Learning to Play Games with PlaNet

Lukas Valatka

Vilnius University, Faculty of Mathematics and Informatics,
Institute of Computer Science, Didlaukio str. 4, LT-08303 Vilnius
lukas.valatka@mif.stud.vu.lt

Summary. An evaluation of a recent state of the art model-based reinforcement learning PlaNet in a gaming environment is presented. Author analyzes PlaNet capabilities to solve several problems in Atari and VizDoom domains. Author identifies that PlaNet's observation and reward encoders have trouble capturing small details in Atari games (Pong, Breakout), often critical to the agent's performance playing games. Hyperparameter tuning strategy is suggested. Author confirms latent overshooting is crucial for VizDoom Take Cover scenario, implying it is necessary for similar environments. This suite of experiments was carried out as a preparatory work for future PlaNet's evaluation to handle simultaneous control of multiple agents in games.

Keywords: PlaNet, Model-based reinforcement learning, Latent space planning, Atari gym suite, VizDoom.

1 Introduction

Deep Planning Network (PlaNet) is a novel (published 12 Nov 2018) deep reinforcement learning algorithm developed by Google Research and Deep Mind [2]. PlaNet is a model-based RL algorithm. Being such, it achieves th same accuracy on the DeepMind open source control suite as state of the art model-free D4PG algorithm [3] more efficiently, which is accomplished by learning environment dynamics and using the model to plan actions. Sample efficiency, as well as possibility to re-use the model between related tasks makes PlaNet an interesting empyric research candidate.

Playing games is an established way to benchmark reinforcement learning algorithms. Games are simplified models of the real world. They feature things such as multiple agents, stochastic dynamics, partial

observability – problems common to our everyday lives. Making algorithm solve problems encapsulating part or all of these components is a step towards making reinforcement learning algorithms solve real world problems [5].

Motivated by the promises of PlaNet, author formulates an objective to evaluate PlaNet's capabilities to solve reinforcement learning game problems. This work encapsulates authors findings adapting PlaNet to solve several popular discrete OpenAI gym Atari [6] and VizDoom [7] scenarios.

2 PlaNet Algorithm

PlaNet is a model-based reinforcement learning algorithm. It uses a learned model to generate most rewarding action sequences. In this section the author describes the mechanics of PlaNet in detail.

Environment

PlaNet models an environment as a partially observable Markov decision process (POMDP). That is, PlaNet is unsure which state the model might be in, yet models a separate mechanism to increase this certainty.

Model

PlaNet is not given a model. It learns environment dynamics by interacting with its suroundings. It records observation, action, reward triplets and learns from them. Consider h_t to be a deterministic latent state, s_t to be a stochastic latent state, o_t to be an observation, a_t to be an action and r_t to be a reward, all at timestep t. PlaNet learns models as follow:

- Transition models
  - Deterministic state model: $h_t = f(h_{t-1},\ s_{t-1},\ a_{t-1})$. Implemented as a recurrent neural network (GRU).
  - Stochastic state posterior model: $s_t \sim p\,(s_t \mid h_t,\ o_t)$. Implemented as a feed-forward neural network parameterized Gaussian distribution.
  - Stochastic state (prior) model: $s_t \sim p\,(s_t \mid h_t)$. Implemented as a feed-forward neural network parameterized Gaussian distribution.

- Observation model: $o_t \sim p\,(o_t \mid s_t)$. Implemented as a Gaussian with mean parameterized by deconvolutional neural network and identity variance.
- Reward model: $r_t \sim p\,(r_t \mid s_t)$. Implemented as a feed-forward neural network parameterized Gaussian.

Observation model is only used for training. Transition models and the reward model are used for planning. Posterior models increases the certainty which latent state the agent currently resides in.

Planning

PlaNet uses cross-entropy method to search for best action sequence under the model [2]. Author suggests reading original PlaNet paper for the details of the planning algorithm. The procedure can be summarized as follows:

1. Infer a belief of a latent state from current observation and past deterministic latent state via a stochastic state posterior model $s_t \sim p\,(s_t \mid h_t,\, o_t)$.
2. Initialize *H* Normal distributions (i.e. time dependent Gaussian), each distribution for each action in the to-be-planned action sequence vector. *H* is a planning horizon.
3. Sample *J* action sequences from the distribution (all *H* length). Actions are sampled by unrolling the stochastic state prior model in time, which does not use encoded observations, thus being a key ingredient to fast planning $s_{t:H} \sim p\,(s_{t:H} \mid h_{t:H-1})$.
4. Re-fit the belief (i.e. action sequence distribution) to best *K* action sequences, based on the sum reward of the sequence of actions (using a reward model).
5. Repeat for *I* iterations.

Lastly, return the first action of the returned final mean action sequence. After receiving a new observation, agent replans again, which helps to avoid local optima.

Learning

The algorithm utilises two objective functions to learn the model – a *reconstrution* objective and a regularizing *latent overshooting* objective. Below is the objective function for the observation model – reward model learning objective follows the same analogy.

$$\max\left(\sum_{t=1}^{T}\left(E_{q(s_t \mid o \le t)}[\ln p\,(o_t \mid s_t)]\right) - \frac{1}{D}\sum_{d=1}^{D}\beta_d E[KL[q(s_t \mid o_{\le t}, a_{<t}) \,||\, p(s_t \mid s_{t-1}, a_{t-1})]]\right)$$

reconstruction — latent overshooting

Reconstruction term forces the model to learn to encode rewards and observations into the latent space and as accurately reconstruct them as possible.

Latent overshooting term works as a regularizer – it encourages consistency between one-step and multi-step predictions, which should be equal in expectation over the training dataset. It is trained on all distances until D (50 by default).

The goal is to maximize this combination of objective functions. Note: the final objective includes the regularization term of latent space distribution $p(s_t)$ as well (author omits it here for readability), a common regularization technique used in variational autoencoders [1].

3 Experiments

This section contains the evaluation results and discussions of PlaNet on Atari and VizDoom OpenAI gym environments.

Experiments with Atari Breakout and Pong

Pong and Breakout games is a popular way to benchmark reinforcement learning algorithms [9]. In a Pong game, the agent has to control a paddle with up and down actions to score points (i.e. play a similar to tennis game). In Breakout, the agent can move either left or right. The goal is to not let a game ball this the ground, as well as hit as many bricks as possible.

Concretelly, the author used OpenAI Gym BreakoutNoFrameskip-v4 and PongNoFrameskip-v4 environments to remove implicit stochastic action repeat and frameskip [8]. The environment was configured as is

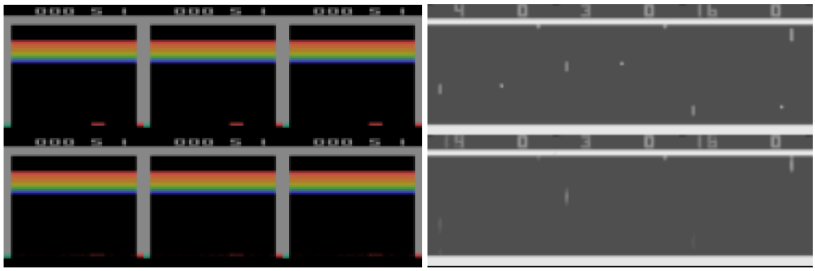
it is configured by OpenAI gym baselines for Atari games [9]. Pong game observations were grayscaled before being inputted to the model to reduce the computational cost.

Results

Table 1. PlaNet's evaluation on Atari Breakout and Pong games summary.

Episodes	Game	Planet score	Gym leader [9]	Minimum score	Maximum score
1500	Breakout	1.5	10	0	+INF
1500	Pong	-20	-10	-20	20

Figure 1. Zero-step reconstruction from Breakout and Pong games, respectivelly. Top rows show ground truth agent observations, bottom rows – decoded posterior state.



Latent space encoding issue: Poor Table 1 results imply issues with PlaNet's default configuration. Figure 1 suggests the latent space is not capable to encapsulate small details. Author observed the reconstruction loss is not affected by small critical objects e.g. a ball. Author observed that the reward model is innaccurate even till 10 – 15 steps into the future, which implies the algorithm cannot perform correct planning (as the default PlaNet's planning horizon is 12).

After extensive discussions with one of the PlaNet's authors Danijar Hafner, author suggests reducing regularization impact of latent overshooting objective: reducing the divergence scale and global prior scale parameters. Author validated that this makes small details visible in much fewer episodes. Increasing free nats parameter increases the lower limit of divergence losses, which helps the model retain small, seemingly unimportant observation details in the latent space.

Experiments with VizDoom

After incorporating all the changes, based on the findings from the Atari game suite experiments, the author decided to begin experiment in the VizDoom domain.

VizDoom is a reinforcement learning algorithm evaluation environment, based on an open-source clone of a once popular first person shooter Doom. VizDoomTakeCover-v0 gym environment was chosen. Agent has two actions – left and right. It is rewarded with +1 for each tick of an episode. Episode ends when the agent losses all lives, which can be lost by getting collided with fireballs, being shooted by monsters on the opposite side of the game room.

Table 2. PlaNet's evaluation on VizDoom TakeCover scenario.

Episodes	Game	Overshooting loss horizon	Planet score	World Model's experiment (10 000 episodes) [4]	Minimum score	Maximum score
1500	VizDoom TakeCover	50	488 +- 250	820 +- 58	0	+INF
1500	VizDoom TakeCover	25	385 +- 128	820 +- 58	0	+INF

Overshooting loss importance: Table 2 result suggest that decreasing overshooting loss horizon decreases performance. It is clear that decreasing the regularization horizon makes the model less robust. This was discussed by the PlaNet's authors as well [2].

4 Conclusion

Author evaluated PlaNet algorithm on two game environments – Atari Breakout and Pong, and VizDoom. Results suggest that default PlaNet's configuration suffers from small visual artefact loss in the latent space, often crucial to the agent's performance. The default configuration, although forcing reconstructions to be robust, overregularizes the model. Although it is important to reduce overshooting loss scale, the horizon is suggested to be kept long to achieve better performance. Tuned according to these findings, PlaNet algorithms is a suitable candidate for sample efficient learning to solve game problems.

References

[1] Kingma, D. P., & Welling, M. (2013). Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114.

[2] Danijar Hafner, Timothy P. Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee ir James Davidson. Learning latent dynamics for planning from pixels. CoRR, abs/1811.04551, 2018.

[3] Thanh Thi Nguyen and Ngoc Duy Nguyen and Saeid Nahavandi. Deep Reinforcement Learning for Multi-Agent Systems: A Review of Challenges, Solutions and Applications. CoRR, abs/1811.04551, 2018.

[4] David Ha and Juergen Schmidhuber (2018). World Models. CoRR, abs/1803.10122.

[5] Open AI. Artificial life: objective and approach. https://docs.google.com/ document/d/1_76rYTPtPysSh2_cFFz3Mfso-9VL3_tF5ziaIZ8qmS8/edit. 2018.

[6] M. G. Bellemare, Y. Naddaf, J. Veness ir M. Bowling. The arcade learning environment: an evaluation platform for general agents. Journal of Artificial Intelligence Research, 47:253–279, 2013-06.

[7] Michal Kempka, Marek Wydmuch, Grzegorz Runc, Jakub Toczek ir Wojciech Jas-kowski. Vizdoom: A doom-based AI research platform for visual reinforcement learning. CoRR, abs/1605.02097, 2016. arXiv:1605.02097. http://arxiv.org/abs/1605.02097.

[8] Bongsang Kim. OpenAI Gym Environment Full List . 2018. https://medium.com/@researchplex/openai-gym-environment-full-list-8b2e8ac4c1f7.

[9] Dhariwal, Prafulla and Hesse, Christopher and Klimov, Oleg and Nichol, Alex and Plappert, Matthias and Radford, Alec and Schulman, John and Sidor, Szymon and Wu, Yuhuai and Zhokhov