Value Functions for Depth-Limited Solving in Imperfect-Information Games

Vojtěch Kovařík[*], **Dominik Seitz**[*], **Viliam Lisý**[†]
Artificial Intelligence Center, FEE,
Czech Technical University in Prague,
Prague, Czech Republic

Abstract

We provide a formal definition of depth-limited games together with an accessible and rigorous explanation of the underlying concepts, both of which were previously missing in imperfect-information games. The definition works for an arbitrary extensive-form game and is not tied to any specific game-solving algorithm. Moreover, this framework unifies and significantly extends three approaches to depth-limited solving that previously existed in extensive-form games and multiagent reinforcement learning but were not known to be compatible. A key ingredient of these depth-limited games is value functions. Focusing on two-player zero-sum imperfect-information games, we show how to obtain optimal value functions and prove that public information provides both necessary and sufficient context for computing them.

We provide a domain-independent encoding of the domain which allows for approximating value functions even by simple feed-forward neural networks. We use the resulting value network to implement a depth-limited version of counterfactual regret minimization. In three distinct domains, we show that the algorithm produces a low-exploitability strategy if and only if it is paired with a near-optimal value network. We show that the value network is capable of generalizing to unseen game situations and that the resulting algorithm performs on par with CFR-D despite being trained on randomly-generated game situations.

Introduction

Sequential decision-making is a key challenge in AI research. As the number of subsequent decisions increases, the size of the state space grows exponentially, to the point where even modern computer clusters soon become unable to even enumerate all states. In perfect information problems, this issue is typically overcome by replacing the states below a certain depth by a value function. This technique vastly reduces the effective size of the game, which is essential for both minimax-like search and reinforcement learning. In imperfect information problems, value functions are much more complex since they depend on the agent's belief about the current state. The situation gets even more complicated in multiagent imperfect information problems, where values additionally depend on each agent's belief about other agents' beliefs (etc.). Despite these challenges, recent results (Moravčík et al. 2017; Brown and Sandholm 2017; Serrino et al. 2019) illustrate that depth-limited approaches can be successful even in this setting. However, many of the concepts needed by these approaches were introduced informally or in domain-specific ways that do not readily generalize to arbitrary games.

In the full version of this paper, (Kovařík et al. 2020), we (1) provide a theory of depth-limited methods and value functions that unifies three recent approaches (Moravčík et al. 2017; Brown, Sandholm, and Amos 2018; Wiggers, Oliehoek, and Roijers 2016), (2) formulate all required concepts in an accessible and domain-independent way, and (3) experimentally demonstrate that depth-limited solving is a viable option for a range of imperfect information games beyond poker.

More specifically, we give an extensive theoretical analysis: We define the basic notions of expected utilities of histories and information sets, reach probabilities, and beliefs in a unified way that is consistent with previous literature but much more intuitive and easy to use. We then provide several technical propositions which substantially simplify our proofs. They also make some of the existing proofs more rigorous and are likely to be instrumental in future research. Using these concepts, we look at imperfect-information games and propose domain- and algorithm-independent notions of value functions and depth-limited games. We also describe (the depth-limited versions of) various algorithmic problems such as game-value computation, equilibrium computation, and best response computation. Our goal for each of these problems is that whenever we find a solution of the depth-limited version of the problem and plug it into the full game, it should fulfil the role of a (partial) solution to the non-depth-limited version of the problem. For example, a depth-limited Nash equilibrium should coincide with a standard Nash equilibrium in all decision-points above the depth limit. However, even some promising choices of value functions can fail to work for all of the above problems. We thus describe a natural hierarchy of conditions on value functions and formally prove that each of them fulfils the above goal for a different class of computational problems. Moreover, we observe that with the proposed formalization, the two previously separate approaches to depth-limited solving of imperfect information

[*]These authors contributed equally

[†]Corresponding author: viliam.lisy@agents.fel.cvut.cz

games – value functions (Moravčík et al. 2017) and multivalued states (Brown, Sandholm, and Amos 2018) – can be viewed as two instances of a single unifying framework. After studying the fundamental properties of value functions, we discuss methods for representing value functions more efficiently. One part of this endeavour is encoding the functions' input and output more compactly — we show that value functions can be defined on either individual histories or information sets and that the two representations can be translated to each other. Moreover, we formally prove that value functions can be factorized based on public information (or common knowledge) and that this factorization cannot be further refined in general. The second part is approximating value functions by neural networks, which is likely to be easier if there is a unique approximation target. Unfortunately, we show that there can sometimes be multiple suitable value functions. Failing uniqueness, we investigate whether the set of suitable value functions is at least well-behaved – i.e., convex. We prove that this is true for certain types of value functions and pose the general question as an open problem. However, we remain optimistic about value function approximation, since the non-uniqueness did not prove to be a problem in practice.

Finally, while all of these results are presented using the extensive-form game formalism, we go on to explain how all of them apply the factored-observation stochastic game (Kovařík et al. 2019) (FOSG) and partially-observable stochastic game (Hansen, Bernstein, and Zilberstein 2004) formalisms (which are primarily used in multiagent reinforcement learning).

The theoretical part describes depth-limited versions of arbitrary domains and various algorithms. Among the various game solving algorithms, counterfactual regret minimization (CFR) seems particularly promising due to its recent successes in poker (Moravčík et al. 2017; Brown and Sandholm 2017, 2019). For this reason, our experiments focus on the depth-limited version of CFR. We show that any game represented as FOSG admits a unified representation of inputs and outputs to the value function. We demonstrate, in three domains with very different properties, that this representation can serve as a suitable encoding for a neural network. This encoding allows for an accurate approximation even with the simplest feed-forward architecture. We show that when this value network is plugged into depth-limited CFR, the algorithm produces strategies with low exploitability. As suggested by (Moravčík et al. 2017), we experiment with many different loss functions in an attempt to identify those that work the best in combination with CFR. To our surprise, the algorithm's performance is not very sensitive to this parameter. We also investigate whether the trained value function is robust to differences between the training distribution and the one demanded by depth limited CFR. In all three domains, the trained value function generalizes well to unseen situations. Furthermore, we investigate the dependence of the quality of the learned value function on the amount of training data and key hyper-parameters.

Summary of Theoretical Contributions

We will now give a brief summary of the theoretical contributions in (Kovařík et al. 2020).

We start by providing accessible and rigorous definitions of basic concepts around EFGs and CFR, which is something that has been missing for some time now. We prove that history and infoset values can be defined in several equivalent ways, which simplifies many of our subsequent proofs. Moreover, these results will likely be of use for various theoretical CFR-related results in the future. We then introduce depth-limited versions of many notions such as exploitability, best-response, and Nash equilibria. Afterwards, we define the abstract notion of depth-limited games — these can be understood as game trees where terminal values depend not just on the leaf in which the game ends, but also on the strategy used to get to the leaf. To utilize this notion, we need to identify depth-limited games that are useful for finding equilibria of non-depth-limited games. We thus provide a sufficient condition for solutions of a depth-limited game to coincide with Nash equilibria of the original game (or, more precisely, with strategies that can be extended into such Nash equilibria). Unfortunately, we also see that this criterion alone does not imply the existence of *practical* methods for solving the depth-limited game.

To allow for such practical methods, we define optimal value functions as those that correspond to values of optimal extensions of the trunk strategy. We make a distinction between (a) reachably-optimal, (b) counterfactually-optimal, and (c) universally-optimal value functions, based on whether the corresponding extension of the trunk-strategy performs well (a) in situations that arise under the trunk strategy, (b) in those that could arise if one player deviated from the trunk strategy, or (c) in all situations. We look at each type of optimality in detail, showing how to compute the corresponding value functions and discussing which depth-limited algorithms are enabled by them — for a summary, see Table 1. In particular, we prove that counterfactually-optimal value functions are both sufficient and necessary to make depth-limited CFR work well.

We continue to relax the notion of optimality to only require being optimal w.r.t. a subset of all strategies and observe that this still enables finding high-quality strategies. Moreover, we observe that (Brown, Sandholm, and Amos 2018)'s multivalued states can be understood as a specific instance of depth-limited solving that relies on this type of optimal value functions. We argue that this connection deserves further attention.

Finally, we show how to represent value functions more compactly and prove that public states provide the minimum context that is still informative enough to enable the computation of optimal values. We also note that optimal values are not uniquely defined, which could prove important (and potentially inconvenient) when approximating them by neural nets. However, we do not currently see this topic as priority since we have not encountered any difficulties in practice.

Optimality	reachable	counterfactual	universal
Optimal at	$P^{\sigma^\tau}(I) > 0$	$P^{\sigma^\tau}_{-p}(I) > 0$	all I
Suff. stats	joint range	range	beliefs
How to compute?	value-solving	post-processing via best-response, CFR	add noise to ranges & take the limit
Algorithms enabled	IS-MCTS, subgame value	CFR, best response, poss. fictitious play	minimax, possibly others

Table 1: A summary of properties of different types of optimal value functions.

Value Functions in EFGs, POSGs, and FOSGs

In this section, we summarize the connection between the EFG model and the POSG model (traditionally used in multi-agent RL). In particular, we conclude that all of our results apply to POSGs as well. We then discuss the connection between the value functions studied in the present text and the v- and q-values used in RL.

Recall that the standard MARL model for zero-sum[1] games is a partially-observable stochastic game (POSG), where players take actions which causes them to transition to a new state, obtain some reward, and receive some observation (Hansen, Bernstein, and Zilberstein 2004). The recent paper (Kovařík et al. 2019) shows that every POSG *naturally* corresponds to a *strategically equivalent* EFG — this is achieved by taking the possibly-cyclic POSG and unrolling it into its tree-structured extensive-form representation. Moreover, essentially every EFG can be obtained by applying this process to some POSG (Kovařík et al. 2019, Thm. 5.4). Informally speaking, this indicates that we should not view EFGs and POSGs as two alternative models, but we should instead treat POSGs as the *underlying* model from which EFGs are *derived*. And whenever we want to apply an "EFG-result" to POSGs, we simply apply it to the POSG's extensive-form representation.

Critically, the derivation process *abstracts away* some information. To the extent that our discussion only involves concepts that are preserved in the EFG representation, we should thus use the conceptually simpler EFG model. On the other hand, whenever the EFG representation misses out on some relevant information, we can use the underlying POSG to recover it. We perform this using an example in this section since it allows us to replace infosets by the more-informative notion of action-observation sequences (which increases the generalization power of resulting value-networks).

One related downside of the standard POSG model is that POSGs do not, by default, have a notion analogous to public states in EFGs. To overcome this problem, (Kovařík et al. 2019) suggest using a minor extension of the classical model called *factored-observation stochastic game* (FOSG), which automatically keeps track of which observations are public.[2] Sequences of public observations then serve the same role as public states in EFGs, allowing exactly the same localized computation of values as EFGs.

Finally, we also claim that values traditionally used in reinforcement learning are closely connected to the values discussed in the present paper. Our definition of a situation's value can be intuitively formulated as the "expected total reward", i.e., both the already-obtained reward and the reward we are yet to receive). This is because EFGs abstract away information about *when* rewards are received, by pushing them all the way down to terminal nodes. In contrast, RL models retain this information, so one could also define the value of a situation in a POSG as the "expected *future* reward". Which approach is more sensible and how are they related? The good news is that the two approaches are equivalent in many games — more specifically, in games that allow players to see their own rewards in real time. Indeed, in such games, the values given by the two approaches only differ by a constant known to the player, which ensures that the player will make identical "decisions" under both approaches (i.e., most algorithms will yield identical outputs).

Domain-Independent Encoding of Value Functions using FOSGs We now explain how (and why) to use FOSGs for encoding the input and output of value functions in a domain-independent manner. We elaborate on our choice of domains that appear in the experiments and explain how the encoding applies to them.

Among other differences, this model enriches the notion of an infoset by the concept of public and private observations. It allows for identifying public states by sequences of public observations, and infosets within a given public state by sequences of private observations and actions. This provides a domain-independent method for encoding a value function's input in a way that can be efficiently used with simple feed-forward networks, removing the need for hand-crafting inputs.

We chose **imperfect information goofspiel** (GS), a card game commonly used for evaluating game-theoretic algorithms (Lanctot et al. 2009; Lisý, Lanctot, and Bowling 2015), a betting game called **imperfect information oshi-zumo** (OZ) (Buro 2004), and — to compare to a commonly used baseline — the poker variant **Leduc hold'em** (LH)

[1] While value-functions in other types of POSGSs — e.g., in Dec-POMDPs (Oliehoek, Vlassis et al. 2006) — are sometimes superficially similar to values in our setting, assumptions often made in these domains (e.g., joint training) give the resulting concepts vastly different properties. As such, their discussion would be outside of the scope of the present paper.

[2] Arguably, distinguishing between private and public information is more natural than not doing so. Rather than saying that the FOSG model tracks public information, we might thus rather say that the other models choose to forget it.

(Moravčík et al. 2017). Notably, all chosen domains have distinct properties. One difference is that while a game of goofspiel never ends before reaching the depth limit, this can happen in both LH and OZ. Moreover, there is a difference in the observability of players' actions: all actions are publicly observable in poker (LH) while actions in GS and OZ are unobservable by the opponent. Relatedly, all public states in LH have the same number of information sets and all infosets in LH have the same number of histories. In contrast, both GS and OZ have very unevenly-sized public states and infosets. To illustrate these distinctions, consider the following example. In OZ, players start with some numbers of coins, e.g., 8. Each round, each player secretly bids some of the coins they have left. Subsequently, a referee publicly announces which player bid more (or that a draw occurred), but not the actual bid size. Suppose that in the first round, P1 bids all 8 coins and publicly observes "win". P1 now knows that their opponent bid 7 coins or less — that is, their information set contains histories corresponding to bet sizes of 7 and less. On the other hand, if the first round resulted in a draw, the information set would only contain a single history, corresponding to both players bidding 8. Contrast this with the situation in Leduc hold'em poker. In LH, imperfect information only arises from the initial assignment of private cards (i.e. the players' *private observations*). As a result, all infosets contain the same number of histories (one for each card in the deck except for the private card of the given player).

Table 2 summarizes the differences between the chosen domains, showcasing their distinct features and provides additional information their approximate sizes.

Settings	GS	OZ	LH
rounds	5	8	2
specifics	5 cards	8 C/min 1	2 B & R
infosets	5 000	1 600	4 000
histories	56 000	20 000	61 000
depth-limit	3rd	4th	2nd
private actions	Yes	Yes	No
early terminals	No	Yes	Yes
constant size IS	No	No	Yes

Table 2: Specific details, sizes, depth-limits (in rounds), and game-specific parameters of the domains used for experimental evaluation.

Encoding of Inputs and Outputs To specify the input and output of the value network, we need to describe the encoding of public states, the information sets contained in them, and the corresponding ranges and counterfactual values. In general domains, each public state can be identified with the corresponding *sequence* of *public observations* (this can include publicly visible actions) received on the way until the public state is encountered. *Within a specific public state*, each player's infosets are uniquely identified by the given player's action-observation history (i.e., by the sequence of the player's actions and private observations). To use the same encoding for *all* public states, we can consider, in each public state, all action-observation histories that are compatible with *some* public state at the given depth. This has the advantage that both the range (input) and the vector of counterfactual values (output) will be in the form of a fixed-size vector. This fixed-size representation results in some "virtual" infosets that are logically incompatible with the given public state — for example, if player 1 loses the first round in oshi-zumo (public state), it is not possible that their secret action was "bid the maximum amount of coins". Fortunately, this can be solved by setting the ranges and counterfactual values of all incompatible infosets to 0.

In all games, the sequence of public observations is encoded as a sequence of one-hot vectors, one for each observation. This is straightforward in GS and OZ, where the public observations only contain the round results (either "win for pl. 1", "draw", or "loss for pl. 1"). In LH, the description of the public state consists of the sequence of all actions taken (since they are publicly visible) and the public card. Since the number of actions (i.e., bet sizes) available in LH is small, we can represent each action as a one-hot encoding over, for example, the vector ("Check", "Bet 2", and "Bet 4"). Another complication with public states in LH is that not all public states at the bottom of the trunk are in the same depth, i.e. they have a varying number of public observations. We solve this by padding the shorter public-observation sequences by zero. The encoding of each player's private action-observation histories is done in a positional manner. In LH, the only private information is the cards the players are holding, which makes the amount of hidden information constant throughout the game. In contrast, all actions in GS and OZ are private. Figure 1 showcases the encoding of the "Win" public state in a three-round GS after the first round.

$S = [Win]$	[0,0,1]
$I_1 = [Play2, Play3]$	$[0, Rng(Play2), Rng(Play3)]$
$I_2 = [Play2, Play1]$	$[Rng(Play1), Rng(Play2), 0]$

Figure 1: Encoding of the "win" (for P1) public state in GS-3 after the first round. The left box shows the public state (top) and P1's information sets (middle) and P2's information sets (bottom). The box on the right shows how each of the three parts is encoded. The "win" public state contains information states where P1 bid 2 and 3 (but not 1). P2 has information states where he bid 2 and 3 (but not 3). There are three possible outcomes for the round result. We use the one-hot encoding $[0, 0, 1]$ to denote a win of P1. After the first round each player has three possible action sequences (bid either card 1,2 or 3). Hence, we allocate a vector of length 3 where each position corresponds to the range (**Rng**) of selecting a card. Note that range is a real number denoting a player's probability of playing into a certain information set. We assign the range of valid information sets to their corresponding positions in the input vectors, leaving incompatible sequences at 0. The full input then comprises of an encoded sequence of public observations and ranges of all information sets for each player.

Summary of Experimental Results

While the theoretical part of the full paper describes depth-limited versions of *general* algorithms, our experiments focus on a depth-limited variant of counterfactual regret minimization (CFR). Since this algorithm was recently highly successful in poker, we think that a particularly promising path is to extend it to other games using the theoretical insights summarized above.

Experimental Setup

The encoding described above enables us to use a simple, fully-connected feed-forward network to approximate the value function. We introduce a depth limit by splitting each game into a **trunk** (e.g., each player performs three moves) and the **bottom** (the remainder of the game).

An optimal value function takes a sequence of public observations and a corresponding range as input and returns counterfactual-value vectors, one for each player. One batch of training data is, therefore, generated as follows. First, we assign a randomly generated strategy to each infoset in the trunk and fix it (i.e., we do not update those strategies while solving the game). We then solve the bottom of the game using 1000 iterations of CFR+ (Tammelin 2014). We refer to this technique as **value solving** and show that it yields a counterfactually (near-)optimal value function. For each public state at the depth-limit, we extract the probabilities of the sequences of private observations (*ranges*) for each player and their corresponding near-optimal values.

Since the value function is defined over public states, one random trunk strategy (for all public states) translates into as many data samples as there are public states at the chosen depth-limit.

Depth-limited Solver In order to find an ϵ-Nash equilibrium in the trunk, we use a depth-limited version of CFR+ in conjunction with a counterfactually optimal value function. We refer to this algorithm as DL-CFR$^+_{\text{NN}}$. In each iteration, it gathers the ranges for all infosets at the depth-limit and feeds them into the neural network. For each public state S at the depth-limit, we then call the network (with the ranges corresponding to S). Afterward, we run the regret matching plus algorithm in each infoset in the trunk. The implementation of DL-CFR$^+_{\text{NN}}$ is available online at (GTLib2 2020).

Experiment Description

Our experiments are intended to address unanswered questions revolving around value functions. First, can value functions be trained for other domains than poker (which has convenient properties) using the same domain-independent encoding and architecture? Second, which loss should be minimized in order to achieve low exploitability? Third, when training value networks in general domains, is it sufficient to use random trunk strategies, solve them, and use their corresponding solutions as training data? Fourth, can these value functions generalize to unseen situations?

The experimental results are presented in the order of their importance to the research questions posed above and their main content summarized. The main part of our analysis consists of three experiments which we describe in greater detail, while the additional experiments which revolve around neural network training are very briefly summarized. We start with the main experiments.

Main Experiments When approximating value functions using neural networks, the key performance criterion is the exploitability of the resulting strategy when using the network in conjunction with DL-CFR$^+_{\text{NN}}$. However, computing exploitability is costly. It would, therefore, be preferable if we could predict DL-CFR$^+_{\text{NN}}$'s performance solely based on the loss of the network it employs. We therefore investigate the effect of the neural network's error on the quality of the resulting strategy provided by the depth-limited solver that uses it. We do this by training multiple networks, saving their weights and corresponding validation losses during the process, and recording the exploitability of the resulting strategy of DL-CFR$^+_{\text{NN}}$ using each of the networks' checkpoints. We see a strong connection between network quality and resulting exploitability. A low (depth-limited) exploitability can be reached long before the network error is minimized. Moreover, we were able to reach exploitability below 0.01 (similar to CFR+), as well as Huber loss below 0.001, l_1-errors below 0.01, and l_∞-error below 0.1. Figure 2 shows all three monitored losses for GS. In all tested domains, all of these variables were strongly correlated.

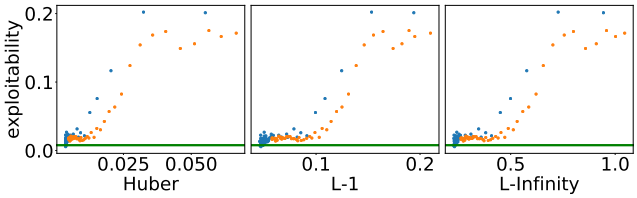

Figure 2: Normalized exploitability vs validation error in Goofspiel for two different networks (blue/orange dots) and exploitability of CFR+ after 1000 iterations (green line). We observe sparse "tails" due to rapid improvement of the loss during the initial epochs.

We would also like to know whether the network can generalize to parts of the game that have not been encountered before. While the standard validation- and test-losses indicate whether the network can generalize to unseen samples, this only informs us about its ability to generalize under the original training distribution. Since the value function is defined over public states, we can omit one of them, train a network on the remaining ones, and compute validation loss on the one we left out. We perform this procedure for every public state at the depth-limit to evaluate if the used encoding captures similarities between public states. We are able to generalize to unseen public states across all domains. However, all monitored validation losses drastically improve with the number of public states available at the depth-limit Indeed, the domains vary in how many public states are at the trunk border (only a few in GS, a medium number in OZ, and many in LH), and this translates into L_∞ error 2.6 in GS, 0.1 in OZ, and 0.08 in LH. We attribute this to the relative

amount of data left out of the training set. We conclude that we would be able to reach exploitabilities below 0.1 in both LH and OZ using neural networks that have not been trained on all possible game situations

Lastly, we investigate the representativeness of the training ranges and analyze DL-CFR$^+_{\text{NN}}$'s game-solving behavior by comparing randomly generated ranges used for training with the ranges DL-CFR$^+_{\text{NN}}$ asks for during the execution and the ranges that the CFR-D algorithm would ask for if used for solving the same depth-limited game. While solving the game, the trained network is likely to be queried for ranges which come from a substantially different distribution than the one it has been trained on. To examine if this indeed happens, we run CFR-D and DL-CFR$^+_{\text{NN}}$, track the ranges (probabilities of reaching information sets) at each iteration, and compare them to the random data used for training. We observe a strong similarity between DL-CFR$^+_{\text{NN}}$ and CFR-D ranges, with the latter being slightly more uniform in the initial iterations. We believe this is caused by CFR-D receiving more precise values (while the value network's minor errors result in a non-uniform strategy). However, randomly generated ranges (used for neural network training) are substantially different from those of both DL-CFR$^+_{\text{NN}}$ and CFR-D. This experiment emphasizes that a value function can be trained on random ranges in the trunk, generalize from them nonetheless, and achieve good results that mimic the behavior of CFR-D. However, for larger domains with high dimensional belief spaces, a training procedure similar to (Brown et al. 2020) might be more appropriate.

Additional Experiments The secondary part of the evaluation contains revolves around training counterfactual-value-networks. We analyze the influence of hyperparameter configurations, such as the number of hidden layers and layer width. We found that using 4-6 hidden layers with approximately 200-500 neurons works best for all tested domains. We elaborate on our in-depth investigation of custom loss functions and analyze the differences between minimizing the Huber loss and validating on l_1 and l_∞, and vice versa. This analysis lets us understand their relationship and learn which of them is crucial to the network's performance. We conclude that using standard losses is sufficient which is also supported by the experiment shown in Figure 2. Since the data generation process can be costly, we additionally investigate how much data is needed to achieve satisfactory network performance. We observe that for our medium-sized domains approximately 50$\dot{0}$00 data samples is sufficient to achieve low validation Huber loss of 0.001 (which, as we show in Figure 2, suffices to achieve $\sim$0.01 exploitability).

Lastly, we show that training on our randomly generated training ranges is enough to achieve satisfactory validation losses on data generated by CFR-D while solving the game, which is of a substantially different distribution than the used training data.

Takeaways Our experiments show that value functions can be approximated in domains other than poker, using the same domain-independent encoding and architecture. We show that validation losses of value networks w.r.t. Huber, l_1, and l_∞ are all highly correlated with the exploitability of the DL-CFR$^+_{\text{NN}}$ that employs the network. In all cases, low validation loss translated into low exploitability. Furthermore, the chosen representation and encoding of the value function enables for a generalization to unseen game situations, across all tested domains. We established that it is sufficient to use random ranges as input to the neural network to achieve game-solving behaviour similar to CFR-D. In the secondary part of our experimental evaluation, we analyzed optimal hyperparameter configurations, validation losses, performance on CFR-D data and examined the amount of training data to achieve satisfactory validation losses.

Conclusion

While this particular paper acts as a short and concise summary of the key insights in (Kovařík et al. 2020), we would like to conclude by stating the contributions of the full paper. We give an accessible description of basic notions used in the CFR literature and introduced a number of concepts that enable reasoning about depth-limited games and value functions. We prove that different degree of value-function optimality is required for different calculations and give a recipe for obtaining each type of value function. Additionally, we proved that public belief states provide the necessary and sufficient context for computing value functions. Our description allows viewing Deepstack's value functions (Moravčík et al. 2017) and Brown et al.'s multivalued states (Brown, Sandholm, and Amos 2018) as two instances of a single unifying framework. Moreover, the results also apply to partially-observable stochastic games and their recent extension, factored-observation stochastic games (Kovařík et al. 2019). The theory shows that depth-limited solving is applicable to arbitrary domains and various algorithms. However, due to CFR's success in recent years, our experimental evaluation focused on CFR. We showed that adopting the FOSG formalism allows for a simple domain-independent encoding which can be used for input and output of a value function. In three distinct domains, we used this encoding to train a simple feed-forward neural network that approximates an optimal value function. We then implemented a depth-limited version of CFR that utilizes this network. We performed an extensive experimental evaluation of this setup. Most importantly, we confirmed that the value network's error is strongly correlated with the exploitability of the strategy found by the corresponding DL-CFR$^+_{\text{NN}}$. Furthermore, we demonstrated that the value function can generalize to unseen public states and that it is highly robust to distributional shift. Lastly, we performed an extensive analysis of the impact of key hyperparameters for data generation and training the value function. Overall, we have shown that depth-limited solving is a viable and robust option for a range of imperfect-information games beyond poker.

References

Brown, N.; Bakhtin, A.; Lerer, A.; and Gong, Q. 2020. Combining Deep Reinforcement Learning and Search for Imperfect-Information Games. *arXiv preprint arXiv:2007.13544* .

Brown, N.; and Sandholm, T. 2017. Superhuman AI for heads-up no-limit poker: Libratus beats top professionals. *Science* eaao1733.

Brown, N.; and Sandholm, T. 2019. Superhuman AI for multiplayer poker. *Science* 365(6456): 885–890.

Brown, N.; Sandholm, T.; and Amos, B. 2018. Depth-Limited Solving for Imperfect-Information Games. *arXiv preprint arXiv:1805.08195* .

Buro, M. 2004. Solving the oshi-zumo game. In *Advances in Computer Games*, 361–366. Springer.

GTLib2. 2020. GTLib2 Implementation of DL-CFRNN. https://gitlab.fel.cvut.cz/game-theory-aic/GTLib2/-/blob/master/algorithms/cfr_dl.cpp. Accessed: 2020-09-17.

Hansen, E. A.; Bernstein, D. S.; and Zilberstein, S. 2004. Dynamic programming for partially observable stochastic games. In *AAAI*, volume 4, 709–715.

Kovařík, V.; Schmid, M.; Burch, N.; Bowling, M.; and Lisý, V. 2019. Rethinking Formal Models of Partially Observable Multiagent Decision Making. *arXiv preprint arXiv:1906.11110* .

Kovařík, V.; Seitz, D.; Lisỳ, V.; Rudolf, J.; Sun, S.; and Ha, K. 2020. Value Functions for Depth-Limited Solving in Imperfect-Information Games. *arXiv preprint arXiv:1906.06412* .

Lanctot, M.; Waugh, K.; Zinkevich, M.; and Bowling, M. 2009. Monte Carlo sampling for regret minimization in extensive games. In *Advances in neural information processing systems*, 1078–1086.

Lisý, V.; Lanctot, M.; and Bowling, M. 2015. Online Monte Carlo counterfactual regret minimization for search in imperfect information games. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, 27–36. International Foundation for Autonomous Agents and Multiagent Systems.

Moravčík, M.; Schmid, M.; Burch, N.; Lisý, V.; Morrill, D.; Bard, N.; Davis, T.; Waugh, K.; Johanson, M.; and Bowling, M. 2017. Deepstack: Expert-level artificial intelligence in heads-up no-limit poker. *Science* 356(6337): 508–513.

Oliehoek, F.; Vlassis, N.; et al. 2006. Dec-POMDPs and extensive form games: equivalence of models and algorithms. *Ias technical report IAS-UVA-06-02, University of Amsterdam, Intelligent Systems Lab, Amsterdam, The Netherlands* .

Serrino, J.; Kleiman-Weiner, M.; Parkes, D. C.; and Tenenbaum, J. B. 2019. Finding Friend and Foe in Multi-Agent Games. *arXiv preprint arXiv:1906.02330* .

Tammelin, O. 2014. CFR+. *CoRR, abs/1407.5042* .

Wiggers, A. J.; Oliehoek, F. A.; and Roijers, D. M. 2016. Structure in the value function of two-player zero-sum games of incomplete information. In *Proceedings of the Twenty-second European Conference on Artificial Intelligence*, 1628–1629. IOS Press.