Stacked Calibration of Off-Policy Policy Evaluation for Video Game Matchmaking

Eric Laufer*, Raul Chandias Ferrari*, Li Yao*, Olivier Delalleau† and Yoshua Bengio*
*Dept. IRO, University of Montreal
†Ubisoft Montreal

***Abstract*—We consider an industrial strength application of recommendation systems for video-game matchmaking in which off-policy policy evaluation is important but where standard approaches can hardly be applied. The objective of the policy is to sequentially form teams of players from those waiting to be matched, in such a way as to produce well-balanced matches. Unfortunately, the available training data comes from a policy that is not known perfectly and that is not stochastic, making it impossible to use methods based on importance weights. Furthermore, we observe that when the estimated reward function and the policy are obtained by training from the same off-policy dataset, the policy evaluation using the estimated reward function is biased. We present a simple calibration procedure that is similar to stacked regression and that removes most of the bias, in the experiments we performed. Data collected during beta tests of *Ghost Recon Online*, a first person shooter from Ubisoft, were used for the experiments.**

I. Introduction

Video games have been a fertile area of research for machine learning and artificial intelligence in the past decade. Applications range from automated content generation [1], behavior modelling [2] to game playing [3]. Of particular interest here are ranking and matchmaking algorithms, stemming from chess rating systems, and motivated by studies reporting that an adequate challenge is an important component of player satisfaction and enjoyment [4], [5]. Starting with [6] and [7], these systems were designed to rank professional chess players according to their history of tournament performance, modelling pairwise comparisons between players, inspired by [8]. Further research on the subject led to modelling uncertainty about the player skills, first introduced by [9], and spawning a variety of Bayesian rating systems, the most famous being Microsoft's own TrueSkill [10]. Going further, [11] and [12] presented feature-based approaches, computing player features based on their behavior (and not only the win/lose results of matches) and using them to model inter-player interactions.

Most of these systems are focused on estimating player skill and predicting game outcomes accurately. Here we will address the problem at a slightly higher level, defining matchmaking as the process of assembling and pairing available users into groups in order to maximize some measure of quality should they play together. The design of such a system is a necessary step for the development of an online multiplayer game, which has become a popular and lucrative paradigm for video games. Since changes in the matchmaking system can potentially impact user experience negatively, careful attention must be given to the evaluation and comparison of matchmaking algorithms. Faced with this policy evaluation challenge in the context of an industrial collaboration, we sought to evaluate methods using reinforcement learning [13] and contextual bandits [14] techniques. However, due to practical issues, they could not be applied on the real task: we propose here an evaluation framework for matchmaking policies, which we show to perform as well as these techniques, would it have been possible to use them in practice.

In the next section we will review and formalize the problem of online matchmaking for video games, describing popular algorithms and presenting an alternative approach to learning to predict game outcomes based on Deep Learning [15], [16]. Section III will address offline evaluation of these systems through reinforcement learning methods called off-policy policy evaluation procedures [13], and explain why it is hard to apply them to our task. Section IV will present the matchmaking simulator used in our experiments in order to evaluate the different off-policy policy evaluation methods. Section V discusses a specific bias induced in direct methods to evaluate a policy and proposes a method to alleviate this bias, called *Stacked Calibration*. Section VI presents results obtained with a methodology that we propose to validate off-policy policy evaluation methods, based on a *two-stage simulation*, i.e., with a model trained on real data being used to generate data and evaluate ground truth inside the simulator.

II. Video Game Matchmaking by Deep Learning

A. Matchmaking in Online Video Games

As an online game grows in popularity, so will the number of players and the amount of data collected concerning their performance. With commercially successful games sometimes involving millions of players online, it has become important for game developers to implement good matchmaking to provide an adequate level of challenge, and to create leagues and leaderboards, which is an important component of player satisfaction [1]. Many of these methods are based on the Bradley-Terry model for pairwise comparison [17], [18]. Given a pool of active players, the goal of these approaches is to estimate a value for each player that conceptually represents the player's level or skill. These skill factors are estimated by modelling the probability of game outcomes between competing players as a function of their skill, and fitting this model through the historical data of wins and losses.

The chess rating system ELO does so by modelling player skills as a Gaussian or logistic distribution from which their game performance is sampled. Performance is measured by a single random scalar, and skill is its expected value. By computing the probability that one player's performance exceeds that of another, ELO provides an estimated score, or outcome

of a match. When new matches are played out, the true and estimated scores are used to update the skill estimate. Several drawbacks of the model such as tracking the uncertainty about players' skills led to the creation of other, more sophisticated, systems which dealt with this uncertainty. Perhaps the most famous recent model proposed is Microsoft's TrueSkill [10], described as a Bayesian skill rating system, that models belief about players' skill, that explicitly models draws and that can also be used with multiple teams of players. The full description of these models is outside of the scope of this paper, and can be retrieved from [7] and [10]. Although chess ratings have initially mostly been used to rank professional players, many video games make extensive use of rating systems for matching players together in online play. Under the assumption that a balanced match has an increased probability of being a draw, the match quality criterion defined by [10] is, in essence, the probability of such an outcome. The goal is thus to launch matches for which the system believes the probability of draw is high.

There has been some interest in matching players to optimize criteria different from draw probability or similarity of skill values, some based on reducing undesired social behavior [12] or matching players to fulfill certain roles through player modelling [19]. In most video games, we observe other outcomes besides the winning team (such as game-defined scores, which can be used in TrueSkill to rank individual players): such outcomes may provide a more objective view of balance. By using a real-valued target based on these outcomes, instead of a binary value (the identity of the winning team), we can obtain more information about the actual balance of each match.

B. Deep Learning Approach to Matchmaking

1) Evaluating the Balance of a Match: We have collected data from *Ghost Recon Online*, an online team-based first person shooting (FPS) game developed by Ubisoft. In this game, players form two teams to compete with each other to achieve a certain goal. The enjoyment of players is affected not only by the design of the game but also by the way the teams are formed. It is important to form teams such that the overall game is balanced — the two teams should have more or less an equal chance to win, or in other words, the combined performance of players from the two teams should be similar. TrueSkill and many other skill-based systems estimate each player's skill by only taking into account the binary result of previous games, and each team's skill by summing the skills of the team members. However, it is natural to assume that the match balance in FPS games strongly depends on the *interaction between players* — how they cooperate as a team. We take advantage of data that could be used to take these interactions and complementarities into account. Furthermore, instead of relying only on the binary win/lose outcome of each game, we use a more fine-grained measurement that is appropriate for FPS games: the relative number of kills made by each team. We apply machine learning methods to estimate this estimator of balance by taking into account the players' performance and their interactions during games played in the past. Note however that the following procedures and experiments can be easily adapted to any other measurement of balance.

Following the approach described in [11], we use the following three steps to form a feature vector summarizing both players and their team performance in a match. First, various statistics of the players' performance were collected during gameplays, such as the number of kills/deaths, wins/losses, firing accuracy, time under cover and other relevant game-dependent measures. These normalized statistics are used to form player profiles. Secondly, for each of the two teams in a match, player profiles are then aggregated to form what we call a team profile by concatenating the sum, mean, variance, minimum and maximum across each individual component of the player profile in that team. Finally, two team profiles are concatenated together to form the final feature vector of a match, denoted by $\mathbf{x}$. We define the measure of balance as

$$y = 1 - \left| log\left(\frac{kills_A}{kills_B}\right)\right|$$

where $kills_A$ (resp. $kills_B$) denotes the total number of kills from players in team A (resp. team B). The logarithm is used to reduce the effect of outliers, and the absolute value makes this quantity symmetric with respect to both teams. It is subtracted from 1 simply so that balance is perfect when equal to one, and to make finding the most balanced match a reward maximization problem. Balance evaluation models $f(\mathbf{x})$ are trained to minimize the mean square error cost between the model's prediction $f(\mathbf{x})$ and the observed target y

$$E_{(\mathbf{x},y)\sim\mathcal{D}}[(y - f(\mathbf{x}))^2] \tag{1}$$

where $\mathbf{x}$ and y are drawn from the unknown distribution $\mathcal{D}$. In practice, the generalization error defined in Eq. 1 is inaccessible due to the unknown $\mathcal{D}$. So instead, given a collection of N historical matches $(\mathbf{x}_i, y_i)$ where $i = 1 \dots N$, the following empirical cost (mean squared error) is minimized instead:

$$\frac{1}{N}\sum_{i=1}^{N}(y_i - f(\mathbf{x}_i))^2. \tag{2}$$

2) Deep Learning and Maxout: The learning algorithm we have found to work best on this task is a deep neural network. Deep Learning [20] has become a field in itself in the machine learning community, with regular workshops at the leading conferences such as NIPS and ICML, and a new conference dedicated to it, ICLR[1], sometimes under the header of *Representation Learning* or *Feature Learning*. The rapid increase in scientific activity on representation learning has been accompanied and nourished by a remarkable string of empirical successes both in academia and in industry, e.g., in speech recognition [21], [22], [23], [24], [25], music information retrieval [26], [27], object recognition [28], [29], [30] and natural language processing [31], [32]. See [15] for a recent review.

Our model is very similar to Maxout networks [16], recently introduced to train deep neural networks in the context of object recognition, but is adjusted to the regression task at hand and applied to matchmaking. It is compared with a standard Multilayer Artificial Neural Network (MLP) that has one or more hidden layers (with hyperbolic tangent non-linearity), each fully connected to the layer below.

[1] International Conference on Learning Representations

Maxout networks use a different type of hidden layer where the hidden units are formed by taking the maximum of k filter outputs (each filter outputs the dot product of a filter weight vector with the input vector into the layer). Since this is a regression problem, the output layer is an affine transformation with a single linear output unit:

$$f(\mathbf{x}) = w'\phi(\mathbf{x}) + b \tag{3}$$

where $\phi(\mathbf{x})$ is the output of the top hidden layer which nonlinearly depends on the neural net input $\mathbf{x}$, w the output weight vector and b is a bias term. Our model can discover and exploit a strong nonlinear dependency between $\mathbf{x}$ and target y.

The training of Maxout networks uses the standard back-propagation equipped with a recently proposed trick called *Dropout* [33]. The basic idea of Dropout is that hidden units outputs are randomly masked (set to 0) with probability $1/2$ during training (and multiplied by $1/2$ at test time). As argued by [16], it is possible to train a deep Maxout network without overfitting due to the model averaging effect brought by Dropout. Meanwhile, by making sure that exactly one filter is being used and receiving gradient for each non-masked hidden unit, it appears that optimizing Maxout networks (even deeper ones) can be done more easily than with rectifier, hyperbolic tangent or sigmoid activation functions. Although Maxout activation functions can easily lead to overfitting, combining them with Dropout provides a form of symmetry breaking and bagging regularization that generally provides substantially better performance than standard MLPs.

To compare, we also consider other standard machine learning models for the regression task. The simplest model is the Elastic Net [34]. It assumes that the interaction between the inputs and outputs are linear. This assumption is usually too strong to hold in practice. To relax this strong assumption, we consider regression trees [35], which assume that the interaction is piece-wise linear, thus overall nonlinear. This assumption works well when the input space can be separated by cells inside which the outputs are similar. However, it suffers when the true interaction is a highly variable but locally smooth function in a high dimensional space. In order to improve over an individual tree, many types of ensemble models have been tried, such as Random Forests [36] and Gradient Boosted Trees [37]. The tree-based ensemble models work well in that they use the composition of multiple weaker learners to reduce the variance of predictions. All these methods are publicly available in scikit-learn[2].

C. Comparative Results

Our dataset consists of 436323 matches played by 159142 players. For each match, we compute a feature vector of dimension 620, the first half of which represents team A's attributes and the second half team B's attributes, as discussed in section II-B. The entire dataset is chronologically split in three sets: 70% as the training set, 15% as the validation set and 15% as the test set. We train the standard MLP and Maxout network on the training set and perform early stopping by checking the cost in Eq. 2 on the validation set. When there is no significant drop (0.0001) for three consecutive training passes through the training set, training is stopped. For Elastic Net, training optimization is stopped when the change in cost does not exceed 0.0001. For tree-based models, training is stopped when the maximal tree depth is reached. The best model is selected based on the validation cost. We report the average cost on the test set as a measurement of models' performance in generalization.

In order to compare with different types of models, we have performed intensive hyper-parameter search for each type. We summarize the most important hyper-parameters of each model family and give the best values found. For the Elastic Net, α controls the strength of both L1 and L2 regularizations and β balances the relative importance between them. We have found that $\alpha = 1.16 \times 10^{-6}$ and $\beta = 1.42 \times 10^{-6}$ work best. For the Random Forest and Gradient Boosted Trees, the most important hyper-parameters are the number of base learners and their depth. The best model we have found for the Random Forest uses 146 base learners each of which has a depth of 16. The best Gradient Boosted Tree has 388 base learners each of which has a depth of 12. For standard MLPs and Maxout networks, there are typically many more hyper-parameters that are important such as the learning rates, the number of hidden layers and number of units per hidden layer. The best Maxout network we have found is with $k = 2$ filters per units and 2 hidden layers with 1929 hidden units for the first and 621 units for the second. The best standard MLP we found had three hidden layers with 742, 911, and 964 hidden units respectively.

Figure 1 shows the mean squared error (MSE) of each model as a measurement of their performance. As a baseline, the constant predictor using the average of y in the training set performs the worst, with the highest MSE. MLP performs the same as Random Forest. Maxout performs the best and has the lowest MSE. Table II-C shows the resulting p-values of the paired t-tests. It shows the significance of the differences between all models' MSE. The standard $p = 0.05$ is adopted to indicate the statistical significance. The bold font indicates that the difference of MSE between two models is statistically significant. In fact, all MSEs are significantly different from each other, except between Random Forest and MLP.

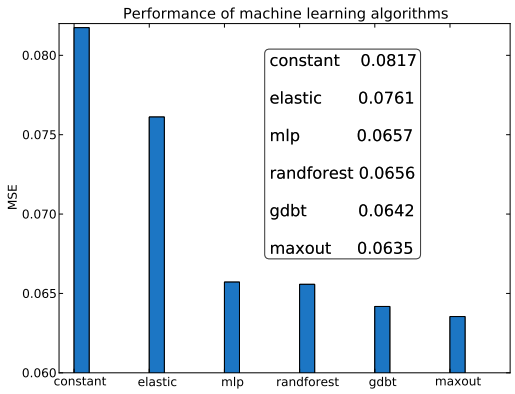

Fig. 1. Performance of different types of models in predicting the balance of a match (based on the kill ratio). *constant* for the mean predictor as a baseline, *elastic* for Elastic Net, *MLP* for a standard MLP, *randforest* for Random Forest, *gdbt* for Gradient Boosted Trees, *Maxout* for Maxout networks. Maxout outperforms all the other models. See also Table II-C for the significance tests.

III. Off-Policy Policy Evaluation

There is a class of problems in reinforcement learning named contextual bandits, also called bandits with side in-

[2]http://scikit-learn.org

TABLE I. P-VALUES OF PAIRED T-TEST ON MSE BETWEEN MODELS. These show if the difference in performance between any pair of models is statistically significant (in bold), i.e. p-value < 0.05, or not. All differences are significant except between RandForest and MLP, which are statistically indistinguishable. Maxout is found superior to each of the other models in a statistically significant way. See Fig. 1 for the labels of each of the methods used in the table below.

	MAXOUT	GDBT	RANDFOREST	MLP	ELASTIC
MAXOUT		**0.001**	$\ll \mathbf{10^{-3}}$	$\ll \mathbf{10^{-3}}$	$\ll \mathbf{10^{-3}}$
GDBT	**0.001**		$\ll \mathbf{10^{-3}}$	$\ll \mathbf{10^{-3}}$	$\ll \mathbf{10^{-3}}$
RANDFOREST	$\ll \mathbf{10^{-3}}$	$\ll \mathbf{10^{-3}}$		0.532	$\ll \mathbf{10^{-3}}$
MLP	$\ll \mathbf{10^{-3}}$	$\ll \mathbf{10^{-3}}$	0.532		$\ll \mathbf{10^{-3}}$
ELASTIC	$\ll \mathbf{10^{-3}}$	$\ll \mathbf{10^{-3}}$	$\ll \mathbf{10^{-3}}$	$\ll \mathbf{10^{-3}}$	

formation, that provide a framework for learning models and evaluating algorithms on partially labeled data [38], [14]. Such data is generated in the following fashion: given a context X, a vectorial representation of the state we are currently in, we sample an action m from a policy $\pi(X)$ that defines a multinomial distribution over K available actions. Afterwards, action m is effectively taken, and a reward y is observed, sampled from the real-world distribution of $p(y|m, X)$. The core of the problem is that $\pi(X)$ has a crucial impact on the data being collected: we will never observe the rewards associated with actions that were not taken (unless the exact same context occurs twice). This formulation lends itself well to the analysis and optimization of web ad campaigns [39], and is one of most popular frameworks for the task. Just like we will never know if a user would have clicked or not on an ad that was never shown, we will never know the outcome of a match until the players actually play it out. In our case, the decision of which players to match is taken by the matchmaking algorithm, which can thus be considered as a policy.

Here, m will denote a match, and X the set of available matches for selection, along with the associated player profiles. For a given context X (available matches), a matchmaking policy will choose one match to be launched. If $y_m \sim p(y_m|m, X)$ is the objective measure of balance given by the outcome of match m, the policy value is defined as:

$$V^\pi = \mathop{\mathbf{E}}_{X\sim D, m\sim\pi(X)}[y_m|m, X] \tag{4}$$

where D is the distribution of contexts, containing both the available matches and the corresponding player profiles, $m \sim \pi(X)$ is the match (selected according to policy π), and $p(y_m|m, X)$ is the underlying conditional distribution of rewards given a match and the associated player profiles. Assuming that higher values of the measure of balance are better, finding the optimal policy is formulated as estimating $\mathrm{argmax}_\pi V^\pi$.

The problem of learning on data collected by a (potentially unknown) policy is known as off-policy learning; the problem of evaluating a policy on data collected using a different policy is known as off-policy policy evaluation. Both have been well-studied for Markov decision processes [13], but are also applied in web applications such as advertising and predicting browsing behavior.

We argue that video game matchmaking suffers from these two problems, in the case where we want to replace a policy that is currently in effect and has been used to generate historical data. Suppose we fit (as in Section II) a regression model $f(m, X) \approx \mathbf{E}[y_m|m, X]$ by minimizing the squared error between targets y_m and predicted values $f(m, X)$ over S_{train}, a subset of the dataset S comprised of triplets (X, m, y_m), generated by the policy b (called behavior policy). One might be tempted to use this approximation to estimate the value of a different policy π, through:

$$\hat{V}^\pi = \frac{1}{|S_{test}|} \sum_{X\in S_{test}, m\sim\pi(X)} f(m, X), \tag{5}$$

where S_{test} is the testing subset of S, non-overlapping with S_{train}. However, since $f(m, X)$ was trained to approximate the rewards associated with actions chosen by the behavior instead of π, this estimator (called "Direct Method") tends to suffer from strong bias, as reported by [40]. An example of this bias would be a model trained on a dataset generated by a very good or a very poor behavior policy, in which case the model's average prediction would be, respectively, very large or very small. Naturally, the optimal way of evaluating a policy is to run it live and observe the average reward. This is problematic in practice for risk management concerns, since running a bad policy would negatively impact user experience, and harm the popularity of the game. Because of this, many methods were studied and developed to address this problem.

A popular class of algorithms for offline off-policy policy evaluation are based on importance sampling techniques. Simply put, we can estimate $\mathbf{E}_d[x]$ empirically using values sampled under a distribution different than d , i.e. $x_i \sim d'$, with

$$\hat{x} = \frac{1}{n} \sum_{i=1}^{n} x_i \frac{d(x_i)}{d'(x_i)} \tag{6}$$

where $\hat{x}$ is an unbiased estimator of $\mathbf{E}_d[x]$, if d and d' are known exactly, i.e.,

$$\mathbf{E}[\hat{x}] = \mathbf{E}_d[x]. \tag{7}$$

Note however that importance weighted estimators, even if they are unbiased, may still have a large variance. This can be a significant issue when there are values of x for which $d'(x)$ is near 0 while $d(x) \neq 0$.

In terms of matchmaking, this means that if we know the distributions over actions given the context, $p_b(m|X)$ for the behavior policy b from which data was sampled, and $p_\pi(m|X)$ for the target policy to evaluate π, then we could use importance weights to estimate V^π:

$$V^\pi \approx \hat{V}^\pi = \frac{1}{|S|} \sum_{(X,m,y_m)\in S} y_m \frac{p_\pi(m|X)}{p_b(m|X)} \tag{8}$$

for data collected under policy b. Several approaches are based on importance weights to evaluate the value of a policy, and have been shown to work well in practice when importance weights are known or properly estimated.

However, to use these methods, one must know the distribution $p_b(m|X)$ of the behavior policy, or at least have a good approximation of it. This is hard in the case of matchmaking since the actual action space is enormous because of the combinatorial explosion of possible teams. The only way to actually store available actions and their probability is to have a sampling mechanism which limits the number of

possible matches to evaluate. Although the core model for the matchmaking system of a game can be known, the candidate matches at each context are not necessarily stored, as is the case in the data we were given access to. This makes the estimation of behavior action probabilites impossible, because we do not know which actions were actually available at the time of creation of each match. Not only that, but the behavior policy may actually be deterministic (as in our data), in which case we cannot direcly apply importance weighting, because the variance of the estimator does not exist and the importance sampling estimator becomes meaningless.

This motivates the exploration of techniques such as the Stacked Calibration methods proposed here, which can be applied even when the behavior policy is deterministic, i.e., where methods based on importance weights cannot be used.

In addition, we opt for a simulation-based approach to policy evaluation. Although susceptible to bias, we benefit from the fact that the core matchmaking system is already in place, and this allows us to more properly model the dynamics of matchmaking.

IV. Matchmaking Recommendation Simulator

For reasons mentioned in the previous section, a simulator was built to evaluate different policy models. The simulator can be viewed as a type of single-server waiting queue. Each player arriving in the queue (also known as "lobby") is assigned attributes by randomly selecting them from the set of real player profiles. This is to simulate the distribution of player attributes. Since scoring each and every possible player configuration (i.e., a match) is impossible with more than a few dozen players, we randomly sample configurations of players, which will serve as match candidates. The set of randomly sampled matches corresponds to the context X from the previous section. The match candidates are scored by the policy model, and the highest-scoring match is selected. The corresponding players leave the queue, and new players can be added.

A practical concern of matchmaking systems is that every player must be matched within a certain time constraint, to avoid letting a player wait too much time before being matched. To this end, players who wait more than a certain threshold delay are forced into the next candidate match.

In our experiments, we used 100000 different attribute profiles to sample from. These are real user profiles corresponding to the profiles in the test set defined in section II-C. The initial waiting queue size is 200, and players arrive at a constant rate; although this is not completely reflective of reality, it is a good approximation on a short time scale for most periods of time. A total of 50 random matches were sampled at each iteration, from which the highest scoring is selected according to the policy model. The time constraint imposed is 20 iterations: after 20 iterations without being assigned in a match, a player is forced into the next match. In a real setting, the time constraint would be influenced by the arrival rate of players. With a simulator, we can run a policy, sample matches and generate the corresponding dataset.

V. Stacked Calibration

There remains an issue if we are to evaluate a model policy using the simulator. Using the same expected balance score both to select matches and to estimate matchmaking quality results in a "collusion bias". For each context, the policy must choose a match. Suppose the policy is deterministic, and chooses the match with highest expected value, given the policy model. The true most balanced match has an expected value that cannot exceed that of the match chosen by the policy. Therefore the chosen match is almost always worse than it appears based on the target policy π. Indeed, if

$$\begin{aligned} m^* &= \operatorname*{argmax}_{m \in X} f_{GT}(m) \\ \hat{m} &= \operatorname*{argmax}_{m \in X} f_{DM}(m) \end{aligned}$$

it follows that, for any given choice of matches,

$$f_{DM}(m^*) \leq f_{DM}(\hat{m}),$$

where f_{DM} is our trained estimator[3] and f_{GT} the true expected value (which we call "ground truth"). Although f_{DM} estimates the ground truth, the expected balance score of the match *selected* by the predictor during simulation (or during the live use of the predictor in the field) might still differ from it, because of the inherent limitations of various learning algorithms. This bias comes about because the same model is used *both for estimating value and for selecting an action*. This problem is analogous to the optimistic bias one observes when using training error to estimate generalization error of a learning algorithm: the learner tends to give lower errors on the training examples because these were used to select parameters. Here we have a policy that selects actions according to a predictor, and because these actions were the selected ones, they tend to receive a higher estimated value than their true value.

This motivates the proposed estimator, which we call *Stacked Calibration*. The procedure is simple. We let most of the parameters of the value predictor be estimated on a training set, but we estimate a calibration transformation on top of the behavior of the predictor on a validation set. We conjecture that this helps the predictor not only generalize better in general but also reduce the collusion bias. This procedure is a variant of the principle of *stacking* [41]), by which the outputs of one or more models are computed on examples not used to train them, and these outputs then enter as part of the input in examples to train a second level of models (and this can be repeated at multiple levels).

The specific procedures we used in the experiments are the following. Given the training, validation and test splits of a dataset S, we first train a value estimator $f_{DM} = f$ on the training set, and use the validation set to estimate

$$\tilde{f}(m, X) \approx \mathbf{E}[y_m | f(m, X)],$$

$\tilde{f}$ being the calibrated version of f, by minimizing

$$\sum_{(X, m, y_m) \in S_{valid}} (y_m - \tilde{f}(m, X))^2.$$

We have have tried two variants of calibration. The first is a simple linear least squares model, which we denote LDM,

[3]DM stands for Direct Method [40].

and the corresponding calibrated function. The second is a non-parametric etimator and a slight variant of radial basis function network (RBF), where

$$\tilde{f}(m, X) = \beta + f(m, X) + \sum_i w_i e^{-\frac{(f(m,X)-\mu_i)^2}{\sigma^2}}. \quad (9)$$

Here μ_i are the radial basis function centers and w_i the associated weights, σ^2 is a scaling hyperparameter and β is a bias term. In our experiments we used 100 centers, sampled uniformly from $f(m, X)$'s predictions on the validation set, as to easily cover the one-dimensional input space. The μ_i's were kept fixed and σ set to 1, so that the calibration parameters could be analytically estimated to minimize squared error over the validation set. In experiments we will denote the linear version as f_{LDM}, and the RBF version as f_{RBFDM}

VI. Experimental Results

A. Experimental Setup

To assess the performance of the Stacked Calibration method we need access to the real expected value of the balance score of a match, which we will call the ground truth. The ground truth represents the underlying conditional distribution of a match's outcome given its players. Without it, we cannot conclude that calibration works without live testing, in which case we are back to square one. To access the ground truth's expected value, we use a **two-stage simulation** methodology, in which a ground truth model f_{GT} is first trained from real data and then used to generate data and assess different learning and policy evaluation procedures. The details of this two-stage procedure are shown in Algorithm 1, allowing us to verify that the proposed stacked calibration procedure indeed improves policy value estimates.

Algorithm 1 Experimental procedure for two-stage simulation.

1: Train model f_{GT} on the real data. This model will be the ground truth.
2: Use the simulator under a specified policy b (behavior policy).
3: For the simulated matches, sample the outcome using f_{GT}. These sampled matches and outcomes constitute the artificial dataset S_b, generated by policy b.
4: Train model f_{DM} on the training subset of S_b, and use the simulator with f_{DM} as policy.
5: Using the validation subset of S_b, apply the calibration procedures described in section V to obtain f_{LDM} and f_{RBFDM}.
6: Evaluate the simulated matches using f_{DM}, the policy model, and calibrated models f_{LDM} and f_{RBFDM}.
7: Use f_{GT} to evaluate the true value of all these matches (and hence the estimated value of the policy), and compare this true policy value with the values estimated by the different off-policy policy evaluation methods compared here.

This methodology allows us to verify which policy evaluation method works best, since f_{GT} is known. In practice, the true underlying conditional distribution $p(y_m|m, X)$ is highly complex and difficult to capture completely with statistical models (assuming the task is complex enough), and the best models tend to be overregularized. To prevent models trained on artificial data generated from an overregularized ground truth to estimate the ground truth model too well, in which case directly estimating the matches with the policy model would be satisfactory, we used as ground truth a highly overfitted gradient boosted tree. In addition, Gaussian noise with standard deviation equal to 10% of the ground truth's standard deviation on the artificial data was added: the result is a difficult to capture (because it has a rich structure that is not too smoothed out by regularization), noisy signal, as we would expect to see in real data. The behavior policy we used was a random, uniform policy. The same simulator parameters were used for policy evaluation and data generation.

In order to provide comparison with already existing methods, we saved the action probabilities of the behavior and the associated sampled matches during simulation. This gives us what we need to compare the standard importance sampling method (IS) and doubly robust estimation [40], which is a method which uses both IS and the reward estimated by the f_{DM}. The doubly robust estimator (DR) is defined by:

$$V_{DR}^{\hat{\pi}} = \frac{1}{|S|} \sum_{(X,m,y_m)\in S} \Big[f_{DM}(m, X) + (y_m - f_{DM}(m, X)) \frac{p_\pi(m|X)}{p_b(m|X)} \Big], \quad (10)$$

where S in the above equation represents a validation set not used to estimate f_{DM}. Given the uniform policy behavior, weights $p_b(m|X)$ are known exactly, which allows us to apply this method. In the case where the evaluated policy is deterministic, $p_\pi(m, X)$ becomes the indicator function for the behavior action being the same as the policy action. For a complete review of doubly robust policy evaluation, see [40].

B. Comparative Results

The artificial data was split in 3 sets: 50000 matches as the training set and 25000 matches each for the validation and test sets. We trained a variety of models on the artificial data: linear regressions with L2 weight decay (ridge regression), gradient boosted decision trees, random forests, k-nearest neighbours regression and linear kernel support vector machines. The point of this diversity in learning algorithms is not necessarily to actually compare these different learning algorithms, but to make sure the evaluation works across a broad range of models. We applied Algorithm 1, performing random sampling of the hyperparameters to obtain various models (as variants of the same learning algorithm type), and we report mean squared error across all the models for each different evaluation method. DM signifies that the same function is used for match selection and evaluation. LDM denotes the linear calibration and RBFDM denotes the radial basis function network version, both described in section V, evaluating the matches chosen by the (uncalibrated) policy. IS stands for importance sampling, and DR for the doubly robust method (both described in section III).

Figure 2 shows, as expected, that the uncalibrated direct method is the worst performing one. The other four are considerably better, with RBFDM leading the chart. To ensure that these differences were significant, we computed a paired

t-test comparing each pair of evaluation procedures, with the null hypothesis being that the expected mean squared errors are the same. The p-values for these tests are reported in table III, with p-values smaller than 0.05 in bold (where the null hypothesis is rejected, i.e. indicating a statistically significant difference).

To further verify the policy value estimation, we computed Spearman's rank correlation coefficient between each model's estimated policy value and the ground truth, for each evaluation method (Table II). Again, DM is the worst performing method while not relying on the access to the behavior policy action probabilities. The important thing to note is that RBFDM can select the best model almost as well as DR, showing that model selection is on par with this method. As reference, the test errors of the models give a correlation coefficient of 0.91 with the ground truth value; we can estimate policy value accurately and select the (near) optimal one.

In analyzing these results, keep in mind that the objective was to verify if the stacked calibration methods could approach the performance of methods such as the doubly robust (DR) estimator, while not requiring a known and stochastic behavior model. The results are clearly positive in this respect.

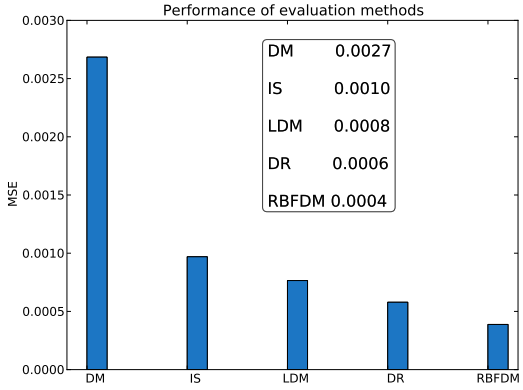

Fig. 2. Precision of value approximation for different evaluation methods: *DM* for the direct uncalibrated method, *IS* for important sampling, *LDM* for linear calibration, *DR* for doubly robust, *RBFDM* for rbf calibration. RBFDM outperforms all the other methods. See also Table III for the significance tests.

TABLE II. SPEARMAN RANK CORRELATION BETWEEN EVALUATION METHODS AND GROUND TRUTH.

	RBFDM	DR	LDM	IS	DM
correlation coefficient	0.9011	0.9366	0.8328	0.611	0.2575
p-value	1.03e-37	6.9e-47	3.6e-27	1.1e-11	0.0093

TABLE III. P VALUES OF PAIRED-T TEST OF SQUARED ERROR BETWEEN EVALUATION METHODS

	GT	RBFDM	DR	LDM	IS	DM
GT		$\ll 10^{-4}$	$\ll 10^{-4}$	$\ll 10^{-4}$	$\ll 10^{-4}$	$\ll 10^{-4}$
RBFDM	$\ll 10^{-4}$		**0.0163**	$\ll 10^{-4}$	$\ll 10^{-4}$	$\ll 10^{-4}$
DR	$\ll 10^{-4}$	**0.0163**		**0.0473**	**0.0011**	$\ll 10^{-4}$
LDM	$\ll 10^{-4}$	$\ll 10^{-4}$	**0.0473**		0.1593	$\ll 10^{-4}$
IS	$\ll 10^{-4}$	$\ll 10^{-4}$	**0.0011**	0.1593		$\ll 10^{-4}$
DM	$\ll 10^{-4}$	$\ll 10^{-4}$	$\ll 10^{-4}$	$\ll 10^{-4}$	$\ll 10^{-4}$	

VII. CONCLUSIONS

In this paper we have reviewed and compared matchmaking procedures. We have shown that modelling player-player interactions through deep architectures is helpful in predicting the outcome of a match, as these interactions are in reality complex and difficult to model with shallower models.

We then reviewed off-policy policy evaluation techniques, and pointed out some practical difficulties encountered in the context of evaluating matchmaking methods in an industrial setting where the behavior policy is unknown or not stochastic. We then proposed a technique called Stacked Calibration to improve off-policy evaluation and tested its effect through a two-stage simulator methodology.

Through these simulations, we verified that Stacked Calibration performs as well as or better than standard offline methods such as the doubly robust estimator, which are not applicable in our industrial setting. We showed that it is a reliable way of evaluating policy value. The whole approach is applicable to any setting where importance weights are not applicable and where simulation is preferred to (or in addition to) A/B testing because A/B testing is expensive, worrisome for product managers concerned with the negative effects of a poor policy, and limited in the number of policies that can be evaluated (only so many different policies can be A/B tested in a given amount of time).

ACKNOWLEDGMENTS

The authors would like to thank Aaron Courville, Myriam Côté and Frédéric Bernard for their collaboration and useful feedback, the NSERC-Ubisoft chair for funding, and the *Ghost Recon Online* team in Ubisoft Singapore for their support throughout this project.

REFERENCES

[1] G.N. Yannakakis and J. Togelius, "Experience-driven procedural content generation," *Affective Computing, IEEE Transactions on*, vol. 2, no. 3, pp. 147–161, 2011.

[2] H Jaap van den Herik, HHLM Donkers, and Pieter HM Spronck, "Opponent modelling and commercial games," *Proceedings of IEEE*, pp. 15–25, 2005.

[3] Johannes Fürnkranz, "Recent advances in machine learning and game playing," *ÖGAI Journal*, vol. 26, no. 2, pp. 19–28, 2007.

[4] Penelope Sweetser and Peta Wyeth, "Gameflow: A model for evaluating player enjoyment in games," *Comput. Entertain.*, vol. 3, no. 3, pp. 3–3, July 2005.

[5] Ben Cowley, Darryl Charles, Michaela Black, and Ray Hickey, "Toward an understanding of flow in video games," *Computers in Entertainment*, vol. 6, no. 2, pp. 20:1–20:27, July 2008.

[6] Kenneth Harkness, *Official Chess Handbook*, McKay, 1967.

[7] A. E. Elo, *The rating of chessplayers, past and present*, Batsford, 1978.

[8] Ralph A. Bradley and Milton E. Terry, "The rank analysis of incomplete block designs — I. The Method of Paired Comparisons," *Biometrika*, vol. 39, pp. 324–345, 1952.

[9] Mark E. Glickman, "Parameter estimation in large dynamic paired comparison experiments," *Applied Statistics*, vol. 48, pp. 377–394, 1999.

[10] Ralf Herbrich, Tom Minka, and Thore Graepel, "Trueskill^{TM}: A bayesian skill rating system," *Advances in Neural Information Processing Systems*, vol. 19, pp. 569, 2007.

[11] Olivier Delalleau, Emile Contal, Eric Thibodeau-Laufer, Raul Chandias Ferrari, Yoshua Bengio, and Frank Zhang, "Beyond skill rating: Advanced matchmaking in ghost recon online," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 4, no. 3, pp. 167–177, Sept. 2012.

[12] Jens Riegelsberger, Scott Counts, Shelly Farnham, and Bruce C. Philips, "Personality matters: Incorporating detailed user attributes and preferences into the matchmaking process," in *HICSS*. 2007, p. 87, IEEE Computer Society.

[13] Doina Precup, Richard S. Sutton, and Satinder Singh, "Eligibility traces for off-policy policy evaluation," in *ICML'2000*, 2000, pp. 759–766.

[14] John Langford and Tong Zhang, "The epoch-greedy algorithm for contextual multi-armed bandits," in *NIPS'2008*, 2008, pp. 1096—1103.

[15] Yoshua Bengio, Aaron Courville, and Pascal Vincent, "Unsupervised feature learning and deep learning: A review and new perspectives," *IEEE Trans. Pattern Analysis and Machine Intelligence (PAMI)*, 2013.

[16] Ian J. Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio, "Maxout networks," in *ICML*, 2013.

[17] Tzu-Kuo Huang, Chih-Jen Lin, and Ruby C. Weng, "A generalized Bradley-Terry model: From group competition to individual skill," in *Advances in Neural Information Processing Systems 17*, Lawrence K. Saul, Yair Weiss, and Léon Bottou, Eds., pp. 601–608. MIT Press, Cambridge, MA, 2005.

[18] Joshua E. Menke and Tony R. Martinez, "A Bradley-Terry artificial neural network model for individual ratings in group competitions.," *Neural Computing and Applications*, vol. 17, pp. 175–186, 2008.

[19] Jorge Jimenez-Rodriguez, Guillermo Jimenez-Diaz, and Belen Diaz-Agudo, "Matchmaking and case-based recommendations," in *Workshop on Case-Based Reasoning for Computer Games, 19th International Conference on Case Based Reasoning*, 2011.

[20] Yoshua Bengio, *Learning deep architectures for AI*, Now Publishers, 2009.

[21] George E. Dahl, Marc'Aurelio Ranzato, Abdel-rahman Mohamed, and Geoffrey E. Hinton, "Phone recognition with the mean-covariance restricted Boltzmann machine," in *NIPS'2010*, 2010.

[22] Frank Seide, Gang Li, and Dong Yu, "Conversational speech transcription using context-dependent deep neural networks," in *Interspeech 2011*, 2011, pp. 437–440.

[23] Abdel-rahman Mohamed, George Dahl, and Geoffrey Hinton, "Acoustic modeling using deep belief networks," *IEEE Trans. on Audio, Speech and Language Processing*, vol. 20, no. 1, pp. 14–22, 2012.

[24] George E. Dahl, Dong Yu, Li Deng, and Alex Acero, "Context-dependent pre-trained deep neural networks for large vocabulary speech recognition," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 1, pp. 33–42, 2012.

[25] Geoffrey Hinton, Li Deng, George E. Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara Sainath, and Brian Kingsbury, "Deep neural networks for acoustic modeling in speech recognition," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, Nov. 2012.

[26] Philippe Hamel, Simon Lemieux, Yoshua Bengio, and Douglas Eck, "Temporal pooling and multiscale learning for automatic annotation and ranking of music audio," in *ISMIR*, 2011.

[27] N. Boulanger-Lewandowski, Y. Bengio, and P. Vincent, "Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription," in *ICML'2012*, 2012.

[28] Dan Ciresan, Ueli Meier, and Juergen Schmidhuber, "Multi-column deep neural networks for image classification," Tech. Rep., arXiv:1202.2745, 2012.

[29] Salah Rifai, Yann Dauphin, Pascal Vincent, Yoshua Bengio, and Xavier Muller, "The manifold tangent classifier," in *NIPS'2011*, 2011.

[30] A. Krizhevsky, I. Sutskever, and G. Hinton, "ImageNet classification with deep convolutional neural networks," in *NIPS'2012*. 2012.

[31] Yoshua Bengio, "Neural net language models," *Scholarpedia*, vol. 3, no. 1, 2008.

[32] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *Journal of Machine Learning Research*, vol. 12, pp. 2493–2537, 2011.

[33] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," Tech. Rep., arXiv:1207.0580, 2012.

[34] Hui Zou and Trevor Hastie, "Regularization and variable selection via the elastic net," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 67, no. 2, pp. 301–320, 2005.

[35] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees*, Wadsworth International Group, Belmont, CA, 1984.

[36] Leo Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[37] J. Friedman, "Greedy function approximation: a gradient boosting machine," *Annals of Statistics*, vol. 29, pp. 1180, 2001.

[38] Peter Auer, Niccolo Cesa-Bianchi, Yoav Freund, and Robert E. Schapire, "The nonstochastic multiarmed bandit problem," *SIAM J. Computing*, vol. 32, no. 1, pp. 48—77, 2002.

[39] Olivier Chapelle and Lihong Li, "An empirical evaluation of Thompson sampling," in *Advances in Neural Information Processing Systems 24 (NIPS'11)*, 2011.

[40] Miroslav Dudik, John Langford, and Lihong Li, "Doubly robust policy evaluation and learning," in *Proceedings of the 28th International Conference on Machine learning*, 2011, ICML '11.

[41] D. H. Wolpert, "Stacked generalization," *Neural Networks*, vol. 5, pp. 241–249, 1992.