Machine Learning Techniques for Analyzing Training Behavior in Serious Gaming

Matthew C. Gombolay, Reed Jensen, Sung-Hyun Son
Massachusetts Institute of Technology Lincoln Laboratory
244 Wood St, Lexington, MA 02421
{matthew.gombolay,rjensen,sson}@ll.mit.edu

***Abstract*—Training time is a costly, scarce resource across domains such as commercial aviation, healthcare, and military operations. In the context of military applications, serious gaming – the training of warfighters through immersive, real-time environments rather than traditional classroom lectures – offers benefits to improve training not only in its hands-on development and application of knowledge, but also in data analytics via machine learning. In this paper, we explore an array of machine learning techniques that allow teachers to visualize the degree to which training objectives are reflected in actual play. First, we investigate the concept of discovery: learning how warfighters utilize their training tools and develop military strategies within their training environment. Second, we develop machine learning techniques that could assist teachers by automatically predicting player performance, identifying player disengagement, and recommending personalized lesson plans. These methods could potentially provide teachers with insight to assist them in developing better lesson plans and tailored instruction for each individual student.**

I. Introduction

An increase in the sheer number and complexity of missile threats to national security have prompted researchers in the Department of Defense to develop innovative decision support tools that promote better decision-making for the warfighter. For the air and missile defense mission, initial research in this area began with simple Red/Blue wargaming exercises, where warfighters played against each other (i.e., red for offense and blue for defense) to solve challenging, unsolved tactical problems. Playing these games not only allowed the warfighter to discover and learn new tactics, techniques, and procedures, but also allowed the researchers to solicit feedback from the warfighter to refine the development of their decision support tools. While the data and feedback collected were invaluable, the training and educational aspects were static and limited by the sample size and update rate.

Limitations in conveying and collecting information across relevant sample sizes have motivated a data-driven, game-based simulation approach. For example, industry and academia alike are keenly interested in understanding player types and behaviors in games to better tailor the gameplay experience [17], [33], [36], [43], [45], [47]. A key component of understanding player behavior is performance prediction. Performance prediction allows the educator to efficiently focus attention on those students who are struggling and need help. Further, performance prediction allows one to determine with less time spent on testing whether a student is actually proficient in a domain and ready to proceed to the next subject.

Still others within the field of education have thereby sought to develop methods for understanding why students, or players, drop out of educational programs [11], [14], [20], [30]. Students becoming disengaged in learning exercises is a chronic problem that greatly hampers the ability of educators to give students the tools they need to succeed [11], [14], [20], [30]. Researchers in artificial intelligence and machine learning have sought to develop methods for predicting student and player retention [3], [13], [29], which is a strong first step in correcting the problem of trainee dropout.

We have conducted a study using machine learning with a serious gaming (i.e., game designed as a professional training tool) platform to support data-driven analytics of human subjects for serious gaming. It is our aim that this analysis (1) demonstrates the power of a variety of machine learning techniques for data-driven analytics, (2) gives insight into how to discover and interpret meaning in human-generated data, and (3) serves as an informative machine-learning use case for researchers who wish to harness the power of data for decision-support. To our knowledge, this is the first such investigative approach into applying machine learning techniques to study the training of warfighters through serious gaming. Further, we believe ours is the first to employ a generative model to learn from demonstration how to best order training experiences in the form of a lesson plan.

The paper is structured as follows. In Section II, we briefly survey related work in the fields of education, which includes both traditional studies of human teachers and students as well as the development of computational methods to augment traditional education techniques. In Section III, we discuss the educational platform used in our investigation: Strike Group Defender. In Section IV, we describe our real-world data set of human players, which we use to perform our computational investigations. In Sections V-VI, we show how unsupervised learning techniques can be applied to serious gaming to derive insights into player types and the strategies players develop during gameplay. Section VIII show how one can accurately predict player disengagement. In Section VII, we investigate how to predict how well players will perform on a test scenario

DISTRIBUTION STATEMENT: A. This material is based upon work supported by the Department of the Navy under Air Force Contract No. FA8721-05-C-0002 and/or FA8702-15-D-0001. Any opinions, findings, conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Department of the Navy. RAMS # 1001485

Fig. 1. SGD enables development of automated teaching tools for ASMD.

as well as how to learn the features that are most important for predicting that performance. In Section IX, we present a novel method for learning how to automatically generate lesson plans based on demonstrations of students' self-play of the game. To our knowledge, our approach is the first to investigate learning to recommend lesson plans from human demonstration. Finally, we review our findings in Section X.

II. Related Work

A. Discovery of Player Types and Tactics

In developing a training curriculum, game, or training environment, it is critical to understand how players interact with the game. If the designer is able to better understand what types of players exist and how they play the game, the designer can improve the gameplay experience for the users. Recent work has sought to better understand player types and strategies for this very reason [17], [33], [36], [43], [45], [47]. For example, van Lankveld et al. [47] perform a correlation analysis between players' personality profiles [15] and how people move and converse with other players and non-player characters. Kim and Kim apply multiple linear regression to identify key correlates relating age and style of game play within a real-time strategy game, StarCraft [26]. Sarratt and Pynadath et al. utilize a Monte-Carlo Tree Search to learn belief distributions over player types to adapt the game in response to user preferences and actions [42]. Drachen et al. use archetypal analysis to identify unique player archetypes to better understand how people naturally assume in-game roles [17]. Thurau and Bauckhage apply matrix factorization to learn lower-dimensional representations of player categories in an massively multi-player online role-playing game with over 192 million recordings of 18 million characters [45].

These approaches provide a panoply of methods for unsupervised learning in gaming. To the best of our knowledge, however, these techniques have not been explored in the context of serious gaming for training of professionals. As such, we believe our investigation provides the first demonstration that these techniques can benefit a largely unexplored world of data-driven training in serious gaming.

B. Personalized Lesson Plans

Generating effective lesson plans is a key role of a teacher [1], [8], [21], [44], [49]. Generally, a lesson plan consists of a 'learning trajectory', or a sequence of topics to teach students. In the context of traditional education (i.e., one not augmented with a technological aid), Griffey and Housner sought to understand how teachers collect information to effectively plan lessons for their students. In their study, they investigated differences between the more experienced and inexperienced physical education teachers. They found that more experienced teachers asked more questions before constructing their lesson plans as compared to the less inexperienced teachers. The more experienced teachers' lesson plans considered more contingencies and garnered more engagement from their students [21]. Researchers have sought to leverage such findings to create frameworks for manually crafting lesson plans [8], [44], [49].

As an alternative to manual, observational studies [8], [44], [49], the field of educational data mining has sought autonomous methods (i.e., machine learning) to augment the educational process [5], [6], [12], [13], [16], [40]. For a survey, we recommend the reviews by Baker and Yacef [2] as well as Romero and Ventura [39]. Nonetheless, we are aware of only one prior investigation into automating the generation of lesson plans [1]. In one study, Yang et al. [1] developed a human-in-the-loop system which learns from teachers how to generate lesson plans. In this framework, teachers would specify certain constraints (e.g., amount of time available to teach) and preferences for topics they want to teach. If the system cannot satisfy the teachers' preferences given the constraints, the system will recommend how to augment the lesson plan to satisfy the constraints.

Butler et al. develop a system to recommend and sequence gameplay content to increase players' skills and knowledge while maintaining their engagement [9]. The approach used by Butler et al. relies on a content-enumeration method from a prior work [10] to create a library of levels and a hand-crafted metric to assess players' mastery of content to recommend which level should be played next [9]. Furthermore, Grappiolo et al., explored the problem of using serious games to teach conflict resolution skills [19]. They developed a genetic algorithm with a hand-coded fitness function to automatically generate content based upon a player's gameplay experience to best improve their skill at conflict resolution [19].

Nonetheless, we are not aware of any investigation which develops a data-driven approach for generating lesson plans, or trajectories of levels, based on demonstrations of students' self-learning. In this paper, we present such a method, based on a Hidden-Markov Model (HMM), that can recommend a personalized, sequence of levels that a player should complete to enhance his/her performance.

C. Classifying Players and Predicting Performance

Predicting students' performance has been studied widely in the field of artificial intelligence [2], [5], [6], [12], [13], [16], [40]. For example, Beck et al. developed a learning agent that models student behavior in the study of mathematics. This model learns from examples of students interacting with a virtual tutor [5], [6]. Using linear regression and hand-crafted features, Baker et al., showed that one can reasonably predict how long students will take to respond to questions and how accurate those students' answers will be [2].

Romero et al. [40] developed a data mining tool, built into an online courseware system, that could be utilized by online instructors. Romero et al. considered data from 438 Cordoba University students enrolled in seven online courses with the goal of classifying students by their final grades in the courses. They found that a decision tree [7], [38] was able to predict grades with $\sim 65\%$ accuracy using hand-crafted features [40]. Within the online gaming community, the TrueSkill™ metric was derived to rank players on the Xbox Live platform [23]. The TrueSkill™ metric is based on a Bayesian formulation and serves as a generalization of the Elo rating, which is widely employed in ranking Chess players [18].

In our work, we seek not only to learn such a method of predicting player performance but also the features that describe those players. We have not seen such an investigation within the context of serious gaming for military training.

D. Player Disengagement

One of the most important aspects to providing a quality education is keeping students engaged in teaching activities. Academia has conducted much work to determine why students become disengaged or quit school at various levels of education [11], [14], [20], [30]. Through case studies, researchers have shown that students' attitudes towards their teachers [11], beliefs of the role in the partnership between home and school, and lack of time due to other responsibilities [30] all have major effects in the engagement and eventual success of students. The challenge of disengagement is common to gaming as well. Anecdotal evidence from industry partners suggests that only 5% of people who download a game for their smart phone even open the game itself.

While professionals may not be able to quit their training programs, they can become disengaged or frustrated, and their education can suffer. Accordingly, educators should also be given downstream tools to identify when a player is about to disengage. With such a tool, educators can can interrupt the student's behavior, identify the problem, and guide the student through the difficulty. Recent work by Bauckhage et al. has shown that such a model can be learned with big data in action-adventure and shooter games [3]. In their analysis, they found that the average player's interest in the game evolves according to a non-homogeneous Poisson process implying that initial gameplay behavior could predict when a player will stop playing [3]. In another study, Mahlmann et al. used a suite of supervised learning techniques, such as logistic regression, decision trees, naive Bayes, and support vector machines, to predict how many levels of a seven-level game a player will complete based on data of the player's experience on the first one or two levels [29]. Within the sports gaming community, Weber et al. [48] developed a model to predict when players would disengage from playing *Madden NFL 11*. Based on their finding that players who employ a variety of "plays" are more likely to quit earlier, Weber et al. recommended the playbook be shortened [48]. Further, Pedersen et al. explored predicting the frustration and challenge a player experiences based on level design within *Super Mario Bros.* [34].

Based on the need to help educators with disengagement [11], [14], [20], [30] and the recent work in this area [3], [13], [29], we have developed a mechanism, which we describe in Section VIII, to predict when players are about to disengage before it happens so that an educator can intervene and facilitate learning. While the problem of player disengagement has been studied, we have not seen it applied in the context of serious gaming for military education.

III. Investigative Platform

We have developed a game-based simulation, called Strike Group Defender (SGD), to emulate anti-ship missile defense exercises. SGD provides users across a variety of locations and platforms with both single- and multi-player training experiences in the context of relevant naval scenarios. SGD collects participant actions and game events to analyze and refine the educational experience of the users either post hoc or in real time. The data-based collection capability of SGD has opened the way for the development of machine learning approaches that can analyze the user's educational experience.

In the most recent version of the SGD application (Figure 1), users must learn and employ the techniques and tactics relevant to the defense of naval assets against anti-ship missiles (hereafter referred to as ASMD). The game focuses on the proper use of naval electronic warfare – the use of signals instead of missiles for ship defense, otherwise known as *soft-kill* weapons (i.e., decoys) – but also includes *hard-kill* weapons (i.e., interceptor missiles) and information, surveillance, and reconnaissance (ISR) elements. Players assign and deploy soft-kill weapons (e.g., flare, chaff, etc.) to deceive or distract enemy missiles away from valuable ships. The proper coordination of soft-kill decoys with hard-kill interceptor missiles and ISR limitations ensures the long-term survivability of the ships in the strike group against a formidable raid of heterogeneous anti-ship cruise missiles. We use this platform for our investigation into how to learn player types and strategies as well as how to educate players to be more proficient executors of ASMD tactics.

IV. Data Set

For our analysis, we collected data of players training and competing in red-blue exercises in SGD. This data consists of people playing *training* and *test* levels. There is one training level for each threat type to teach players general techniques to combat each missile type as well as an introductory tutorial level and an tutorial exam. The tutorial levels are as follows: "Basics Tutorial," "Hungry Missile Tutorial," "Moth Missile Tutorial," "Catfish Missile Tutorial," "Longshot Missile Tutorial," "Weasel Missile Tutorial," "Muffled Missile Tutorial," "Headhunter Missile Tutorial," "Cerberus Missile Tutorial," and "Tutorial Exam."

There are also three test levels: "Daily Performance Evaluation (DPE)," "Operation Neptune," and "Final Countdown." DPE is a level where threat types are randomized, and threat bearings are spread across a range of angles such that it appears one's own ship is surrounded. The DPE scenario changes once per day. Operation Neptune is a deterministic level consisting of three raids of threats (i.e. swarms of many missiles) , and the player must defend three ships: two destroyers and an aircraft carrier. This level is particularly challenging because of the sheer volume and difficulty of the missile types and the large radar cross-section of the aircraft carrier. Final Countdown is a level designed to be a final test in which players can only play the scenario once. Lastly, players could construct their own custom levels as well as watch gameplay replays of any game played by another player.

We collected data in two phases. First, we conducted a month-long March Madness tournament at MIT Lincoln Laboratory. Lincoln Laboratory personnel were then invited to play SGD for two weeks. Players with the top scores were invited to compete in a bracket-style tournament. Players scores were based on a composite measure of performance across the three test levels, DPE, Operation Neptune, and Final Countdown. Specifically, the composite score for each player was the sum of (1) the average score across DPE games, (2) the maximum score across Operation Neptune games, and (3) the score from the one Final Countdown game. We refer to this score as the "tournament score." A total of 16 players were selected for participation in the tournament based on their tournament score.

Bracket tournament players would play SGD in red-versus-blue mode, and the player with the top score serving in the defensive role would advance. There were two such red-versus-blue levels: "Round 1 Challenge" and "Round 2 Challenge." The winner of the bracket won \$500. We hereafter refer to data collected from the first phase as the March Madness data. The March Madness data set consists of 148 players, > 100 hours of gameplay, and $> 3,000$ games played. In these data, 34 accounts played the DPE level at least once, 45 accounts played Operation Neptune at least once, and 29 accounts played Final Countdown.

In a second phase, we collected data from an additional 76 players, for a total of 224 players. This additional data increased the number of players who played each test level at least once to 70 for the DPE, 82 for Operation Neptune, and 61 for Final Countdown.

Due to the nature of their employment, all participants were experienced in the use of computational technology. The only inclusion criterion was employment at MIT Lincoln Laboratory at the time of participation. Participants demonstrated a variety of skill levels. Participants' tournament scores had a mean and standard deviation of 65 ± 19 (normalized to the interval $[0, 100]$). This mean and standard deviation include only 49 of all 226 players, as not all players completed the levels required to achieve a tournament score.

V. Discovering Player Types

First we sought to answer what types of players emerge and what types of strategies players develop. We employed k-means clustering to discover k types of players that emerge during gameplay based on features describing how they used

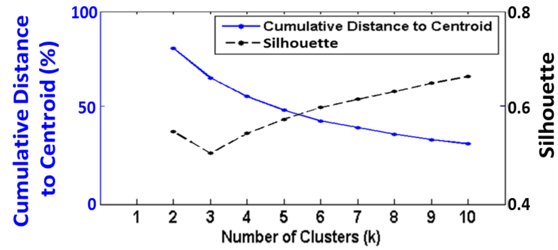

Fig. 2. This figure depicts the cumulative distance and silhouette for clustering our data with $k \in \{2, 3, \ldots, 10\}$ centroids.

the game (e.g., how often they pause the game, how efficiently they use soft-kill, etc.). The k-means algorithm, proposed by MacQueen et al. [28], finds the k centroids that minimizes the sum of the distances from each centroid, k, to the location of each players associated with that centroid. Specifically, if we have a set of n observations $\{x_i | i \in \{1, 2, \ldots, n\}, x_i \in \Re^m\}$ (e.g., m features describing each player), k-means attempts to create k partitions $\boldsymbol{S} = \{S_1, S_2, \ldots, S_k\}$ such that Equation 1 is minimized. In this equation, $\mu_j = \frac{1}{n} \sum_{x_i \in S_j} x_i$ is the centroid (i.e., mean) of cluster j.

$$\arg\min_{\boldsymbol{S}} \sum_{j=1}^{k} \sum_{x_i \in S_j} \| x_i - \mu_j \| \tag{1}$$

Solving Equation 1 is an NP-Hard problem and is not computationally tractable for large, real-world data sets. As such, we use an adaptation of the approximation technique known as Expectation Maximization (EM), which is an iterative method for finding the maximum a posteriori estimate of your decision variables [32].

We apply the k-means clustering algorithm to our data to determine what natural player types arise when playing SGD. To determine the number of clusters, k, we employ two metrics: the cumulative distance and the silhouette. The cumulative distance metric is equal to the inner error term, $\sum_{j=1}^{k} \sum_{x_i \in S_j} \| x_i - \mu_j \|$, from Equation 1. This metric gives us a sense of how far away individual players are from their representative centroid. The closer the player is to their centroid, the better. Alternatively, the silhouette [41] is a measure of how far a player is to the second-closest centroid. Specifically, the silhouette of a set of centroids and points assigned to those centroids is defined by Equation 2, where $a(i)$ is the distance between x_i and the closest centroid, and $b(i)$ is the distance between x_i and the second-closest centroid. The silhouette is in the interval $[-1, 1]$. We note that the silhouette will be > 0 and the denominator (i.e., $\max(a(i), b(i))$ from Equation 2) will equal $b(i)$ if one has assigned x_i to the closest cluster; otherwise, the silhouette will be < 0 and the denominator will equal $a(i)$.

$$\text{Silhouette} := \sum_{x_i} \frac{b(i) - a(i)}{\max(a(i), b(i))} \tag{2}$$

We report the cumulative distance and silhouette of our data for $k \in \{2, 3, \ldots, 10\}$ in Figure 2.These two metrics help to balance key trade-offs in clustering data. As we increase the number of clusters, we strictly decrease the cumulative distance metric, as shown in Figure 2, which is generally a good goal. However, as we increase the number of clusters, we also make the data more sparse; there are fewer members of each cluster. With fewer members in each cluster, we cannot be as certain about the parameters defining the clusters (i.e., the mean feature values). Thus, we want to find an "elbow" in the curve where the cumulative distance plateaus. However, we also must consider how far each cluster is from the other clusters. The silhouette metrics helps us to see how well we have separated the data. If the silhouette is relatively low, then there are points that are near the boundaries between two or more centroids. If the silhouette is high, then the clusters are spaced further apart. The silhouette does not strictly increase with the k. As such, one must set k so that the cumulative distance is low, the silhouette is high, and there are enough data points within each cluster. Of course, one could use a Bayesian prior (e.g., Chinese restaurant process) over k if one believes their data are well-suited to such an interpretation.

Based on our data and the metrics shown in Figure 2, we select $k = 4$ for our analysis. The cluster centroids for $k = 4$ are shown in Figure 3. The features we study are as follows:

- # Games Quit / # Games Played - number of games a player quits divided by the total number of games played by the player.
- Mean (# Repeats per Tutorial Level) - average number of times a player repeats a tutorial level.
- # Replays / # Games Played - number of times a player has watched a replay of another player's game divided by the total number of games played by the player.
- # Repeated Games / # Games Played - number of times a player starts an already-played level divided by the total number of games played by the player.
- # Unique Tests / # of Games Played - number of unique test levels the player has attempted divided by the total number of games played.
- # Avg Pause Time - average amount of time the player pauses a game.
- # Unique Tutorials / # Unique Tests - number of unique tutorial levels attempted divided by the number of unique test levels attempted by the player.
- # Unique Tutorials / # Games Played - number of unique tutorial levels attempted by the player divided by the total number of games played by the player.
- # Unique Tutorials - number of unique tutorial levels attempted by the player.

We evaluate these features for players after the first 20 games (shown in purple) and at the conclusion of the players' participation (shown in blue). We consider the feature percentiles for each player so as to better separate the data.

For our study, we did not use measures of performance (i.e., game scores) or the total number of games played. Our aim was to understand player behavior rather than player performance. Understanding player behavior can help game designers better develop training levels to guide players towards exploring the game in a desired fashion. Furthermore, instructors can use data based on player behavior to tailor their instruction so that it is best-suited for each type of player. Rather than emphasizing short-term performance, the goal is to find the right behaviors to solicit good performance in the long run and help generalize accumulated experience across future learning disciplines.

While we did not use measures of performance or the total number of games played, we can associate those feature values with the cluster post hoc. In Figure 3, we report both the tournament score described in Section IV, which considers performance on three specific levels, and the "average score," which averages a player's score across all levels. The tournament score was conveyed to players as the most important evaluation metric; however, we also report the average score across all levels and the average number of games attempted by players in each cluster.

In studying the clusters, we can identify several trends about different player types. For example, the best players (far left) tend to explore the game by repeating each level attempted before moving on to another tutorial level. These players also tend to quit rather than pause games. Medium-high performers (center-left), on the other hand, tend to explore the tutorial levels in a breadth-first fashion, rather than practicing levels already attempted. Medium-low performers (center-right) explore very few tutorial levels, repeat the same few levels, and do not typically pause. The worst performers (far right) spend more time with the game paused and quit levels before their completion less frequently.

Yet, it is the third cluster (center-right) that stands in most contrast with the rest. This cluster is characteristic of players who infrequently pause, proceed immediately to the test levels, which are significantly more difficult, and repeat those difficult levels ad nauseam. While this cluster has a slightly higher, though not with statistical significance, tournament score than that of the far-right cluster, the average score is by far the lowest. Further, these players have attempted the fewest number of games. As such, it is likely that this cluster represents players whose knowledge is the least robust.

This analysis indicates that certain player types naturally develop within our gaming environment. However, this analysis is correlative rather than causative. For example, high-performing players appear to quit games frequently when they know that achieving a high score is no longer possible. However, it is unclear whether forcing low-performing players to quit (rather than pause), after a few missiles leak through their defenses, would cause them to improve their skills. Further, it is unclear what would happen if certain features of the game (e.g., the ability to pause) were removed. Nonetheless, the value of this cluster analysis is two-fold. First, an instructor could use this cluster analysis to develop training plans for players of each type. Second, the instructor could identify which lesson training plan is then appropriate given the algorithm's assignment of a new player's cluster membership.

VI. Discovering Player Tactics

To discover what strategies exist, we used k-medoids clustering, as presented in Kaufman et al. [25]. In k-medoids

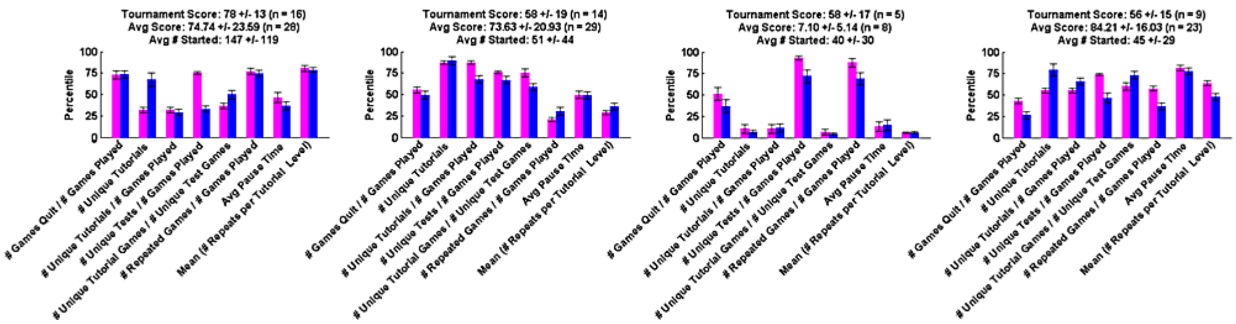

Fig. 3. This figure depicts the cluster centroids for player types with $k = 4$.

clustering, one attempts to find the k points (i.e., medoids or exemplars), as opposed to centroids, that best represent the other points assigned to each medoid's partition. Specifically, if we have a set of n observations $\{x_i | i \in \{1, 2, \ldots, n\}, x_i \in \Re^m\}$ (e.g., m features describing each player), k-medoids attempts to create k partitions $\mathbf{S} = \{S_1, S_2, \ldots, S_k\}$ such that Equation 3 is minimized. In this equation, m_j is the medoid of cluster j, and $\boldsymbol{M}$ is the set of k medoids.

$$\arg\min_{\boldsymbol{S},\boldsymbol{M}} \sum_{j=1}^{k} \sum_{x_i \in S_j} \|x_i - m_j\| \tag{3}$$

We use k-medoids here because while it is easy to conceptualize the average of features describing players, taking the average of two strategies does not intuitively yield a descriptive strategy representing the two original strategies.

However, the k-medoids formulation in Equation 3 is not specific enough to cluster players' games. Each game is comprised of a set of points describing the actions taken in the game. At each moment in time, a player may choose to deploy one of a number of assets in the game to defend one's ship against anti-ship missiles. The player may deploy this asset in any location along the surface of the environment. The surface is modeled as a plane in $\mathbb{R}^2$ as opposed to the surface of a spherical or ellipsoidal Earth. Thus, games may contain differing numbers of deployments, changing the cardinality of any feature vector describing the game. Thus, we need some non-parametric means of finding the distance between two games to partition the games into clusters.

A common method for measuring the distances between two non-empty sets of points, A and B, is the Hausdorff Distance, $H(A, B)$, as shown in Equations 4-5.

$$H(A, B) = \max\{h(A, B), h(B, A)\} \tag{4}$$
$$h(A, B) = \max_{a \in A} \min_{b \in B} \|a - b\| \tag{5}$$

In our application, a and b would be feature vectors describing individual asset deployments in games A and B, respectively.

We must make one further alteration, however. Each action taken in the game, such as the deployment of an asset or the relocating of a ship, is not always directly comparable to another action taken in a different game or even within the same game. For example, the "distance" between two deployments of the same asset would presumably be smaller than the distance between deployments of different assets, all other things being equal. As such, we adopt a tunable, weighting scheme to more fairly compare pairs of actions taken within games. We state that the distance $\delta(a, b)$ between two points a and b in games A and B is the vector from b to a weighted by $\theta_{a,b}$, as shown in Equations 6.

$$\delta(a, b) := \theta_{a,b} \|a - b\| \tag{6}$$

In our application, we model only asset deployments. We say that each deployment a is a point in $\mathbb{R}^3$ consisting of the time of deployment and the physical location of the deployment on the $\mathbb{R}^2$ surface of the world. Furthermore, we state that there are two characteristics defining each asset: radar-based and infrared-based (IR-based). Each asset can have radar-based and IR-based defensive characteristics. Our application-specific definition of $\theta_{a,b}$ is shown in Equation 7.

$$\theta_{a,b} := \begin{cases} 1, & \text{if } a \text{ and } b \text{ are both exclusively IR-based,} \\ & \quad \text{RCS-based, or IR- and RCS-based} \\ 10, & \text{if } a \text{ is both IR-based and radar-based} \\ & \quad \text{while } b \text{ is exclusively only IR-based or} \\ & \quad \text{radar-based (or vice versa with respect to} \\ & \quad a \text{ and } b) \\ 100, & \text{if } a \text{ is exclusively radar-based and } b \text{ is} \\ & \quad \text{exclusively IR-based (or vice versa with} \\ & \quad \text{respect to } a \text{ and } b) \end{cases} \tag{7}$$

With these modifications, we fully define our criteria for k-medoids clustering in Equation 8.

$$\begin{aligned} & \arg\min_{\boldsymbol{S},\boldsymbol{M}} \sum_{j=1}^{k} \sum_{x_i \in S_j} H(A, B) \\ & = \arg\min_{\boldsymbol{S},\boldsymbol{M}} \sum_{j=1}^{k} \sum_{x_i \in S_j} \max\{h(A, B), h(B, A)\} \\ & = \arg\min_{\boldsymbol{S},\boldsymbol{M}} \sum_{j=1}^{k} \sum_{x_i \in S_j} \max\left\{\max_{a \in A} \min_{b \in B} \delta(a, b), \max_{b \in B} \min_{a \in A} \delta(b, a)\right\} \\ & = \arg\min_{\boldsymbol{S},\boldsymbol{M}} \sum_{j=1}^{k} \sum_{x_i \in S_j} \max\left\{\max_{a \in A} \min_{b \in B} \left(\theta_{a,b} \|a - b\|\right), \right. \\ & \qquad \left. \max_{b \in B} \min_{a \in A} \left(\theta_{b,a} \|b - a\|\right)\right\} \end{aligned} \tag{8}$$

To perform k-medoids clustering, we again perform EM; however, we now are solving Equation 8 as opposed to Equation 1.

A. Results: Player Tactics

We apply a k-medoids clustering algorithm on data of games played. We specifically investigate games played on the DPE level. This level consists of anti-ship missiles launched from a wide range of bearings, which means that players must account for threats coming from essentially all directions. Further, this level changes each day; the type of each missile is a random variable. Thus, players must either 1) develop a strategy robust to varying missile deployments or 2) adapt their strategy in real time. Because the time pressure is so great, we find that developing a robust strategy is a more viable option. As such, we believe that analyzing this level can provide us with general tactics rather than point-solutions.

As with clustering players, we perform an a priori analysis to determine the optimal number of clusters, k, which we set to $k = 4$. Player tactics are shown in Figure 4. We calculate the average and standard deviation of the scores of games in each cluster post hoc. Scores are normalized linearly to fit a range of $[0, 100]$. We find that the unique game tactics we discover also have statistically significantly different efficacies. The left-most cluster in Figure 4 (M $=$ 74, SD = 27) is statistically significantly better than the center-right (M $=$ 65, SD = 25), $t(81, 112) = 2.40$, $p < 0.05$, and far-right clusters (M $=$ 61, SD = 32), $t(81, 28) = 2.1, p < 0.05$. Further, the center-left cluster (M $=$ 71, SD = 27) is statistically significantly better than the far-right cluster, $t(86, 28) = 1.7$, $p < 0.05$.

Our analysis shows that the single best cluster involves a symmetric deployment of persistent soft-kill weapons (i.e., soft-kill weapons that do not lose effectiveness), such as one mobile persistent and two fixed persistent decoys early in the game to form an "iron triangle," which works well to defeat missiles seeking targets with large radar cross-sections incoming on any bearing. To counter heat-seeking missiles, players in this cluster judiciously deploy flares, which last a short duration, at bearings effective to defeat those incoming missiles. However, players in the far-right cluster are not judicious with their deployments and deploy decoys at bearings that do not safely seduce missiles away from the ship. This k-medoid analysis shows that the players who perform the worst did not develop an intuition for how to reason about the geometrical structure of the problem nor an understanding of which decoys would work against which threats.

VII. Predicting Player Performance and Feature Learning

Predicting how well a player will perform is an important way to determine whether players are ready to move on to more difficult levels or identify weaknesses before those weaknesses are exploited in combat. The conventional method for assessing player capability is through a barrage of tests – essentially a verification and validation process for human operators. However, this process is lengthy, and, if weaknesses are identified, further training will be prescribed. In turn, trainees may need to go through a continuing cycle of training and testing. If an educator were equipped with a tool that could predict areas of weakness during the training process, that educator could reduce the time required to pinpoint weakness and expedite the training process. Further, if that algorithm could identify the key feature for an educator to monitor, that educator could more efficiently use his or her attention.

We construct a machine learning algorithm to predict player performance, which we show in Figure 5. The algorithm, **predictPlayerPerformance**(), takes as input as a set of training data (i.e., examples and labels), testing data (i.e., examples), the number of folds, numFolds, to use during training, and the number of distinct values to evaluate during training for the shrinkage parameter, λ. Our approach combines a feature selection subroutine [37] as well as LASSO regression [46] to learn a model for predicting player performance.

In Line 1, the algorithm calls the feature selection subroutine. Within machine learning, there are such techniques for learning which features are most important. In general, if one has m features, one must search $O(2^m)$ possible combinations of those features to determine which features are the best to employ in prediction. This exponential search space is intractable, especially if one is using a polynomial kernel. Instead, we use a greedy, polynomial-time approximation algorithm [37]. Other such methods exist and could be substituted [22], [24], [27], [35], [50].

In essence, the iterative, sequential, feature-selection algorithm works in two steps. First, after an one-time initialization step that trains the model on a subset of the features, the algorithm adds the one unused features that most improves prediction accuracy (if one exists). Second, the algorithm removes one already-incorporated feature that does not decrease prediction accuracy with its removal (if one exists). The algorithm terminates once the feature set converges [37].

In Line 2, our prediction algorithm prunes features from X^{train},X^{test} not in our feature set F. In Line 3, we select a shrinkage parameter, $\hat{\lambda}$, which has the lowest three-fold cross-validation error on the training data. To estimate the optimal shrinkage parameter, we enumerate a set of possible values for λ, perform three-fold cross-validation using LASSO regression, and select the λ value for which the lowest cross validation error is achieved. Line 4 trains a regression model on the pruned training data and best shrinkage parameter value, $\hat{\lambda}$, (Line 4). Lastly, we predict (Line 5) and return (Line 6) how well we expect players from X^{train} to perform. We note that we use LASSO regression here, but other regression algorithms (e.g., Ridge Regression, Regression Trees, etc.) would be suitable as well.

A. Results: Predicting Player Performance

We report the results of two investigations: 1) The accuracy when predicting whether players will be in the top or bottom 50% of performers using the regression algorithm **predictPlayerPerformance**(), as depicted in Figure 6, and 2) The features that are most helpful predicting the top and bottom performers using our regression algorithm **predictPlayerPerformance**() (Figure 7). For our investigation, we perform leave-one-out cross-validation (LOOCV). We hold one player example as testing data, X^{test}, while we use the examples of the remaining players as training data, X^{train}. We train and test leaving out each player once.

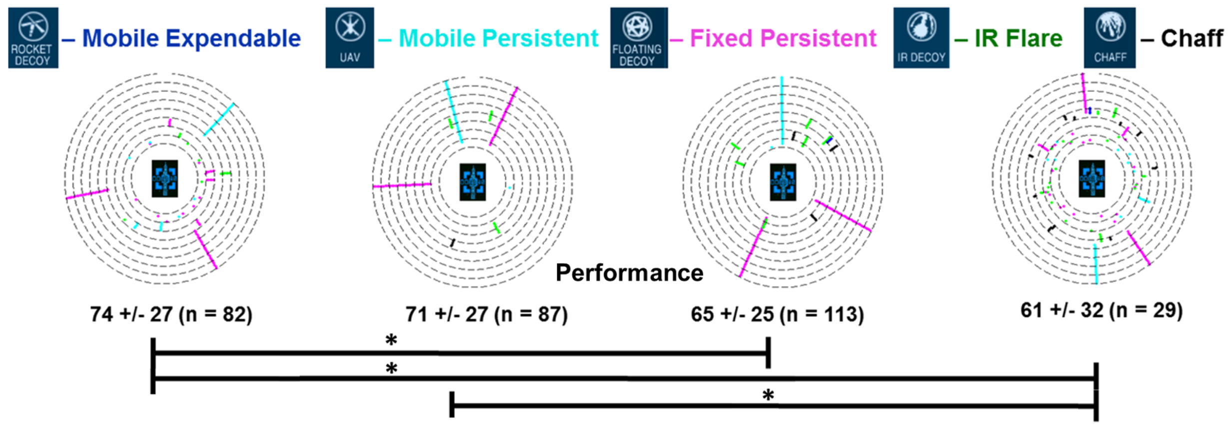

Fig. 4. This figure shows four unique player tactics (i.e., game-cluster medoids) for confronting a scenario with random threat types (i.e., the DPE). The tactics are shown on radial plots such that the bearing of a soft-kill weapon at a given moment in time is shown where time extends radially from the center. For example, a point close to the center corresponds to the existence of a soft-kill weapon early in the game scenario, and a point far from the center corresponds to the existence of a soft-kill weapon late in the game. The mean and standard deviation of the scores for each cluster (normalized to the interval $[0, 100]$) are depicted below each cluster. Horizontal bars with asterisks denotes statistically significant differences between clusters.

predictPlayerPerformance(X^{test},X^{train},Y^{train},... numFolds,numLambdas)

1: $F \leftarrow$**selectBestFeatures**(X^{train},Y^{train})
2: Prune X^{train}, X^{test} such that only features in F remain
3: $\hat{\lambda} \leftarrow$ **selectBestShrinkageParameter**(X^{train},... Y^{train},numFolds,numLambdas)
4: $\hat{\theta} = \arg\min_{\theta} \left(\left(\sum_{x_i \in X^{train}} \left(y_i - \theta^T x_i\right)^2 \right) + \hat{\lambda}\|\theta\|_1 \right)$
5: $\hat{Y} \leftarrow (\hat{\theta}^T X^{train})^T$
6: **return** $\hat{Y}$

Fig. 5. This figure depicts pseudo-code for approximately solving our application of the k-medoids optimization problem in Equation 3 using expectation maximization

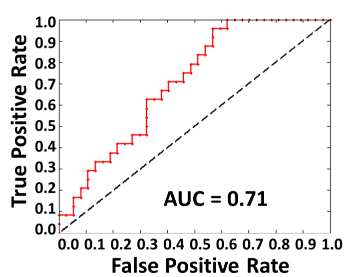

Fig. 6. This figure shows the ROC of the **predictPlayerPerformance**() in Figure 5 evaluated using LOOCV for predicting whether a player will be in the top or bottom 50% of performers.

Figure 6 shows the Receiver Operating Characteristic (ROC) curve for our prediction algorithm **predictPlayerPerformance**() (Figure 5) trained using leave-on-out cross-validation. The model learns information on how to predict player performance, which results in a 63% True Positive Rate and a 33% False Positive Rate (AUC = 0.710). Figure 7 shows a bar graph of how often each feature was selected during LOOCV (Line 1 of Figure 5) by the feature selection subroutine. We can see from Figure 7 that the proportion of games quit relative to the total number of games, the number of unique levels played, and the proportion of unique testing levels attempted are strong indicators of performance.

An initially surprising result is that players who are predicted to perform highly also tend to quit a high proportion of the games they start. This result is in keeping with player types we found through k-means clustering in Section V, shown in Figure 3. We find that players who are the best performers tend to have a higher number of games started, and quit those games frequently. We speculate that this behavior is consistent with a player who would rather restart a level from scratch when a mistake is made rather than continue playing the level. This quitting behavior could indicate determination in achieving perfection, whereas a player who continues playing a level might indicate a state of confusion, indifference, or stubbornness. We note that these results are nuanced: these features are dependent on the specific machine learning algorithm used for prediction and feature selection. It is possible that, with a more or less accurate predictor or a different feature selection algorithm than used in our implementation in Figure 3, the feature importance could change.

Finally, we note that we divided the data into two classes for two reasons. First, we sought to provide an opportunity for a quantitative comparison with our subsequent analysis in Section IX, which examines the predictive power of knowing which sequence of levels a player explored during training. Second, we sought to learn which features were indicative of performance from the set of features shown in Figure 7. If one's goal were to predict the skill level of a player, then one could employ the TrueSkill™ metric [23] or other acceptable metric; however, this was not the case in our investigation.

VIII. Predicting Player Disengagement

In the data set we collected, participants created 226 unique player accounts. Of these 226 accounts, only 146 accounts (65%) played at least one game. After playing only one game, 30 of 146 of those players disengaged, and 20 of the remaining 116 stopped after 2 or 3 games. While having a game that keeps 65% of people interested long enough to play one game

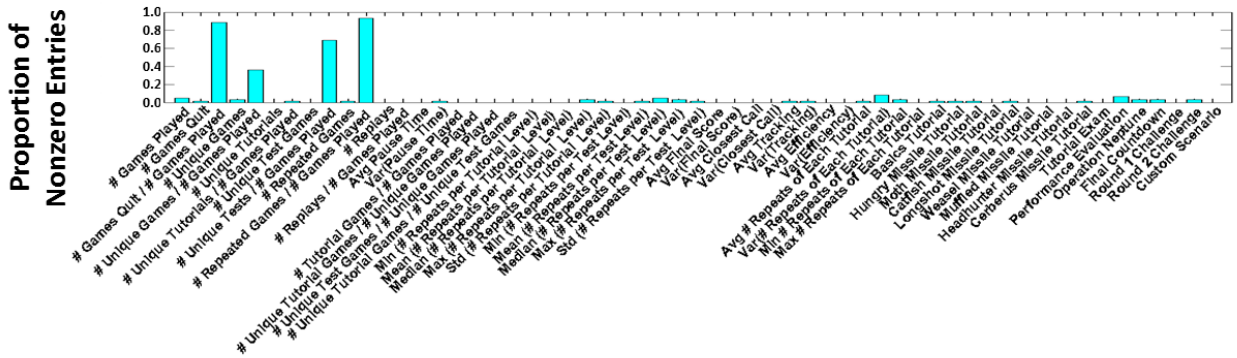

Fig. 7. This figure shows a bar graph depicting how often each feature was found to be informative by the feature selection subroutine (Line 1 of Figure 5).

is a significant improvement over common expectations in industry, we want 100% of warfighters engaged due to the serious importance of the subject matter. As such, we develop a method to identify players who are about to disengage. With such a tool, we believe educators could know when to intervene in a student's education and keep them engaged.

We trained four models: one model each to predict if a player will disengage after 1 game, after 2-5 games, after 6-15 games, and 16-30 games. We binned the players in this manner to provide a sufficient number of positive training examples ($\sim$ 30 examples) for each model. Because we bin the games, we have to be careful in selecting our features. Consider a scenario where a player has started three games and we want to know if the player will quit by the fifth game. All other things being equal, the variance of features describing the performance of a player will decrease as the number of games played increases. As such, if many players quit after only three games, then the model may learn that a player with low variance in the number of games played means that the player is more likely to quit. Similarly, using features describing the raw number of games played would also tend to cause the model to learn that players who have played fewer games would be more likely to quit. Therefore, we must use features that describe the behavior of the player in a way that does not relay to the model the number of games played.

We used the following features: the number of games quit, scenarios played, unique tutorial scenarios played, unique test scenarios played, the number of games in which the player repeated a previously-played scenario, the number of replays, the average scenario score, the average "closest call" (i.e., the minimum distance between an enemy missile and a players ship experienced during the scenario; if a player's ship is hit, the closest call equals zero), and the average efficiency (i.e., players are penalized by the type and number of hard-/soft-kill weapons used to simulate the financial cost of producing these weapons). The feature values for each player were normalized to the total number of games started by that player so that the neural network would be forced to learn to a model that was not parameterized by how many games had been played.

We generate our negative examples (i.e., examples of players who continue playing) by evaluating our features for the first 1, 5, 15, and 30 games started by players who played at least that number of games, respectively. To generate our positive examples (i.e., players who did not continue playing the game), we evaluate the features for all of the games started by players who quit after playing one game, 2-5 games, 6-15 games, and 16-30 games, respectively. For example, if a player started four games, the player would be included as a negative example in the first model and only the first game would be used to evaluate the features. This same player who started four games would then be included as a positive example in the second model, and all four of the player's games would be used to evaluate the features.

We employed a deep neural network to classify whether a player will quit after playing a certain number of games. We construct a neural network with an input and output layer and four hidden layers with twelve, ten, eight, and six neurons each. The hidden layers used a hyperbolic tangent transfer function and the output layer used a sigmoid transfer function. The network was trained using scaled conjugate gradient [31].

We perform leave-one-out cross-validation (LOOCV) to maximize the quality of the learned network and leverage the full power of the data set and report the performance of our models in Figure 8. The neural network is able to predict if a player will quit after playing only one game with almost perfect accuracy. Similarly, we are able to identify approximately 80% of players who will become disengaged after playing 2-5, 6-16, and 16-30 games with only 20% false positive rate. This performance comes even though the total number of training examples decreases by approximately thirty examples in each subsequent interval.

It is perhaps surprising that the performance of the algorithm degrades with each consecutive interval. When predicting whether players would disengage after playing 6-16 games, the features provided to the algorithm should represent a more accurate picture of a player's characteristics versus those features provided to the algorithm after only a single game. Yet, our results in Figure 8 seem to indicate otherwise.

Upon inspection, we can offer at least two potential explanations. First, the total amount of data decreases with each consecutive interval. While the number of positive examples is held approximately constant as players disengage in earlier intervals, there are fewer negative examples in later intervals. Second, predicting whether players drop out in earlier intervals could simply be easier to predict than players who drop out in subsequent intervals. For example, a player who will drop

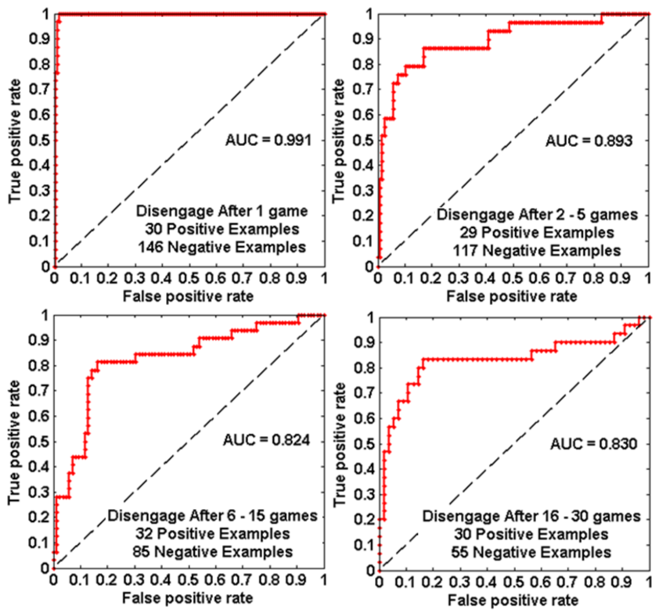

Fig. 8. This figure depicts the ROC for deep neural networks trained to identify players who will disengage after 1 (top left), 2-5 (top right), 6-15 (bottom left), and 16-30 (bottom right) games.

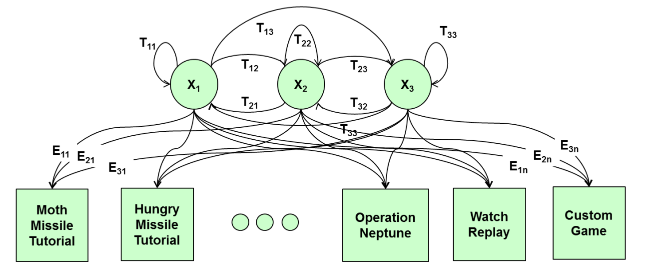

Fig. 9. This figure describes how we formulate the problem of learning how players explore the levels within SGD as a Hidden Markov Model.

out in a later interval may initially act like an engaged player and drop out later due to a frustrating episode of gameplay.

IX. Data-Driven, Personalized Lesson Plans

Developing a lesson plan for the order of tutorials to be played to make training as efficient as possible is a challenging process. In SGD and other games of interest, students are presented a set of tutorial levels that they can play in any order and as often as they like. However, an educator might suspect that the trajectory, or sequence of attempted tutorial levels, would affect how well the student learns the necessary skills to be proficient. Constructing a lesson plan requires a significant effort from educators and domain experts. Instead, we ask whether we could learn what constitutes an effective or ineffective lesson plan with a data-driven approach.

One way to model the underlying process of time-series data (i.e., sequences of tutorial levels) is through a Hidden Markov Model [4]. A HMM is a 5-tuple $\Omega = (X, T, \pi, \Sigma, E)$, where X is the set of hidden states, T is an $|X| \times |X|$ matrix with $T_{i,j} \in T$ being the probability of transitioning from state X_i to state X_j, π is an $|X| \times 1$ vector describing the a priori probability of starting in each state, Σ is the set of emissions, and E an $X \times \Sigma$ matrix with $E_{i,k}$ being the probability of observing emission Σ_k in state X_i.

We model the process of students playing various levels in SGD with a Hidden Markov Model where Σ is the set of levels to be considered. One must then learn the parameters $(S, T, \pi, \Sigma, \lambda)$ that best describes a set of observed emissions from the students. In our application, we want to learn a model that describes a good lesson plan and one for a bad lesson plan. As such, we train two models: one HMM using the example trajectories of the top 50% of players and one HMM using the example trajectories of the bottom 50% of players. We rank players according to their qualifying score for the March Madness Tournament. Our data set included the twenty-six players who completed the necessary levels to be ranked. Figure 9 depicts how we model our problem as an HMM. These HMMs can be used in one of two ways. First, the model serves as a non-parametric (not parameterized by the total number of games) means of predicting whether a new player will be one of the best or worst players given the sequence of tutorial games he or she has played. Second, the HMM is a generative model, meaning that we can generate representative tutorial sequences that a good or bad player might play. With this generative model, we can not only suggest lesson plans for players, we can also customize the lesson plan in real time to recommend the best tutorial to play next to improve the players skill. A key benefit of the HMM is that each set of emissions does not need to be of equal length. Thus, we do not need to trim or align the sequences of players to have the same number of and type played scenarios.

Based on examples of the best and worst performing players in SGD, we trained a model to describe how players navigate the tutorial levels. Specifically, we trained an HMM using the Baum-Welch algorithm. The Baum-Welch requires an initial guess for T and Σ. We initialized T according to a uniform distribution, and we initialized Σ equal to the empirical proportion of observations for each emission uniformly across each hidden state. In other words, $\Sigma_{i,k} = \Sigma_{j,k}, \forall k$ is the proportion of times scenario k was played. We perform LOOCV. Within each iteration of the LOOCV, we perform five-fold cross-validation to learn the HMM. Within each of the five folds, we train fifty HMMs with randomly initialized T and keep the one HMM out of the fifty that has the lowest cross-validation error. We then use the one HMM out of five that performs the best during this five-fold cross-validation to evaluate the one example we left out in the current iteration of the LOOCV. To determine whether a player will be in the top or bottom 50% of performers, we compare the likelihood of the player's behavior being governed by the trained HMMs.

A. Results: Predicting the Value of Training Experience

With our HMM, we can now predict whether a player will be one of the top or bottom performers based not on the player's previous scores, but solely based on the sequence of tutorials he or she has played. We train and test our approach to determine whether a player will be one of the

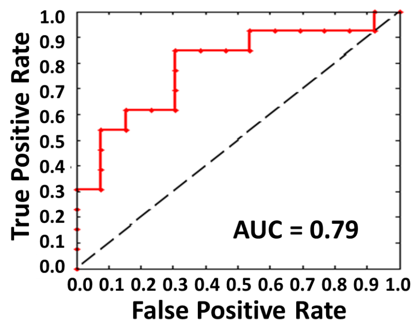

Fig. 10. This figure depicts the ROC for an HMM-based algorithm trained to identify whether a player will be one of the best or worst performers.

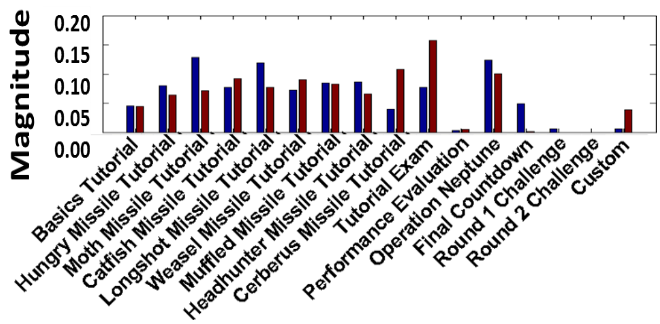

Fig. 11. This figure depicts the frequency with which best players (shown in blue) and worst players (shown in red) played each of the levels in SGD.

top or bottom players according to their ranking on the test levels. The ROC for our approach is shown in Figure 10. As shown in this figure, we are able to predict with high accuracy whether a player will perform well based on the sequence of scenarios played. The performance of this HMM formulation is compelling relative to our prior analysis in Section VII. This analysis shows that the sequence of levels a player explored was a more powerful predictor of performance than the *entire* set of features depicted in Figure 7. Specifically, the HMM-formulation for the sequence of levels explored achieved an AUC $= 0.79$ (Figure 10) whereas the algorithm depicted in Figure 5 achieved an AUC $= 0.71$ (Figure 6).

We also report the distribution of played scenarios for the players in each group in Figure 11. The number of games played of each scenario for the highest- and lowest-scoring players are shown in blue and red, respectively. We perform a two-sample, χ^2 goodness-of-fit test to assess the validity of the null hypothesis that these distributions are the same. We do not reject the null hypothesis as the $\alpha = 0.05$ level ($p \approx 1$). In other words, the proportions of time the highest- and lowest-scoring players spend on each level are statistically indistinguishable. The implication of this finding is that even if the best and worst players play each level with similar frequency, it is the order in which they play those levels that contributes the most to performance.

B. Generating Lesson Plans

In addition to serving as a useful means to predict how well a player will perform based on the sequence of levels he or she played, an HMM can also be used to generate recommendations for which sequence of levels player could explore. Specifically, an HMM is a generative model, which means that it can determine how well a sequence of emissions fits the model and it can generate a representative sequence of emissions that is likely to be seen while observing the process modeled by the HMM. With this capability, we can both prescribe lesson plans a priori (i.e., before the player ever starts) and recommend the best (or worst) tutorial scenario to play next. For example, after a player has played zero, one, or more scenarios, we can provide a tailored recommendation as to which tutorial that player should explore next to improve his or her proficiency. We show an example of a scenario trajectory generated from HMMs trained on the best- and worst-performing players below. We note that we forced the models to start with the first emission as the "Basics Tutorial," which is the first tutorial level, to demonstrate the model differences, and that also we set the length of the trajectory to ten levels. In future work, we will conduct a human-subject experiment to test the efficacy of using an HMM to prescribe which scenarios players should explore and in what order.

Sequence Generated from HMM trained on Bottom 50% of Players	Sequence Generated from HMM trained on Top 50% of Players
1) Basics Tutorial	1) Basics Tutorial
2) Hungry Missile Tutorial	2) Operation Neptune
3) Basics Tutorial	3) Performance Evaluation
4) Hungry Missile Tutorial	4) Operation Neptune
5) Hungry Missile Tutorial	5) Operation Neptune
6) Hungry Missile Tutorial	6) Basics Tutorial
7) Basics Tutorial	7) Moth Missile Tutorial
8) Basics Tutorial	8) Replay
9) Basics Tutorial	9) Hungry Missile Tutorial
10) Hungry Missile Tutorial	10) Hungry Missile Tutorial

Fig. 12. This figure depicts two randomly sampled lesson plans generated using HMMs trained on sequences of levels completed by the top and bottom 50% of players, respectively.

X. Conclusion

Machine learning and serious gaming is opening up the possibility of developing automated teaching aids for improving the training of professionals. In this paper, we demonstrate a variety of machine learning techniques using serious-gaming platform developed to train navy professionals tactics in anti-ship missile defense. We utilize unsupervised learning techniques to discover what types of player behaviors exist and what tactics players develop to defeat anti-ship missile raids. Next, we show how one can employ supervised learning to identify effective player features and behaviors, identify players likely to disengage from training, and how to train a generative model to recommend lesson plans to improve player performance. We believe this suite of machine learning techniques could provide educators with helpful insight in their quest to better educate students. This insight could be used to construct better lesson plans and tailor guidance for individual students' learning styles and behaviors. In future work, we propose working with educators to test the benefit of using our suite of machine learning algorithms to better educate students.

XI. Acknowledgements

We want to thank Commander Scott Orosz, USA, Retired, Deputy Director of Electronic Warfare Programs for the US

Navy (PMR-51), for the generous support he has provided us in our research. We also thank Jessica Stigile for being an invaluable sounding board. This work was sponsored by ONR PMR-51. SGD was developed in collaboration with Pipeworks Inc. and MetaTeq Inc. under PMR-51.

References

[1] *An agent-based recommender system for lesson plan sequencing*, 2002.

[2] Ryan SJD Baker and Kalina Yacef. The state of educational data mining in 2009: A review and future visions. *JEDM*, 1(1):3–17, 2009.

[3] C. Bauckhage, K. Kersting, R. Sifa, C. Thurau, A. Drachen, and A. Canossa. How players lose interest in playing a game: An empirical study based on distributions of total playing times. In *CIG*, pages 139–146, Sept 2012.

[4] Leonard E. Baum and Ted Petrie. Statistical inference for probabilistic functions of finite state markov chains. *Ann. Math. Statist.*, 37(6):1554–1563, 12 1966.

[5] Joseph E Beck and Jack Mostow. How who should practice: Using learning decomposition to evaluate the efficacy of different types of practice for different types of students. In *Intelligent tutoring systems*, pages 353–362. Springer, 2008.

[6] Joseph E Beck and Beverly Park Woolf. High-level student modeling with machine learning. In *Intelligent tutoring systems*, pages 584–593. Springer, 2000.

[7] Leo Breiman, Jerome Friedman, Charles J Stone, and Richard A Olshen. *Classification and regression trees*. CRC press, 1984.

[8] Ross Brooker, David Kirk, Sandy Braiuka, and Aarjon Bransgrove. Implementing a game sense approach to teaching junior high school basketball in a naturalistic setting. *European Physical Education Review*, 6(1):7–26, 2000.

[9] Eric Butler, Erik Andersen, Adam M Smith, Sumit Gulwani, and Zoran Popović. Automatic game progression design through analysis of solution features. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pages 2407–2416. ACM, 2015.

[10] Eric Butler, Adam M Smith, Yun-En Liu, and Zoran Popovic. A mixed-initiative tool for designing level progressions in games. In *Proceedings of the 26th annual ACM symposium on User interface software and technology*, pages 377–386. ACM, 2013.

[11] Rodney A. Clifton, David Mandzuk, and Lance W. Roberts. The alienation of undergraduate education students: a case study of a canadian university. *Journal of Education for Teaching*, 20(2):179–192, 1994.

[12] Mihaela Cocea, Arnon Hershkovitz, and Ryan S.J.D. Baker. The impact of off-task and gaming behaviors on learning: immediate or aggregate? In *Proc. AIED*, pages 507–514. IOS Press, 2009.

[13] Gerben W Dekker, Mykola Pechenizkiy, and Jan M Vleeshouwers. Predicting students drop out: A case study. *International Working Group on Educational Data Mining*, 2009.

[14] Rollande Deslandes, gide Royer, Pierre Potvin, and Danielle Leclerc. Patterns of home and school partnership for general and special education students at the secondary level. *Exceptional Children*, 65(4):496–506, 1999.

[15] John M. Digman. Personality structure: Emergence of the five-factor model. *Annual Review of Psychology*, 41(1):417–440, 1990.

[16] Sidney K Dmello, Scotty D Craig, Amy Witherspoon, Bethany Mcdaniel, and Arthur Graesser. Automatic detection of learners affect from conversational cues. *User modeling and user-adapted interaction*, 18(1-2):45–80, 2008.

[17] A. Drachen, R. Sifa, C. Bauckhage, and C. Thurau. Guns, swords and data: Clustering of player behavior in computer games in the wild. In *CIG*, pages 163–170, Sept 2012.

[18] Arpad E Elo. *The rating of chessplayers, past and present*. Arco Pub., 1978.

[19] Corrado Grappiolo, Yun-Gyung Cheong, Julian Togelius, Rilla Khaled, and Georgios N Yannakakis. Towards player adaptivity in a serious game for conflict resolution. In *Games and Virtual Worlds for Serious Applications (VS-GAMES), 2011 Third International Conference on*, pages 192–198. IEEE, 2011.

[20] Steven S. Graunke and Sherry A. Woosley. An exploration of the factors that affect the academic success of college sophomores. *College Student Journal*, 39(2):367 – 376, 2005.

[21] David C Griffey and Lynn Dale Housner. Differences between experienced and inexperienced teachers' planning decisions, interactions, student engagement, and instructional climate. *Research Quarterly for Exercise and Sport*, 62(2):196–204, 1991.

[22] Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *JMLR*, 3:1157–1182, 2003.

[23] Ralf Herbrich, Tom Minka, and Thore Graepel. Trueskill$^{\text{TM}}$: a bayesian skill rating system. In *Proceedings of the 19th International Conference on Neural Information Processing Systems*, pages 569–576. MIT Press, 2006.

[24] Anil Jain and Douglas Zongker. Feature selection: Evaluation, application, and small sample performance. *TPAMI*, 19(2):153–158, 1997.

[25] Leonard Kaufman and Peter J Rousseeuw. *Finding groups in data: an introduction to cluster analysis*, volume 344. John Wiley & Sons, 2009.

[26] Hyun-Tae Kim and Kyung-Joong Kim. Learning to recommend game contents for real-time strategy gamers. In *CIG*, pages 1–8, Aug 2014.

[27] Kenji Kira and Larry A Rendell. A practical approach to feature selection. In *Proc. International Workshop on Machine learning*, pages 249–256, 1992.

[28] James et al. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proc. Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA., 1967.

[29] T. Mahlmann, A. Drachen, J. Togelius, A. Canossa, and G.N. Yannakakis. Predicting player behavior in tomb raider: Underworld. In *CIG*, pages 178–185, Aug 2010.

[30] Craig Mcinnis. Signs of disengagement? In Jrgen Enders and Oliver Fulton, editors, *Higher Education in a Globalising World*, volume 1 of *Higher Education Dynamics*, pages 175–189. Springer Netherlands, 2002.

[31] Martin Fodslette Møller. A scaled conjugate gradient algorithm for fast supervised learning. *Neural Networks*, 6(4):525 – 533, 1993.

[32] Tood K Moon. The expectation-maximization algorithm. *Signal processing magazine, IEEE*, 13(6):47–60, 1996.

[33] N. Nygren, J. Denzinger, B. Stephenson, and J. Aycock. User-preference-based automated level generation for platform games. In *Computational Intelligence and Games*, pages 55–62, Aug 2011.

[34] Chris Pedersen, Julian Togelius, and Georgios N Yannakakis. Modeling player experience in super mario bros. In *Computational Intelligence and Games, 2009. CIG 2009. IEEE Symposium on*, pages 132–139. IEEE, 2009.

[35] Hanchuan Peng, Fuhui Long, and Chris Ding. Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *Trans. on Pattern Analysis and Machine Intelligence*, 27(8):1226–1238, 2005.

[36] M. Pirovano, R. Mainetti, G. Baud-Bovy, P.L. Lanzi, and N. A. Borghese. Self-adaptive games for rehabilitation at home. In *CIG*, pages 179–186, Sept 2012.

[37] P. Pudil, J. Novoviov, and J. Kittler. Floating search methods in feature selection. *Pattern Recognition Letters*, 15(11):1119 – 1125, 1994.

[38] J Ross Quinlan. *C4. 5: programs for machine learning*. Elsevier, 2014.

[39] Cristóbal Romero and Sebastián Ventura. Educational data mining: a review of the state of the art. *SMCC*, 40(6):601–618, 2010.

[40] Cristóbal Romero, Sebastián Ventura, Pedro G Espejo, and César Hervás. Data mining algorithms to classify students. In *EDM*, pages 8–17, 2008.

[41] Peter J Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65, 1987.

[42] T. Sarratt, D.V. Pynadath, and A. Jhala. Converging to a player model in monte-carlo tree search. In *CIG*, pages 1–7, Aug 2014.

[43] N. Shaker, G.N. Yannakakis, and J. Togelius. Feature analysis for modeling game content quality. In *Proc. CIG*, pages 126–133, Aug 2011.

[44] RD Thorpe and DJ Bunker. A changing focus in games teaching. *Physical education in schools*, 2, 1997.

[45] C. Thurau and C. Bauckhage. Analyzing the evolution of social groups in world of warcraft. In *CIG*, pages 170–177, Aug 2010.

[46] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

[47] G. van Lankveld, P. Spronck, J. Van den Herik, and A. Arntz. Games as personality profiling tools. In *CIG*, pages 197–202, Aug 2011.

[48] Ben George Weber, Michael John, Michael Mateas, and Arnav Jhala. Modeling player retention in madden nfl 11. In *IAAI*, 2011.

[49] Peter Werner, Rod Thorpe, and David Bunker. Teaching games for understanding: Evolution of a model. *Journal of Physical Education, Recreation & Dance*, 67(1):28–33, 1996.

[50] Yiming Yang and Jan O Pedersen. A comparative study on feature selection in text categorization. In *ICML*, volume 97, pages 412–420, 1997.