Bayesian Action Decoder for Deep Multi-Agent Reinforcement Learning

Jakob N. Foerster [*1 2] **H. Francis Song** [*2] **Edward Hughes** [2] **Neil Burch** [2] **Iain Dunning** [2] **Shimon Whiteson** [1]
Matthew M. Botvinick [2] **Michael Bowling** [2]

Abstract

When observing the actions of others, humans carry out inferences about why the others acted as they did, and what this implies about their view of the world. Humans also use the fact that their actions will be interpreted in this manner when observed by others, allowing them to act informatively and thereby communicate efficiently with others. Although learning algorithms have recently achieved superhuman performance in a number of two-player, zero-sum games, scalable multi-agent reinforcement learning algorithms that can discover effective strategies and conventions in complex, partially observable settings have proven elusive. We present the *Bayesian action decoder* (BAD), a new multi-agent learning method that uses an approximate Bayesian update to obtain a public belief that conditions on the actions taken by all agents in the environment. Together with the public belief, this Bayesian update effectively defines a new Markov decision process, the *public belief MDP*, in which the action space consists of deterministic partial policies, parameterised by deep neural networks, that can be sampled for a given public state. It exploits the fact that an agent acting only on this public belief state can still learn to use its private information if the action space is augmented to be over partial policies mapping private information into environment actions. The Bayesian update is also closely related to the *theory of mind* reasoning that humans carry out when observing others' actions. We first validate BAD on a proof-of-principle two-step matrix game, where it outperforms traditional policy gradient methods. We then evaluate BAD on the challenging, cooperative partial-information card game Hanabi, where in the two-player setting the method surpasses all previously published learning and hand-coded approaches.

[*]Equal contribution [1]University of Oxford, UK [2]DeepMind, London, UK. Correspondence to: Jakob Foerster <jakob.foerster@cs.ox.ac.uk>, Francis Song <songf@google.com>.

1. Introduction

In multi-agent reinforcement learning, agents must learn to act in an environment that contains multiple learning agents, often under partial observability (Littman, 1994). In recent years, a variety of deep reinforcement learning (RL) methods have been adapted to this setting (Foerster et al., 2016a; Lowe et al., 2017; Perolat et al., 2017; Jaderberg et al., 2018). In the particular case of cooperative, partially observable multi-agent settings, a key challenge is to discover communication protocols while simultaneously learning policies. The ability to learn such protocols is essential for many real-world tasks where agents have to interact and communicate seamlessly with other agents.

State-of-the-art deep RL methods for learning communication protocols mostly use backpropagation across a communication channel (Sukhbaatar et al., 2016; Foerster et al., 2016a). This approach has two limitations. First, it can only be applied to *cheap-talk* channels in which the communication action has no effect on the environment. Second, it misses the conceptual connection between communication and reasoning over the beliefs of others, which is known to be important to how humans learn to communicate (Grice, 1975; Frank & Goodman, 2012).

A well-known domain that highlights these challenges is Hanabi, a popular, fully cooperative card game of incomplete information that is difficult even for humans (Hanabi won the *Spiel des Jahres* award in 2013). A distinguishing feature of the game is that players can see everyone's hands but their own. To succeed, players must find effective conventions for communication. Since there is no cheap-talk channel, most recent methods for emergent communication are inapplicable, necessitating a novel approach.

The goal in Hanabi is to play a legal sequence of cards and, to aid this process, players are allowed to give each other hints indicating which cards are of a specific rank or colour. These hints have two levels of semantics. The first level is the surface-level content of the hint, which is grounded in the properties of the cards that they describe. This level of

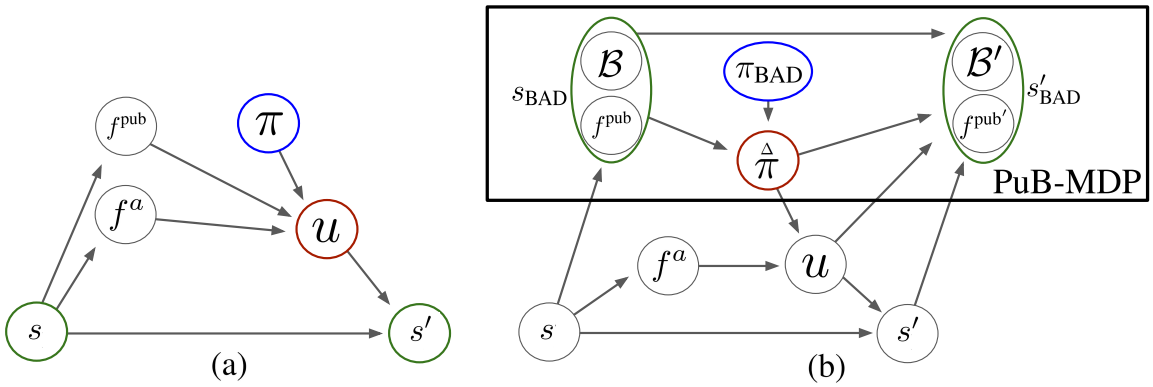


Figure 1: a) In an MDP the action u is sampled from a policy π that conditions on the state features (here separated into f^{pub} and f^a). The next state is sampled from $P(s'|s,u)$. b) In a PuB-MDP, public features f^{pub} generated by the environment and the public belief together constitute the Markov state s_{BAD}. The 'action' sampled by the BAD agent is in fact a deterministic partial policy $\hat{\pi} \sim \pi_{\text{BAD}}(\hat{\pi}|s_{\text{BAD}})$ that maps from private observations f^a to actions. Only the acting agent observes f^a and deterministically computes $u = \hat{\pi}(f^a)$. u is provided to the environment, which transitions to state s' and produces the new observation $f^{\text{pub}'}$. BAD then uses the public belief update to compute a new belief $\mathcal{B}'$ conditioned on u and $\hat{\pi}$ (Equation 1), thereby completing the state transition.

semantics is independent of any possible intent of the agent in providing the hint, and would be equally meaningful if provided by a random agent. For example, knowing which cards are of a specific colour often does not indicate whether they can be safely played or discarded.

A second level of semantics arises from information contained in the actions themselves, i.e., the very fact that an agent decided to take a particular action and not another, rather than the information resulting from the state transition induced by the action. This is essential to the formation of conventions and to discovering good strategies in Hanabi.

To address these challenges, we propose the *Bayesian action decoder* (BAD), a novel multi-agent RL algorithm for discovering effective communication protocols and policies in cooperative, partially observable multi-agent settings. BAD uses all publicly observable features in the environment to compute a *public belief* over the players' private features. This effectively defines a new Markov process, the *public belief Markov decision process* (PuB-MDP), in which the action space is the set of deterministic partial policies, parameterised by deep neural networks, that can be sampled for a given public state. By acting in the space of deterministic partial policies that map from private observations into environment actions, an agent acting only on this public belief state can still learn an optimal policy. Using approximate, factorised Bayesian updates, we show that the public belief method can scale to large state spaces and allow agents to carry out a form of counterfactual reasoning.

When an agent observes the action of another agent, the public belief is updated by sampling a set of possible private states from the public belief and filtering for those states in which the teammate chose the observed action. This process is closely related to the kind of *theory of mind* reasoning that humans routinely undertake (Baker et al., 2017). Such reasoning seeks to understand why a person took a specific action among several, and what information this contains about the distribution over private observations.

We experimentally validate an exact version of BAD on a simple two-step matrix game, showing that it outperforms policy gradient methods. We then apply an approximate version to Hanabi, where BAD achieves an average score of 24.174 points in the two-player setting, surpassing the best previously published results for learning agents by around 9 points and approaching the best known performance of 24.9 points for (cheating) open-hand gameplay. We further show that the beliefs obtained via Bayesian reasoning have 40% less uncertainty over possible hands than those using only grounded information.

2. Background and Setting

Consider a partially observable multi-agent environment with A agents. At time t each agent a takes action u_t^a sampled from policy $\pi^a(u^a|\tau_t^a)$, where τ_t^a is its action-observation history $\tau_t^a = \{o_0^a, u_0^a, .., o_t^a\}$. Here o_t^a are the observations of agent a at time t, which is given by the observation function $O(a, s_t)$ in state s_t. The next Markov state s_{t+1} of the environment is produced by the transition function $P(s_{t+1}|s_t, \mathbf{u}_t)$, which conditions on the joint action $\mathbf{u}_t = \{u_t^1, .., u_t^A\}$, where $u_t^a \in \mathcal{U}$. In the fully cooperative setting that we consider in this work, each agent

receives a per-timestep team reward $r_{t+1}(s_t, \mathbf{u}_t)$ that depends on the last state and last joint action. We also adopt the notion of centralised training and decentralised execution, from which follows that the policies π^a are known to all agents. We also note that this setting can be formalised as a Dec-POMDP (Oliehoek, 2012).

The goal of multi-agent RL is to find a set of agent policies $\{\pi^a\}_{a=1,...,A}$ that maximise the total expected return per episode $J = \mathbb{E}_{\tau \sim P(\tau|\pi^a)}\left[\sum_t \gamma^t r_t\right]$, where γ is the discount factor. In deep RL, optimisation involves training neural networks that represent policies and value functions. In partially observable settings, the networks are typically recurrent, e.g., LSTMs (Wierstra et al., 2009), as they can learn to represent a sufficient statistic of the action-observation history τ_t^a in the hidden activations.

Here we consider a setting where the Markov state s_t consists of a set of discrete features f_t composed of public features f_t^{pub}, which are common knowledge to all agents, and private features f_t^{pri}. Each of the private features is observable by at least one, but not all, of the agents. f_t^a are the private features observable by agent a. For example, in a typical card game the cards being played openly on the table are part of f_t^{pub}, while the cards being held by each player are in f_t^{pri}, with the cards visible to a particular agent in f_t^a. We assume that this separation of state features is common knowledge to all agents. An example of this separation for the case of an MDP is illustrated in Figure 1a.

3. Method

3.1. Public belief

In single-agent partially observable settings, it is clearly useful for an agent to maintain beliefs about the hidden environment state, since this is a sufficient statistic for its action-observation history (Kaelbling et al., 1998). In multi-agent settings, however, it is not obvious what the beliefs should be over. It is not enough to maintain beliefs over the environment state alone, as other agents also have unobservable internal states. In interactive POMDPs (I-POMDPs; Gmytrasiewicz & Doshi 2005), agents model each other's beliefs, beliefs over these beliefs, and so on, but this is often computationally intractable.

Fortunately, in our setting the common knowledge described above makes it possible to compute a *public belief* that makes the recursion of I-POMDPs unnecessary. The public belief $\mathcal{B}_t$ is the posterior over all of the state features given only the public features, i.e., $\mathcal{B}_t = P(f_t | f_{\leq t}^{\text{pub}})$, where $\leq t$ indicates history: $f_{\leq t}^{\text{pub}} = (f_0^{\text{pub}}, .., f_t^{\text{pub}})$. Because $\mathcal{B}_t$ conditions only on publicly available information, it can be computed independently by every agent using a common algorithm that leads to the same result for all agents. Furthermore, since all agents know f^{pub}, we restrict $\mathcal{B}$ to be a posterior only over f^{pri}, not f^{pub}.

While the public belief avoids recursive reasoning, it is not obvious how it can be used to guide behaviour: agents that condition their actions only on the public belief will never exploit their private observations. The key insight behind BAD is that we can construct a special *public agent* whose policy π_{BAD} conditions only on the public observation and the public belief but which nonetheless can generate optimal behaviour. This is possible because an action selected by π_{BAD} specifies a *partial policy*, $\hat{\pi} : \{f^a\} \to \mathcal{U}$, for the acting agent, deterministically mapping private observations to environment actions. The sampling of a deterministic partial policy also addresses a fundamental tension in using policy gradients to learn communication protocols, namely, differentiation and exploration require high-entropy policies, while communication requires low-entropy policies. By sampling in the space of deterministic policies, both can be achieved.

Intuitively, the public agent can be viewed as a third party that can observe only the public observation and belief. While π_{BAD} cannot observe the private state, it can tell each agent what to do for any private observation it might receive. Thus at each timestep, the public agent selects $\hat{\pi}$ based on $\mathcal{B}_t$ and f_t^{pub}; the acting agent then selects the action $u_t^a = \hat{\pi}(f^a)$ by supplying the private observation hidden from the public agent; the public agent then uses the observed action u_t^a to construct the new belief $\mathcal{B}_{t+1}$. While BAD can be applied to synchronous action settings with the help of additional formalism, this complicates the presentation and is not required for our experiments.

3.2. Public Belief MDP

Since $\hat{\pi}$ and u_t^a are public information, observing u_t^a induces a posterior belief over the possible private state features f_t^{pri} given by the *public belief update*:

$$P(f_t^a | u_t^a, \mathcal{B}_t, f_t^{\text{pub}}, \hat{\pi}) = \frac{P(u_t^a | f_t^a, \hat{\pi}) P(f_t^a | \mathcal{B}_t, f_t^{\text{pub}})}{P(u_t^a | \mathcal{B}_t, f_t^{\text{pub}}, \hat{\pi})} \quad (1)$$

$$\propto \mathbb{1}(\hat{\pi}(f_t^a), u_t^a) P(f_t^a | \mathcal{B}_t, f_t^{\text{pub}}). \quad (2)$$

Using this Bayesian belief update, we can define a new Markov process, the *public belief MDP* (PuB-MDP), as illustrated in Figure 1b. The state $s_{\text{BAD}} \in S_{\text{BAD}}$ of the PuB-MDP consists of the public observation and public belief; the action space is the set of deterministic partial policies that map from private observations to environment actions; and the transition function is given by $P(s'_{\text{BAD}} | s_{\text{BAD}}, \hat{\pi})$. The next state contains the new public belief calculated using the public belief update. The reward function marginalises

over the private state features:

$$r_{\text{BAD}}(s_{\text{BAD}}, \hat{\pi}) = \sum_{f^{\text{pri}}} \mathcal{B}(f^{\text{pri}}) r(s, \hat{\pi}(f^{\text{pri}})), \quad (3)$$

where $s_{\text{BAD}} = \{\mathcal{B}, f^{\text{pub}}\}$. Since s'_{BAD} includes the new public belief, and that belief is computed via an update which conditions on $\hat{\pi}$, the PuB-MDP transition function conditions on all of $\hat{\pi}$, not just the selected action u_t^a. Thus the state transition depends not just on the executed action, but on the *counterfactual actions*, i.e., those specified by $\hat{\pi}$ for private observations other than f_t^a.

In the remainder of this section, we describe how factorised beliefs and policies can be used to learn a public policy π_{BAD} for the PuB-MDP efficiently.

3.3. Sampling Deterministic Partial Policies

For each public state, π_{BAD} must select a distribution $\pi_{\text{BAD}}(\hat{\pi}|s_{\text{BAD}})$ over deterministic partial policies . The size of this space is exponential in the number of possible private observations $|f^a|$, but we can reduce this to a linear dependence by assuming a distribution across $\hat{\pi}$ that is factorised across the different private observations, i.e., for all $\hat{\pi}$,

$$\pi_{\text{BAD}}(\hat{\pi}|\mathcal{B}_t, f^{\text{pub}}) := \prod_{f^a} \pi_{\text{BAD}}(\hat{\pi}(f^a)|\mathcal{B}_t, f^{\text{pub}}, f^a). \quad (4)$$

With this restriction, we can easily parameterise π_{BAD} with factors of the form $\pi^{\theta}_{\text{BAD}}(u^a|\mathcal{B}_t, f^{\text{pub}}, f^a)$ using a function approximator such as a deep neural network.

In order for all of the agents to perform the public belief update, the sampled $\hat{\pi}$ must be public. We resolve this by having $\hat{\pi}$ sampled determinstically from a given $\mathcal{B}_t$ and f_t^{pub}, using a common knowledge random seed ξ_t. The seeds are then shared prior to the game so that all agents sample the same $\hat{\pi}$: this resembles the way humans share common ways of reasoning in card games and allows the agents to explore alternative policies jointly as a team.

3.4. Factorised Belief Updates.

In general, representing exact beliefs is intractable in all but the smallest state spaces. For example, in card games the number of possible hands is typically exponential in the number of cards held by all players. To avoid this unfavourable scaling, we can instead represent an approximate factorised belief state

$$P(f_t^{\text{pri}}|f_{\leq t}^{\text{pub}}) \approx \prod_i P(f_t^{\text{pri}}[i]|f_{\leq t}^{\text{pub}}) := \mathcal{B}_t^{\text{fact}}. \quad (5)$$

From here on we drop the superscript and use $\mathcal{B}$ exclusively to refer to the factorised belief. In a card game each factor represents per-card probability distributions, assuming approximate independence across the different cards

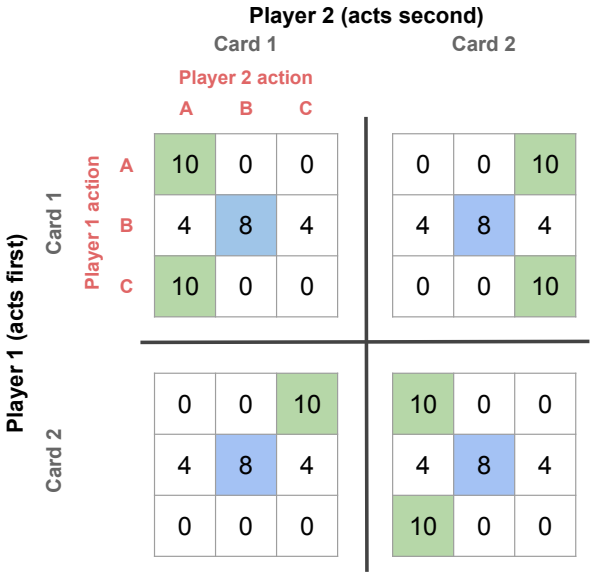


Figure 2: Payoffs for the toy matrix-like game. The two outer dimensions correspond to the card held by each player, the two inner dimensions to the action chosen by each player. Payouts are structured such that Player 1 must encode information about their card in the action they chose in order to obtain maximum payoffs. Although presented here in matrix form for compactness, this is a two-step, turn-based game, with Player 1 always taking the first action and Player 2 taking an action after observing Player 1's action.

both within a hand and across players. This approximation makes it possible to represent and reason over the otherwise intractably large state spaces that commonly occur in many settings, including card games.

To carry out the public belief update with a factorised representation we maintain factorised likelihood terms $\mathcal{L}_t[f[i]]$ for each private feature that we update recursively:

$$\mathcal{L}_t[f[i]] := P(u^a_{\leq t}|f[i], \mathcal{B}_{\leq t}, f^{\text{pub}}_{\leq t}, \hat{\pi}_{\leq t}) \quad (6)$$

$$\approx \mathcal{L}_{t-1}[f[i]] \cdot P(u^a_t|f[i], \mathcal{B}_t, f^{\text{pub}}_t, \hat{\pi}_t) \quad (7)$$

$$= \mathcal{L}_{t-1}[f[i]] \cdot \frac{\mathbb{E}_{f_t \sim \mathcal{B}_t}\left[\mathbb{1}(f_t[i], f[i])\,\mathbb{1}(\hat{\pi}(f_t^a), u_t^a)\right]}{\mathbb{E}_{f_t \sim \mathcal{B}_t}\left[\mathbb{1}(f_t[i], f[i])\right]}, \quad (8)$$

where (7) assumes that actions are (approximately) conditionally independent of the future given the past. As indicated, these likelihood terms are calculated by sampling, and the larger number of samples the better. We sampled $S = 3,000$ hands during training and $S = 20,000$ hands for the final test games.

3.5. Self-Consistent Beliefs

This factorisation is only an approximation, even in very simple card games — knowledge that a player is holding a specific card clearly influences the probability that another player is holding that same card. Furthermore, using our approximation can result in beliefs that are not even self-consistent, i.e., they are not the marginalisation of any belief over joint features. While not central to the key ideas behind BAD, we introduce a general iterative procedure that can account for feature interactions in factorised models. Starting with a public belief $\mathcal{B}_t$ we can iteratively update the belief to make it more self-consistent through re-marginalisation:

$$\mathcal{B}^0 = \mathcal{B}_t, \tag{9}$$

$$\mathcal{B}^{k+1}(f[i]) = \sum_{f[\text{-}i]} \mathcal{B}^k(f[\text{-}i]) P(f[i]|f[\text{-}i], f^{\text{pub}}_{\leq t}, u^a_{\leq t}, \hat{\pi}_{\leq t}) \tag{10}$$

$$\propto \mathbb{E}_{f[\text{-}i]\sim\mathcal{B}^k}\left[\mathcal{L}_t(f[i]) P(f[i]|f[\text{-}i], f^{\text{pub}}_t)\right], \tag{11}$$

where $f[\text{-}i]$ denotes all features excluding $f[i]$. In the last step we used the factorised likelihood terms from above and converted to an expectation, so that we can use samples to approximate the intractable sum across features. The notion of refining the probability across one feature while keeping the probability across all other features fixed is similar to the Expectation Propagation algorithm used in factor graphs (Minka, 2001). However, the card counts constitute a global factor, which renders the factor graph formulation less useful. While this iterative update can in principle be carried out until convergence, in practice we terminate after a fixed number of iterations.

4. Experiments and Results

4.1. Matrix Game

We first present proof-of-principle results for a two-player, two-step partially observable matrix-like game (Figure 2). The state consists of 2 random bits (the cards for Player 1 and 2) and the action space consists of 3 discrete actions. Each player observes its own card, with Player 2 also observing Player 1's action before acting, which in principle allows Player 1 to encode information about its card with its action. The reward is specified by a payoff tensor, $r = \text{Payoff}[\text{card}^1][\text{card}^2][u^1][u^2]$, where card^a and u^a are the card and action of the two players, respectively. The payout tensor is structured such that the optimal reward can only be achieved if the two players establish a convention, in particular if Player 1 chooses informative actions that can be decoded by Player 2.

As shown in Figure 3, BAD clearly outperforms the baseline policy-gradient method on the toy matrix game. In this small, exact setting, it is also possible to estimate counterfactual (CF) policy gradients that reinforce not only the action taken, but also these counterfactual actions. This can be achieved by replacing $\log \pi^a(u^a_t|\tau^a)$ with $\log P(\hat{\pi}|\mathcal{B}_t, f^{\text{pub}})$ in the estimation of the policy gradient. However, the additional improvement in performance from using CF gradients is minor compared to the initial performance gain from using a counterfactual belief state.

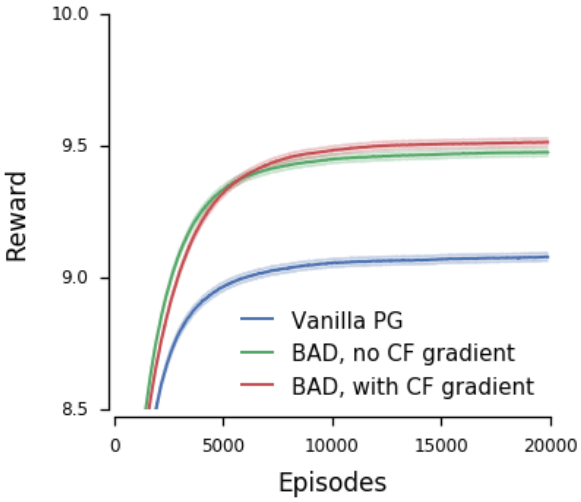


Figure 3: BAD, both with and without counterfactual (CF) gradients, outperforms vanilla policy gradient on the matrix-like game. Each line is the mean over 1000 games, and the shade indicates the standard error of the mean (s.e.m.).

Code for the matrix game with a proof-of-principle implementation of BAD is available at https://bit.ly/2P3YOyd.

4.2. Hanabi

Here we briefly describe the rules of Hanabi.

Let N_h the number of cards in a hand. In the standard game of Hanabi, $N_h = 5$ for $A = 2$ or 3 and $N_h = 4$ for $A = 4$ or 5 players. For generality, we consider that for each colour there are three cards with rank $= 1$, one rank $= N_{\text{rank}}$, and two each of rank $= 2, \ldots, (N_{\text{rank}} - 1)$, i.e., $2N_{\text{rank}}$ cards per colour for a total of $N_{\text{deck}} = 2N_{\text{color}}N_{\text{rank}}$ cards in the deck. In the standard game of Hanabi, $N_{\text{color}} = N_{\text{rank}} = 5$ for a total of $N_{\text{deck}} = 50$. While this is a modestly large number of cards, even for 2 players it leads to 6.2×10^{13} possible joint hands at the beginning of the game.

4.3. Observations and Actions

Each player observes the hands of all other players, but not their own. The action space consists of $N_h \times 2$ options for discarding and playing cards, and $N_{\text{color}} + N_{\text{rank}}$ options per

teammate for hinting colours and ranks. Hints reveal all cards of a specific rank or colour to one of the teammates, e.g., 'Player 2's card 3 and 5 are red'. Hinting for colours and ranks not present in the hand of the teammate (so-called 'empty hints') is not allowed.

Each hint costs one hint token. The game starts with 8 hint tokens, which can be recovered by discarding cards. After a player has played or discarded a card, it draws a new card from the deck. When a player picks up the last card, everyone (including that player) gets to take one more action before the game terminates. Legal gameplay consists of building N_{color} fireworks, which are piles of ascending numbers, starting at 1, for each colour. When the N_{rank}-th card has been added to a pile the firework is complete and the team obtains another hint token (unless they already have 8). Each time an agent plays an illegal card the players lose a life token, after three mistakes the game also terminates. Players receive 1 point after playing any playable card, with a perfect score being $N_{\text{color}}N_{\text{rank}}$.

The number of hint and life tokens at any time are observed by all players, as are the played and discarded cards, the last action of the acting player and any hints provided.

4.4. Beliefs in Hanabi

The basic belief calculation in Hanabi is straightforward: f_t^{pub} consists of a vector of 'candidates' C containing counts for all remaining cards, and a 'hint mask' HM, an $AN_h \times (N_{\text{color}}N_{\text{rank}} + 1)$ binary matrix that is 1 if in a given 'slot' the player could be holding a specific card according to the hints so far, and 0 otherwise; the additional 1 accounts for the possibility that the card may not exist in the final round of play. Slots correspond to the features of the private state space $f[i]$, for example the 3rd card of the second player. Hints contain both positive and negative information: for example, the statement 'the 2nd and 4th cards are red' also implies that all other cards are not red.

The basic belief B^0 can be calculated as

$$B^0(f[i]) = P(f[i]|f^{\text{pub}}) \propto C(f) \times \text{HM}(f[i]). \quad (12)$$

We call this the 'V0 belief', in which the belief for each card depends only on the publicly available information for that card. In our experiments, we focus on baseline agents that receive this basic belief, rather than the raw hints, as public observation inputs; while the problem of simply remembering all hints and their most immediate implication for card counts is potentially challenging for humans in recreational play, we are here more interested in the problem of forming effective conventions for high-level play.

As noted above, this basic belief misses an important interaction between the hints for different slots. We can calculate an approximate version of the self-consistent beliefs that avoids the potentially expensive and noisy sampling step in Equation 11 (note that this sampling is distinct from the sampling required to compute the marginal likelihood in Equation 8). A derivation is in the Appendix:

$$B^{k+1}(f[i]) \propto \left(C(f) - \sum_{j \neq i} B^k(f[j]) \right) \times \text{HM}(f[i]). \quad (13)$$

We call the resulting belief at convergence (or after a maximum number of iterations) the 'V1 belief'. It does not condition on the Bayesian probabilities but considers interactions between hints for different cards. In essence, at each iteration the belief for a given slot is updated by reducing the candidate count by the number of cards believed to be held across all other slots.

By running the same algorithm but including $\mathcal{L}$, we obtain the Bayesian beliefs BB that lie at the core of BAD:

$$\text{BB}^0(f[i]) \propto C(f) \times \text{HM}(f[i]) \times \mathcal{L}(f[i]), \quad (14)$$

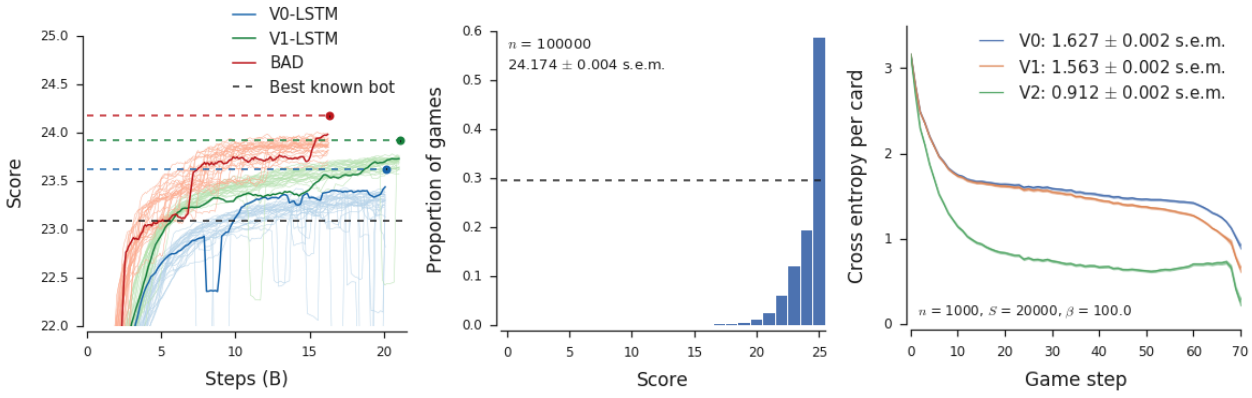
$$\text{BB}^{k+1}(f[i]) \propto \left(C(f) - \sum_{j \neq i} B^k(f[j]) \right) \times \text{HM}(f[i]) \times \mathcal{L}(f[i]). \quad (15)$$

In practice, to ensure stability, the final 'V2 belief' that we use is an interpolation between the Bayesian belief and the V1 belief: $\text{V2} = (1-\alpha)\text{BB} + \alpha\text{V1}$ with $\alpha = 0.01$ (we found $\alpha = 0.1$ to also work).

4.5. Architecture Details for Baselines and Method

Advantage actor-critic agents were trained using the Importance-Weighted Actor-Learner Architecture (Espeholt et al., 2018), in particular the multi-agent implementation described in (Jaderberg et al., 2018). In this framework, 'actors' continually generate trajectories of experience (sequences of states, actions, and rewards) by having agents (self-)playing the game, which are then used by 'learners' to perform batched gradient updates (batch size was 32 for all agents). Because the policy used to generate the trajectory can be several gradient updates behind the policy at the time of the gradient update, V-trace was applied to correct for the off-policy trajectories. The length of the trajectories, or rollouts, was 65, the maximum length of a winning game.

In the V0-LSTM and V1-LSTM agents, all observations were first processed by an MLP with a single 256-unit hidden layer and ReLU activations, then fed into a 2-layer LSTM with 256 units in each layer. The policy π was a linear softmax readout of the LSTM output. The baseline network was an MLP with a single 256-unit hidden layer and ReLU activations, which then projected linearly to a single value. Since the baseline network is only used to compute gradient updates, we followed Foerster et al. (2018)



(a) (b) (c)

Figure 4: a) Hanabi training curves for BAD and the V0 and V1 baseline methods using LSTMs rather than the Bayesian belief. Thick lines indicate the final evaluated agent for each agent type, with the dots showing the final test score. Error bars (standard error of the mean, s.e.m.) are smaller than the dots. Upward kinks in the curves are generally due to agents 'evolving' in PBT by copying its weights and hyperparameters (plus perturbations) from a superior agent. b) Distribution of game scores for BAD on Hanabi under testing conditions. BAD achieves a perfect score in almost 60% of the games. The dashed line shows the proportion of perfect games reported for SmartBot, the best known heuristic for two-player Hanabi. c) Per-card cross entropy with the true hand for different belief mechanisms in Hanabi during BAD play. V0 is the basic belief based on hints and card counts, V1 is the self-consistent belief, and V2 is the BAD belief which also includes the Bayesian update. The BAD agent conveys around 40% of the information via conventions, rather than grounded information.

in feeding each agent's own hand (i.e., the other agent's private observation) into the baseline by concatenating it with the LSTM output; thus we make the common assumption of centralised training and decentralised execution. We note that the V0 and V1-LSTM agents differed *only* in their public belief inputs.

The BAD agent consisted of an MLP with two 384-unit hidden layers and ReLU activations that processed all observations, followed by a linear softmax policy readout. To compute the baseline, we used the same MLP as the policy but included the agent's own hand in the input (this input was present but zeroed out for the computation of the policy).

For all agents, illegal actions (such as hint for a red card when there are no red cards) were masked out by setting the corresponding policy logits to a large negative value before sampling an action. In particular, for the non-acting agent at each turn the only allowed action was the 'no-action'.

Descriptions of the hyperparameters and additional training details are given in the Supplemental Material.

4.6. Results on Hanabi

The BAD agent achieves a new state-of-the-art mean performance of 24.174 points on two-player Hanabi. In Figure 4a we show training curves and test performance for BAD and two LSTM-based baseline methods, as well as the performance of the the best known hand-coded bot for two-player Hanabi. For the LSTM agents, test performance was obtained by using the greedy version of the trained policy, resulting in slightly higher scores than during training. To select the agent, we first performed a sweep over all agents for 10,000 games, then carried out a final test run of 100,000 games on the best agent from the sweep, since taking the maximum of a sweep introduces bias in the score. We carried out a similar procedure for the BAD agent but with additional hyperparameters, also varying the V1 mix-in factor, and number of sampled hands.

The results for other learning methods from the literature perform well below the range of the y-axis (far below 20 points) and are omitted for readability. We note that, under a strict interpretation of the rules of Hanabi, games in which all three error tokens are exhausted should be awarded a score of zero. Under these rules the same BAD agent achieves 23.917 ± 0.009 s.e.m., still better than the hand-coded bot (for whom only results in which all points are counted have been reported). This is true even though our agents were not trained under these conditions.

While not all of the game play BAD learns is easy to follow, some conventions can be understood simply from inspecting the game. Printouts of 100 random games can be found at `https://bit.ly/2zeEShh`. One convention stands out: Hinting for 'red' or 'yellow' indicates that the newest card of the other player is playable. We found that in over

Agent	Learning steps	Mean $\pm$ s.e.m.	Prop. perfect
SmartBot	-	23.09	29.52%
V0-LSTM	20.2B	23.622 ± 0.005	36.5%
V1-LSTM	21.1B	23.919 ± 0.004	47.5%
BAD	16.3B	**24.174± 0.004**	**58.6%**

Table 1: Test scores on 100K games. The LSTM agents were tested with a greedy version of the trained policy, while the final BAD agent was evaluated with V1 mix-in $\alpha = 0.01$, 20K sampled hands, and inverse softmax temperature 100.0.

80% of cases when an agent hints 'red' or 'yellow', the next action of the other agent is to play the newest card. This convention is very powerful: Typically agents know the least about the newest card, so by hinting 'red' or 'yellow', agents can use a single hint to tell the other agent that the card is playable. Indeed, the use of two colours to indicate 'play newest card' was present all of the highest-performing agents we studied. Hinting 'white' and 'blue' are followed by a discard of the newest card in over 25% of cases. We also found that the agent sometimes attempts to play cards which are not playable in order to convey information to their team mate. In general, unlike human players, agents play and discard predominantly from the last card.

Figure 4c shows the quality of the different beliefs. While the iterated belief update leads to a reduction in cross entropy compared to the basic belief, a much greater reduction in cross entropy is obtained using counterfactual beliefs. This clearly demonstrates the importance of learning conventions for successful gameplay in Hanabi: Roughly 40% of the information is obtained through conventions rather than through the grounded information and card counting.

5. Related Work

5.1. Learning to Communicate

Many works have addressed problem settings where agents must learn to communicate in order to cooperatively solve a toy problem. These tasks typically involve a cheap-talk communication channel that can be modeled as a continuous variable during training, which allows differentiation through the channel. This was first proposed by Foerster et al. (2016a) and Sukhbaatar et al. (2016), and has since been applied to a number of different settings. In this work we focused on the case where, rather than relying on a cheap-talk channel, agents must learn to communicate via grounded hinting actions and observable environment actions. This setting is closest to the 'hat game' in Foerster et al. (2016b). In their work the authors proposed a simple extension to recurrent deep Q-networks rather than explicitly modeling action-conditioned Bayesian beliefs. An idea very similar to the Pub-MDP was introduced in the context of decentralised stochastic control by Nayyar et al. (2013), who also formulated a coordinator that uses 'common information' to map local controller information to actions. However, they did not provide a concrete solution method that can scale to a high-dimensional problem like Hanabi.

5.2. Research on Hanabi

A number of papers have been published on Hanabi. Baffier et al. (2016) showed that optimal gameplay in Hanabi is NP-hard even when players can observe their own cards. Encoding schemes similar to the hat game essentially solves the 5-player case (Cox et al., 2015), but only achieve 17.8 points in the two-player setting (Bouzy, 2017). Walton-Rivers et al. (2017) developed a variety of Monte Carlo tree search and rule-based methods for learning to play the game, but the reported scores were roughly 50% lower than those achieved by BAD. Osawa (2015) defined a number of heuristics for the two-player case that reason over possible hands given the other player's action. While this is similar in spirit to our approach, the work was limited to hand-coded heuristics, and the reported scores were around 8 points lower than our results. Eger et al. (2017) investigated humans playing with hand-coded agents, but no pairing resulted in scores higher than 15 points on average.

The best result for two-player Hanabi we could find was for the 'SmartBot' described at `github.com/Quuxplusone/Hanabi`, which has been reported to achieve an average of 23.09 points (29.52% perfect games). While SmartBot uses the same game rules as those used in our work, it is entirely hand-coded and involves no learning.

5.3. Belief State Methods

The continual re-solving (nested solving) algorithm used by DeepStack (Moravčík et al., 2017) and Libratus (Brown & Sandholm, 2018) for poker also used a belief state space. Like BAD, when making a decision in a player state, continual re-solving considers the belief state associated with the current player and generates a joint policy across all player states consistent with this belief. The policy for the actual player is then selected from this joint policy. Continual re-solving also does a Bayesian update of the beliefs after an action. There are key differences, however. Continual re-solving performed exact belief updates, which requires that the joint policy space be small enough to enumerate; belief states were also augmented with opponent values. Continual re-solving is a value-based method, where the training process consists of learning the values of belief states under optimal play. Finally, the algorithm was designed for two-player, zero-sum games, where it can independently consider player state values while guaranteeing that an optimal choice for the joint action policy can be found.

6. Conclusion and Future Work

We presented the *Bayesian action decoder* (BAD), a novel algorithm for multi-agent reinforcement learning in cooperative partially observable settings. BAD uses a factorised, approximate belief state that allows agents to efficiently learn informative actions, leading to the discovery of conventions. We showed that BAD outperforms policy gradients in a proof-of-principle matrix game, and achieves a state-of-the-art performance of 24.174 points on average in the card game Hanabi. We also showed that using the Bayesian update leads to a reduction in uncertainty across the private hands in Hanabi by around 40%. To the best of our knowledge, this is the first instance in which deep RL has been successfully applied to a problem setting that both requires the discovery of communication protocols and was originally designed to be challenging for humans. BAD also illustrates clearly that using an explicit belief computation achieves better performance in such settings than current state-of-the-art RL methods using implicit beliefs, such as recurrent neural networks.

In the future we would like to apply BAD to more than 2 players and further generalise BAD by learning more of the components. While the belief update necessarily involves a sampling step, most of the other components can likely be learned end-to-end. We also plan to extend the BAD mechanism to value-based methods and further investigate the relevance of counterfactual gradients.

Acknowledgements

We thank Marc Lantot, Shibl Mourad, Jelena Luketina, Anuj Mahajan, Gregory Farquhar and Kelsey Allen for valuable discussions.

References

Baffier, J.-F., Chiu, M.-K., Diez, Y., Korman, M., and Mitsou, V. Hanabi is NP-complete, even for cheaters who look at their cards. *arXiv:1603.01911*, 2016. URL https://arxiv.org/abs/1603.01911.

Baker, C. L., Jara-Ettinger, J., Saxe, R., and Tenenbaum, J. B. Rational quantitative attribution of beliefs, desires and percepts in human mentalizing. *Nat. Hum. Behav.*, 1(4):1–10, 2017. doi: 10.1038/s41562-017-0064. URL http://dx.doi.org/10.1038/s41562-017-0064.

Bouzy, B. Playing Hanabi near-optimally. In *Advances in Computer Games*, pp. 51–62. Springer, 2017.

Brown, N. and Sandholm, T. Superhuman ai for heads-up no-limit poker: Libratus beats top professionals. *Science*, 359(6374):418–424, 2018.

Cox, C., De Silva, J., Deorsey, P., Kenter, F. H. J., Retter, T., and Tobin, J. How to Make the Perfect Fireworks Display : Two Strategies for Hanabi. *Math. Mag.*, 88:323, 2015. doi: 10.4169/math.mag.88.5.323. URL http://www.jstor.org/stable/10.4169/math.mag.88.5.323.

Eger, M., Martens, C., and Cordoba, M. A. An intentional AI for hanabi. *2017 IEEE Conf. Comput. Intell. Games, CIG 2017*, pp. 68–75, 2017. doi: 10.1109/CIG.2017.8080417.

Espeholt, L., Soyer, H., Munos, R., Simonyan, K., Mnih, V., Ward, T., Doron, Y., Firoiu, V., Harley, T., Dunning, I., Legg, S., and Kavukcuoglu, K. IMPALA: Scalable Distributed Deep-RL with Importance Weighted Actor-Learner Architectures. *arXiv:1802.01561*, 2018. URL http://arxiv.org/abs/1802.01561.

Foerster, J., Assael, Y. M., de Freitas, N., and Whiteson, S. Learning to communicate with deep multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 2137–2145, 2016a. URL https://arxiv.org/abs/1605.06676.

Foerster, J., Farquhar, G., Afouras, T., Nardelli, N., and Whiteson, S. Counterfactual multi-agent policy gradients. In *The Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18)*, pp. 2974–2982, 2018.

Foerster, J. N., Assael, Y. M., de Freitas, N., and Whiteson, S. Learning to communicate to solve riddles with deep distributed recurrent Q-networks. *arXiv:1602.02672*, 2016b. URL https://arxiv.org/abs/1602.02672.

Frank, M. C. and Goodman, N. D. Predicting pragmatic reasoning in language games. *Science*, 336(6084):998, 2012. doi: 10.1126/science.1218633.

Gmytrasiewicz, P. J. and Doshi, P. A framework for sequential planning in multi-agent settings. *J. Artif. Intell. Res.*, 24:49–79, 2005. doi: 10.1613/jair.1579.

Grice, H. P. Logic and conversation. In Cole, P. and Morgan, J. L. (eds.), *Syntax and Semantics: Vol. 3: Speech Acts*, pp. 41–58. Academic Press, New York, 1975. URL http://www.ucl.ac.uk/ls/studypacks/Grice-Logic.pdf.

Jaderberg, M., Dalibard, V., Osindero, S., Czarnecki, W. M., Donahue, J., Razavi, A., Vinyals, O., Green, T., Dunning, I., Simonyan, K., Fernando, C., and Kavukcuoglu, K. Population Based Training of Neural Networks. *arXiv:1711.09846*, 2017. URL http://arxiv.org/abs/1711.09846.

Jaderberg, M., Czarnecki, W. M., Dunning, I., Marris, L., Lever, G., Castaneda, A. G., Beattie, C., Rabinowitz, N. C., Morcos, A. S., Ruderman, A., Sonnerat, N., Green, T., Deason, L., Leibo, J. Z., Silver, D., Hassabis, D., Kavukcuoglu, K., and Graepel, T. Human-level performance in first-person multiplayer games with population-based deep reinforcement learning. *arXiv:1807.01281*, 2018. doi: arXiv:1807.01281. URL http://arxiv.org/abs/1807.01281.

Kaelbling, L. P., Littman, M. L., and Cassandra, A. R. Planning and acting in partially observable stochastic domains. *Artif. Intell.*, 101(1-2):99–134, 1998. doi: 10.1016/S0004-3702(98)00023-X. URL http://dx.doi.org/10.1016/S0004-3702(98)00023-X.

Littman, M. L. Markov games as a framework for multi-agent reinforcement learning. *Mach. Learn. Proc. 1994*, pp. 157–163, 1994. doi: 10.1016/B978-1-55860-335-6.50027-1. URL http://linkinghub.elsevier.com/retrieve/pii/B9781558603356500271.

Lowe, R., Wu, Y., Tamar, A., Harb, J., Abbeel, O. P., and Mordatch, I. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in Neural Information Processing Systems*, pp. 6379–6390, 2017.

Minka, T. P. Expectation Propagation for Approximate Bayesian Inference. *Uncertain. Artif. Intell.*, 17(2):362–369, 2001. URL https://arxiv.org/abs/1301.2294.

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

Nayyar, A., Mahajan, A., and Teneketzis, D. Decentralized stochastic control with partial history sharing: A common information approach. *IEEE Trans. Automat. Contr.*, 58(7):1644–1658, 2013. ISSN 00189286. doi: 10.1109/TAC.2013.2239000. URL https://arxiv.org/abs/1209.1695.

Oliehoek, F. A. Decentralized pomdps. In *Reinforcement Learning*, pp. 471–503. Springer, 2012.

Osawa, H. Solving hanabi: Estimating hands by opponent's actions in cooperative game with incomplete information. In *AAAI workshop: Computer Poker and Imperfect Information*, pp. 37–43, 2015.

Perolat, J., Leibo, J. Z., Zambaldi, V., Beattie, C., Tuyls, K., and Graepel, T. A multi-agent reinforcement learning model of common-pool resource appropriation. *arXiv:1707.06600*, 2017. ISSN 10495258. URL http://arxiv.org/abs/1707.06600.

Sukhbaatar, S., Szlam, A., and Fergus, R. Learning Multiagent Communication with Backpropagation. *Advances in Neural Information Processing Systems*, 2016. ISSN 10495258. URL http://arxiv.org/abs/1605.07736.

Walton-Rivers, J., Williams, P. R., Bartle, R., Perez-Liebana, D., and Lucas, S. M. Evaluating and modelling Hanabi-playing agents. In *Evolutionary Computation (CEC), 2017 IEEE Congress on*, pp. 1382–1389. IEEE, 2017.

Wierstra, D., Förster, A., Peters, J., and Schmidhuber, J. Recurrent policy gradients. *Log. J. IGPL*, 18(5):620–634, 2009. doi: 10.1093/jigpal/jzp049.

Supplemental Material

A. Hyperparameters and Training Details

For the toy matrix game, we used a batch size of 32 and the Adam optimiser with all default TensorFlow settings; we did not tune hyperparameters for any runs.

For Hanabi, we used the RMSProp optimiser with $\epsilon = 10^{-10}$, momentum 0, and decay 0.99. The RL discounting factor γ was set to 0.999. The baseline loss was multiplied by 0.25 and added to the policy-gradient loss. We used population-based training (PBT) (Jaderberg et al., 2017; 2018) to 'evolve' the learning rate and entropy regularisation parameter during the course of training, with each training run consisting of a population of 30 agents. For the LSTM agents, learning rates were sampled log-uniformly from the interval $[1, 4) \times 10^{-4}$ while the entropy regularisation parameter was sampled log-uniformly from the interval $[1, 5) \times 10^{-2}$. For the BAD agents, learning rates were sampled log-uniformly from the interval $[9 \times 10^{-5},\ 3 \times 10^{-4})$ while the entropy regularisation parameter was sampled log-uniformly from the interval $[3, 7) \times 10^{-2}$. Agents evolved within the PBT framework by copying weights and hyperparameters (plus perturbations) according to each agent's rating, which was an exponentially moving average of the episode rewards with factor 0.01. An agent was considered for copying roughly every 200M steps if a randomly chosen copy-to agent had a rating at least 0.5 points higher. To allow the best hyperparameters to manifest sufficiently, PBT was turned off for the first 1B steps of training.

The BAD agent was trained with 100 self-consistent iterations, a V1 mix-in of $\alpha = 0.01$, BAD discount factor $\gamma_{\text{BAD}} = 1$, inverse temperature 1.0, and 3000 sampled hands. Since sampling from card-factorised beliefs can result in hands that are not compatible with the deck, we sampled 5 times the number of hands and accepted the first 3000 legal hands, zeroing out any hands that were illegal.

B. Self-Consistent Belief Approximation for Hanabi

We will use the same notation as in the main text: "f_t^{pub} consists of a vector of 'candidates' C containing counts for all remaining cards, and a 'hint mask' HM, an $AN_h \times N_{\text{color}}N_{\text{rank}}$ binary matrix that is 1 if in a given 'slot' the player could be holding a specific card according to the hints given so far, and 0 otherwise". Furthermore, $\mathcal{L}(f[i])$, is the marginal likelyhood.

Then the basic per-card belief is simply:

$$B^0(f[i]) \propto C(f) \times \text{HM}(f[i]) \times \mathcal{L}(f[i]), \tag{16}$$

$$B^0(f[i]) = \frac{C(f) \times \text{HM}(f[i]) \times \mathcal{L}(f[i])}{\sum_g C(g) \times \text{HM}(g[i]) \times \mathcal{L}(g[i])} \tag{17}$$

$$= \beta_i \big(C(f) \times \text{HM}(f[i]) \times \mathcal{L}(f[i])\big). \tag{18}$$

In the last two lines we are normalising the probability, since the probability of the i-th feature being one of the possible values must sum to 1. For convenience we also introduced the notation β_i for the normalisation factor.

Next we apply the same logic to the iterative belief update. The key insight here is to note that conditioning on the features $f[\text{-}i]$, i.e., the other cards in the slots, corresponds to reducing the card counts in the candidates. Below we use $M(f[i]) = \text{HM}(f[i]) \times \mathcal{L}(f[i])$ for notational convenience:

$$\mathcal{B}^{k+1}(f[i])$$

$$= \sum_{f[-i]} \mathcal{B}^k(f[\text{-}i]) P(f[i] | f[\text{-}i], f^{\text{pub}}_{\leq t}, u^a_{\leq t}, \hat{\pi}_{\leq t}) \tag{19}$$

$$= \sum_{g[-i]} \mathcal{B}^k(g[\text{-}i]) \beta_i \bigg(C(f) - \sum_{j \neq i} \mathbb{1}(g[j] = f) \bigg) M(f[i]). \tag{20}$$

In the last line we relabelled the dummy index $f[\text{-}i]$ to $g[\text{-}i]$ for clarity and used the result from above. Next we substitute the factorised belief assumption across the features, $\mathcal{B}^k(g[\text{-}i]) = \prod_{j \neq i} \mathcal{B}^k(g[j])$:

$$\mathcal{B}^{k+1}(f[i])$$

$$= \sum_{g[-i]} \mathcal{B}^k(g[\text{-}i]) \beta_i \bigg(C(f) - \sum_{j \neq i} \mathbb{1}(g[j] = f) \bigg) M(f[i]) \tag{21}$$

$$= \sum_{g[-i]} \prod_{j \neq i} \mathcal{B}^k(g[j]) \beta_i \bigg(C(f) - \sum_{j \neq i} \mathbb{1}(g[j] = f) \bigg) M(f[i]) \tag{22}$$

$$\simeq \beta_i \sum_{g[-i]} \prod_{j \neq i} \mathcal{B}^k(g[j]) \bigg(C(f) - \sum_{j \neq i} \mathbb{1}(g[j] = f) \bigg) M(f[i]). \tag{23}$$

In the last line we have ommited the dependency of β_i on the sampled hands $f[\text{-}i]$. It corresponds to calculating the average across sampled hands first and then normalising (which is approximate but tractable) rather than normalising and then averaging (which is exact but intractable). We can

now use product-sum rules to simplify the expression.

$$\mathcal{B}^{k+1}(f[i])$$

$$\simeq \beta_i \left(C(f) - \sum_{g[-i]} \prod_{j \neq i} \mathcal{B}^k(g[j]) \sum_{j \neq i} \mathbb{1}(g[j] = f) \right) M(f[i]) \tag{24}$$

$$= \beta_i \left(C(f) - \sum_{j \neq i} \sum_{g} \mathcal{B}^k(g[j]) \mathbb{1}(g[j] = f) \right) M(f[i]) \tag{25}$$

$$= \beta_i \left(C(f) - \sum_{j \neq i} \mathcal{B}^k(f[j]) \right) M(f[i]) \tag{26}$$

$$\propto \left(C(f) - \sum_{j \neq i} \mathcal{B}^k(f[j]) \right) M(f[i]). \tag{27}$$

This concludes the proof.