Supplementary Material for 'Bayesian Action Decoder for Deep Multi-Agent Reinforcement Learning'

Jakob N. Foerster [*1,2] **H. Francis Song** [*3] **Edward Hughes** [3] **Neil Burch** [3] **Iain Dunning** [3] **Shimon Whiteson** [1] **Matthew M. Botvinick** [3] **Michael Bowling** [3]

1. Parameterising and Sampling from the Distribution over Partial Policies

BAD requires us to parameterise a probability distribution over partial policies using a deep neural network:

$$P(\hat{\pi}|s_{\text{BAD}}) = \pi^{\theta}_{\text{BAD}}(\hat{\pi}|s_{\text{BAD}}). \tag{1}$$

The first insight is that we can trivially use a neural network to map from public states, s_{BAD}, into probabilistic partial policies. To do so, we simply start with a feedforward policy that takes as input both s_{BAD} and f^a and produces a distributions over actions:

$$\pi^{\theta}(s_{\text{BAD}}, f^a) \rightarrow P(u|s_{\text{BAD}}, f^a). \tag{2}$$

Next, we note that if we fix a given s_{BAD}, we now have a probabilistic partial policy which maps each private observation f^a into a probability distribution over actions. This partial policy is produced deterministically as a function of s_{BAD} via the parameters θ:

$$\pi(u|f^a) : \{f^a\} \rightarrow \{P(\mathcal{U})\} \; |s_{\text{BAD}}, \tag{3}$$

$$\pi(u|f^a) = \pi^{\theta}(u|s_{\text{BAD}}, f^a). \tag{4}$$

Now, this is close, but not quite what we want. Above we have a deterministic map from s_{BAD} into probabilistic partial policies, $\pi(u|f^a)$. Instead, we require a differentiable distribution over deterministic partial policies.

Perhaps surprisingly, this can be accomplished by conditioning the sampling from $\pi(u|f^a)$ on a common knowledge random seed, ξ:

$$\hat{\pi} : \{f^a\} \rightarrow \mathcal{U} \; |s_{\text{BAD}}, \tag{5}$$

$$\{f^a\} \rightarrow u \sim \pi(u|f^a) \; |\xi, \tag{6}$$

$$\{f^a\} \rightarrow u \sim \pi^{\theta}(u|s_{\text{BAD}}, f^a) \; |\xi. \tag{7}$$

$$\tag{8}$$

Thus, when we sample ξ we are effectively sampling an entire deterministic partial policy.

2. 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.

In the V0-LSTM and V1-LSTM BAD 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 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) 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 Hanabi 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

[*]Equal contribution [1]University of Oxford, UK [2]Work done at DeepMind. JF has since moved to Facebook AI Research, Menlo Park, USA. [3]DeepMind, London, UK. Correspondence to: Jakob Foerster <jnf@fb.com>, Francis Song <songf@google.com>.

Proceedings of the 36^{th} International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019.

each turn the only allowed action was the 'no-action'. 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.

3. 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:

$$B^0(f[i]) \propto C(f) \times \text{HM}(f[i]) \times \mathcal{L}(f[i]), \tag{9}$$

$$B^0(f[i]) = \frac{C(f) \times \text{HM}(f[i]) \times \mathcal{L}(f[i])}{\sum_g C(g) \times \text{HM}(g[i]) \times \mathcal{L}(g[i])} \tag{10}$$

$$= \beta_i \big(C(f) \times \text{HM}(f[i]) \times \mathcal{L}(f[i])\big). \tag{11}$$

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:

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

$$= \sum_{f[\text{-}i]} \mathcal{B}^k(f[\text{-}i]) P(f[i] | f[\text{-}i], f^{\text{pub}}_{\leq t}, u^a_{\leq t}, \hat{\pi}_{\leq t}) \tag{12}$$

$$= \sum_{g[\text{-}i]} \mathcal{B}^k(g[\text{-}i]) \beta_i \Big(C(f) - \sum_{j \neq i} \mathbb{1}(g[j] = f) \Big) M(f[i]). \tag{13}$$

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, $\mathcal{B}^k(g[\text{-}i]) = \prod_{j \neq i} \mathcal{B}^k(g[j])$:

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

$$= \sum_{g[\text{-}i]} \mathcal{B}^k(g[\text{-}i]) \beta_i \Big(C(f) - \sum_{j \neq i} \mathbb{1}(g[j] = f) \Big) M(f[i]) \tag{14}$$

$$= \sum_{g[\text{-}i]} \prod_{j \neq i} \mathcal{B}^k(g[j]) \beta_i \Big(C(f) - \sum_{j \neq i} \mathbb{1}(g[j] = f) \Big) M(f[i]) \tag{15}$$

$$\simeq \beta_i \sum_{g[\text{-}i]} \prod_{j \neq i} \mathcal{B}^k(g[j]) \Big(C(f) - \sum_{j \neq i} \mathbb{1}(g[j] = f) \Big) M(f[i]). \tag{16}$$

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.

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

$$\simeq \beta_i \Big(C(f) - \sum_{g[\text{-}i]} \prod_{j \neq i} \mathcal{B}^k(g[j]) \sum_{j \neq i} \mathbb{1}(g[j] = f) \Big) M(f[i]) \tag{17}$$

$$= \beta_i \Big(C(f) - \sum_{j \neq i} \sum_g \mathcal{B}^k(g[j]) \mathbb{1}(g[j] = f) \Big) M(f[i]) \tag{18}$$

$$= \beta_i \Big(C(f) - \sum_{j \neq i} \mathcal{B}^k(f[j]) \Big) M(f[i]) \tag{19}$$

$$\propto \Big(C(f) - \sum_{j \neq i} \mathcal{B}^k(f[j]) \Big) M(f[i]). \tag{20}$$

This concludes the proof.

4. Anecdotal Analysis

Below we present commentary from David Wu (https://github.com/lightvector/), the creator of the FireFlower bot, on our BAD agent. While this is anecdotal evidence, we believe it provides some interesting insights into the gameplay that our BAD agent discovers. The comments are taking verbatim from an email exchange with David:

4.1. Communicating Playables

- As you observed before, the bot uses R and Y often to hint newest-card-playability.
- In addition to the R and Y hints, it also often uses direct hints to the newest card to indicate playability, in the way that natural human conventions do, and I think these include both color and number hints.
- When the R and Y hints or direct hints to the newest card hit multiple cards, the bot often was indicating multiple plays. In the small sample size of cases we looked over, it tended to be the case that the R/Y hints were more often "play in the order from newest to oldest" while the direct hints were more often in the order of "play from oldest to newest". I think this was not 100% consistent though, but in all cases when looking at the direct beliefs, it was clear that in each case there was a strong ordering convention was in force for that hint, it's just that we didn't see enough cases to be able to determine the precise rules for which one when. Generally though, it makes a lot of sense to vary the ordering convention in different parts of the hint space to add flexibility in hinting.
- The bot uses certain other kinds of direct hints to older cards to suggest that those cards are one step away from playable, or something of that nature. Sometimes the belief state shows that this is not absolutely certain, but over time as other things happen the probability mass sometimes gradually updates and concentrates on the card on the truth, such that once the preceding card is played, the bot may then play the formerly-one-step-away card without any further suggestion.
- For these "delayed" one-step-removed hints to older cards, there is also a similar variation in ordering conventions in the case those hints hit more than one card, sometimes they're in "age-order" and sometimes they're in "reverse-age-order".
- Commonly the R and Y hints also indicate other plays or delayed plays besides the play of the newest card. The bot chooses the manner of hinting the first card as playable (R vs Y vs direct hint) to try to communicate other useful information at the same time, if possible.
- I think occasionally the bot seems to "single out" a card by directly hinting all other cards in the hand *besides* that card over successive turns, and sometimes this implies that the singled-out unhinted card is playable. I'm not sure on this one though, I'd need to see more cases.
- I think there seems to be some interesting other conventions that seem to function to give information to allow play of older red and yellow cards, which are necessary since direct hints of R and Y mean to play the newest card rather than the card hinted.

4.2. Communicating Protection

- As you observed before, the bot discards its newest card by default.
- G hints that do not directly hit the first card appear to mean that the newest card is dangerous and should not be discarded. Possibly it is more specific, and actually just means that it's a 5, the examples I recall all involved 5s. The bot also can just directly hint the newest card in various ways.
- The bot is very aggressive about protecting the newest card if the newest card is a 5 or otherwise dangerous (the last copy in the deck), whether by giving a G hint, or a direct hint, or otherwise. This is so consistent that pretty much any action other than an immediate protection causes the other bot to infer that the newest card is NOT a 5 or the last copy of a card whose first copy has been lost.
- However, the bot does *not* do this any longer if there is a common-knowledge-extremely-safe discard in that player's hand (e.g. a redundant copy of a card already played). In that case, it is understood that the bot will prefer to discard that instead. Then, protection of the newest card is not necessarily urgent any more, and neither will a player necessarily infer that the newest card is safe from a failure by their partner to protect it immediately.
- There seems to be some interesting dictionary of hints that we haven't worked out yet about ways to signal to discard cards besides the newest, which prevents junk from accumulating in the hand as non-playable but useful-to-hold-on-to cards enter the hand.

Miscellaneous Communication

- Often the hints, and sometimes its other actions just come "attached" with miscellaneous information. The most extreme example is I observed one game where as a result of the bots discarding, it was immediately implied that a particular card in the other player's hand was almost certainly red. This information was not immediately useful (the red card was not yet playable, nor was it likely to have been discarded soon), it was simply just extra information attached to the action of discarding in that particular case. Presumably the bot was by convention constrained to almost certainly do some other action in that situation had that partner's card counterfactually not been red.

- This kind of extra not-immediately-useful "attached" information is perhaps the most non-human part of the bot's convention set. But actually it doesn't happen as often as one might expect from a "nonhuman" agent. For the most part I didn't see this all that much for plays and discards (that one extreme example notwithstanding). This makes sense, as having too many such conventions would overly constrain the ability of the players to act, as discard/play are both critical actions you need to take very frequently regardless of the other player's hand.

- Even for hint actions, most hints were very sensible and humanly explainable, or clearly appeared that they would be humanly explainable had we had a larger sample size so that we could be surer about the generality of its meaning and exactly how the bot had packed different meanings into the hint space. There were only a few hint actions that I found particularly "weird" in what inference was made.

- A priori, there's no particular reason why a bot's conventions couldn't, for example, completely change depending on the turn number modulo 3, and be extremely hard for humans to comprehend. But for the most part, the conventions of this bot weren't like that - they were pretty understandable, or at least seemed consistent and sensible even if we didn't have all the exact meanings mapped out.

Overall Quality of Play and Game Flow

- The bot is *very* strong in the early game, and there its convention set is overall far more efficient than "natural" human convention sets (although not-necessarily human convention sets that were constructed to be more artificial and encoding-like). It's really quite beautiful.

- The bot is superhumanish at tracking inferred information over time, e.g. on the one hand inferring that a card is not scary in first position, then as it drifts back later in the hand, inferring this or that other property incidentally, and inferring based on the "aging" of the card that it is probably not this or that, and so on, until only a couple possibilities remain. It's not uncommon that in the midgame, both players know almost all the relevant things about their hands.

- The bot might be tactically weak in occasional situations on or near 0 hints, where the ensuing sequence of actions is heavily constrained. It seems to have a very strong preference to discard and get away from 0 hints, even when as far as we can tell based on its convention set it should be possible to just stay at 0 hints and play out some cards, and where discarding at that moment is suboptimal. For example if the ensuing sequence of plays would result in a few 5s being played thereby recovering some hints for free, and the partner's immediate discard is also completely safe in the event that the partner wants to discard, whereas one's own discard unnecessarily loses a copy of a card that could be useful in the future. (If I read the paper right, there is no explicit lookahead in this bot?).

- The bot makes a few seemingly-clear mistakes in the endgame (as far as we can tell), although only slight ones. For example, one of the games we looked at:

  - The players were in a close-to-winning state - they both knew all the playable cards in their hands or had inferred them with high confidence, and all they needed to do was play those cards and wait to draw the few remaining cards to play.
  - They had plenty enough hints and headroom to theoretically execute essentially-perfect play thereafter (i.e. getting plays out in a timely manner, collectively never discarding any card that could be useful thereafter, optimizing who draws the next card for parity, etc), and by my understanding of their conventions, nothing stopping them from doing so.
  - But instead of playing, the bot wasted a turn giving their partner a hint. When you inspected the V2 belief state, it gave no useful information - the dominant effect of the hint was actually to concentrate probability mass *away* from the truth giving the partner a misleading belief about a card, and had almost no other effects.
  - Their partner then proceeded to also not play and instead discarded their newest card, which unnecessarily lost one of the copies of a useful 4. There was a copy of the 4 left in the deck, but such a discard is still bad. If the remaining copy of that card is the very bottom card of the deck, it guarantees

that you cannot get 25 points, so every unnecessary discard of the first copy of any card loses you EV due to the chance for the other copy to be the last card.

Speculating a little here - perhaps something about the bot's policy or convention set hasn't converged as sharply in the endgame? It's certainly the case that the gradient there is much smaller - even a clear mistake near the end tends to cost you only a little in EV if you're measuring by score, whereas near the start of the game it can cost you a lot. And the expected penalty for discarding the first copy of a useful 4 when otherwise well ahead is slight, since it then usually only harms you when that 4 is precisely the last card in the deck which only happens 1/N times, so one might imagine the average gradient there for good behavior to be very small.

References

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.

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.