analysis of each of the formation combinations (Figure 3) shows the yardage gain varies from as much as 100% to as little as 0.1%. Overall, performance is consistently better for every configuration tested. In all cases, the new average yardage is over 2.3 yards per play with no weak plays as seen in the baseline with Power vs. 23 (1.4 average yards per play) and Power vs. 2222 (1.3 average yards per play). Results with $\epsilon = MAX$ clearly shows simply changing to the greatest yardage generally results in

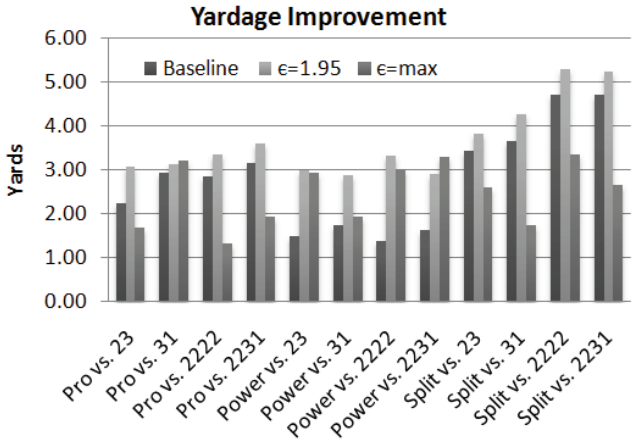


Figure 3: Comparison of greedy play switch and similarity-based switching. Our similarity-based play switch method outperforms both baseline Rush offense and a greedy play switch metric.

poor performance from the offense. Power vs. 23 is boosted from 1.5 yards to 3 yards per play, doubling yards gained. Other combinations, such as Split vs. 23 and Pro vs. 32 already gained high yardage and improved less dramatically at about .2 to .4 yards more than the gains in the baseline sample. In Figure 3 we see all the split configurations do quite well; this is unsurprising given our calculations of the best response. However, when the threshold is not in use and the plays are allowed to change regardless of current yardage, the results are greatly reduced. The reason seems to be associated player miscoordinations induced by the play switch; by maximizing the play similarity simultaneously, the pos-

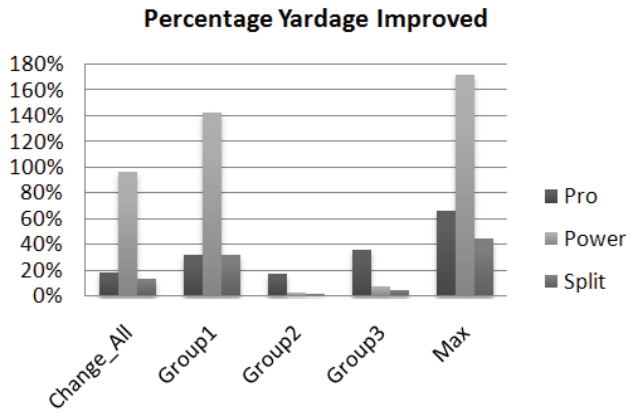


Figure 4: The play-yardage gain over baseline Rush offense yielded by various play switch strategies.

sibility of miscoordination errors is reduced.

To evaluate the subgroup switching, we ran the simulation over all 3 subgroups and compared them to the baseline yardage gained and the results of total play switch. Results clearly indicated the best subgroup switch (consistently Group 1) produced greater gains than the total team switch, which still performed better than the baseline. Figure 4 is a side-by-side comparison of the results. We also compared results to the yardage gained if the team had initially chosen the best response play (the play that on average results in the greatest yardage gain) for that formation. Early play recognition combined with subgroup switching yields the best results, assuming no oracular knowledge of the other team's intentions prior to run-time.

8. Conclusion

Accurate opponent modeling is an important stepping-stone toward the creation of interesting autonomous adversaries. In this paper, we present an approach for online strategy recognition in the Rush 2008 football simulator. Using information about the defense's intent, our system evaluates the competitive advantage of executing a play switch based on the potential of other plays to improve the yardage gained and the similarity of the candidate plays to the current play. Our play switch selection mechanism outperforms both the built-in Rush offense and a greedy yardage-based switching strategy, increasing yardage while avoiding miscoordinations induced by the greedy strategy during the transition from the old play to the new one. Additionally, we demonstrate limiting play switching to a subgroup of key players further improves performance. In future work, we plan to explore methods for automatically identifying key subgroups by examining motion correlations between players.

Acknowledgments

Gita Sukthankar was partially supported by an ONR ASEE Summer Faculty Fellowship (2008) and the DARPA Computer Science Study Group (2009). A special thanks to David Tristan Brewster for providing key insight about American football strategies.

References

iPod Video "Saves" the Day for Manchester United, 2009. http://www.jeffbennett.org/2009/03/ipod-video-saves-the-day-for-manchester-united.

C.-C. Chang and C.-J. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

S. Intille and A. Bobick. A framework for recognizing multi-agent action from visual evidence. In *Proceedings of National Conference on Artificial Intelligence*, 1999.

M. Jug, J. Pers, B. Dezman, and S. Kovacic. Trajectory based assessment of coordinated human activity. In *Proceedings of the International Conference on Computer Vision Systems (ICVS)*, 2003.

G. Kuhlmann, W. Knox, and P. Stone. Know thine enemy: A champion RoboCup coach agent. In *Proceedings of National Conference on Artificial Intelligence*, 2006.

N. Li, D. Stracuzzi, G. Cleveland, P. Langley, T. Konik, D. Shapiro, K. Ali, M. Molineaux, and D. Aha. Constructing game agents from video of human behavior. In *Proceedings of IJCAI Workshop on Learning Structural Knowledge from Observations*, 2009.

M. Molineaux, D. Aha, and G. Sukthankar. Beating the defense: Using plan recognition to inform learning agents. In *Proceedings of Florida Artifical Intelligence Research Society*, 2009.

P. Riley and M. Veloso. Recognizing probabilistic opponent movement models. In A. Birk, S. Coradeschi, and S. Tadorokoro, editors, *RoboCup-2001: Robot Soccer World Cup V*. Springer Verlag, 2002.

P. Riley, M. Veloso, and G. Kaminka. An empirical study of coaching. In H. Asama, T. Arai, T. Fukuda, and T. Hasegawa, editors, *Distributed Autonomous Robotic Systems 5*. Springer-Verlag, 2002.

D. Rubine. Specifying gestures by example. *Computer Graphics, Volume 25, Number 4*, pages 329–337, 1991.

Rush, 2005. http://sourceforge.net/projects/rush2005/.

Gita Sukthankar and Katia Sycara. Policy recognition for multi-player tactical scenarios. In *Proceedings of International Conference on Autonomous Agents and Multi-agent Systems (AAMAS)*, May 2007.

V. Vapnik. *Statistical Learning Theory*. Wiley & Sons, Inc, 1998.

J. Wobbrock, D. Wilson, and L. Yang. Gestures without libraries, toolkits or training: a $1 recognizer for user interface prototypes. In *Symposium on User Interface Software and , Proceedings of the 20th annual ACM symposium on User interface software and technology*, 2007.