ATLANTIS PRESS
International Journal of Computational Intelligence Systems, Vol. 12 (2018) 164-171

Bayesian Deep Reinforcement Learning via Deep Kernel Learning

Junyu Xuan, Jie Lu, Zheng Yan, Guangquan Zhang

Centre for Artificial Intelligence, University of Technology Sydney
Sydney, NSW 2007, Australia
E-mail: {Junyu.Xuan, Jie.Lu, Yan.Zheng, Guangquan.Zhang}@uts.edu.au

Received 15 October 2018

Accepted 26 October 2018

Abstract

Reinforcement learning (RL) aims to resolve the sequential decision-making under uncertainty problem where an agent needs to interact with an unknown environment with the expectation of optimising the cumulative long-term reward. Many real-world problems could benefit from RL, e.g., industrial robotics, medical treatment, and trade execution. As a representative model-free RL algorithm, deep Q-network (DQN) has recently achieved great success on RL problems and even exceed the human performance through introducing deep neural networks. However, such classical deep neural network-based models cannot well handle the uncertainty in sequential decision-making and then limit their learning performance. In this paper, we propose a new model-free RL algorithm based on a Bayesian deep model. To be specific, deep kernel learning (i.e., a Gaussian process with deep kernel) is adopted to learn the hidden complex action-value function instead of classical deep learning models, which could encode more uncertainty and fully take advantage of the replay memory. The comparative experiments on standard RL testing platform, i.e., OpenAI-Gym, show that the proposed algorithm outweighs the DQN. Further investigations will be directed to applying RL for supporting dynamic decision-making in complex environments.

Keywords: Reinforcement learning, Uncertainty, Bayesian deep model, Gaussian process

1. Introduction

Reinforcement learning (RL)[22], as an important branch of machine learning, aims to resolve the sequential decision-making under uncertainty problems where an agent needs to interact with an unknown environment with the expectation of optimising the cumulative long-term reward. A motivating example is the self-driving car. If we consider a car as the agent and the traffic conditions (e.g., traffic light signal and speed limitation) as the environment, RL can endow the car the ability to autonomously and safely choose actions (e.g., breaking, turning, and lighting) corresponding to difference traffic conditions when driving on road. Since the environment is unknown to the car, such ability needs to be learned through the so-called 'trail-and-error' strategy that is to learn an optimised action-selection policy through interacting with this environment[7]. Thus, there are two goals for the agent during the interaction with the environment: *exploration* which aims to perceive more about the environment and *exploitation* which aims to optimise the policy under current information about the environment. Apart from self-driving cars, many real-world tasks could be formalised as sequential decision-making under uncertainty problems and then benefit from RL, such as: industrial robotics[9], medical treatment[11], and trade execution[14].

According to whether maintaining a computa-

tional model for the environment, existing RL algorithms could be categorised as being either model-based or model-free. Model-based RL algorithms improve the policy learning through modelling the environment, e.g., simulating the interactions with the environment model in hand or planning ahead using the model. Due to the lack of such environment model, model-free RL cannot infer the effect from environment dynamics and efficiently explore the space. However, for some complex environments with large scale states, it is too difficult to accurately build their models. Model-free RL algorithms directly learn the policy without requiring an explicit environment model. Recently, one seminal model-free RL algorithm, i.e., deep Q-network (DQN)[13], has achieved great success on Atari games and substantially promote the development of RL. The advanced ability of DQN comes from two aspects: 1) *deep learning models*; DQN adopts deep neural networks to replace the traditional Q-table for the hidden action-value function learning. For complex environments with large or even near infinite number of states, Q-table is not appropriate due to the large-scale storage and high computation requirements; 2) *experience replay*; The historical interactions are saved as replay memory and sampled in DQN to stochastically update deep neural networks, which can reduce the dependency between sequential interactions and then increase the efficiency of the algorithm. However, the classical deep neural networks used in DQN cannot handle uncertainty well due to its deterministic nature. As defined, RL aims to handle sequential decision-making under uncertainty, so how to capture and deal with uncertainty is essential for RL. Besides, all the saved historical interactions are equally treated in DQN, but they are in fact different for model training and convergence.

In this paper, we propose a new model-free RL algorithm based on a Bayesian deep model - deep kernel learning[28], which is able to encode more uncertainty and fully take advantage of the saved historical interactions. Specially, the deep kernel learning is a Gaussian process (GP)[16] with a deep kernel modelled by a deep neural network, which has both advantages of deep neural network and GP. We adopt the deep kernel learning to replace the original deep neural network, which could encode more uncertainty. For example, the prediction of action-value function from this algorithm is not just the action but also the variance of such action. Further, a new weighted sampling strategy is developed. The prediction variance from deep kernel learning is used as the weight for each historical interaction, and then the model is updated using the samples from weighted interactions. The experimental evaluation on standard RL testing platform -OpenAI.Gym* has demonstrated that the new algorithm could reach larger scores. In summary, the main two contributions of this article are as follows:

- we propose a new model-free RL algorithm based on deep kernel learning with better performance comparing the classical deep neural network-based one;
- a new weighted sampling strategy is developed to fully take advantage of the replay memory to further improve the algorithm performance.

The remainder of this article is organised as follows. Section 2 discuses related work. The new model-free RL algorithm is introduced in Section 3. Section 4 evaluates the proposed RL algorithm on standard RL testing platform through comparing the classical algorithm. Section 5 concludes this study and discusses possible future work.

2. Related Work

In this section, we briefly review the related work of this study. The first part summarizes the literature on model-free reinforcement learning and the second part summarizes the literatures on Bayesian deep models.

2.1. Model-free RL

Model-free RL aims to learn an optimal policy for an agent without explicitly modelling the environment. Existing works in this direction can be mainly

* https://gym.openai.com/

grouped into three categories: value-based, policy-based, and combined[22]. Policy-based RL is to directly optimise the parameters of the policy aiming to maximise the expected reward, and policy gradient is a typical algorithm of this category. Value-based RL is to learn the action-value function first aiming to minimise the temporal difference error and then design the policy based on this function. According to the difference on the computation of temporal difference error, there are two kinds in value-based RL: *on-policy* one which computes the temporal difference error based on current policy, e.g., SARSA; *off-policy* one which computes the temporal difference error based on greedy policy, e.g., Q-learning. Deep Q-Networks (DQN)[13] is a seminal work of Q-learning, which adopts deep neural network to approximate the Q-function. Such algorithm is proofed successful on Atari games and even exceeds the human performance. Based on DQN, Double-DQN[24] is proposed to separately model action-selection and Q-function by two deep neural networks, which resolves the possible overestimate problem in DQN. The performance can be further improved by splitting Q-function into state function and advantage function in Dueling DQN[26]. There are also works that try to combine two categories, e.g., Deep Deterministic Policy Gradient[10]. This paper will only target on DQN and replace the traditional deep neural network with Bayesian deep models, and other advanced extensions can also be easily adopted later.

2.2. *Bayesian deep model*

The Bayesian paradigm for machine learning, also known as Bayesian (machine) learning, is to apply probability theories and techniques to represent and learn knowledge from data. Some renowned examples are Bayesian network, Gaussian mixture models, hidden Markov model[15], Markov random field, conditional random field, and latent Dirichlet allocation[1]. Compared to other learning paradigms, Bayesian learning has distinctive advantages: 1) representing, manipulating, and mitigating uncertainty based on a solid theoretical foundation - probability; 2) encoding the prior knowledge about a problem; 3) good interpretability thanks to its clear and meaningful probabilistic structure. Inspired by the recent success of deep learning, researchers start to pay attention to build Bayesian deep models[25] aiming to combine the advantages from two areas. One straightforward strategy is to assign probabilistic prior for parameters of traditional deep neural networks, e.g., Gaussian prior [6,4]. Such Bayesian method could avoid some of the pitfalls of stochastic optimisation. Bayes by Backprop[2] -a variational inference method- is proposed to efficiently resolve these models. Another strategy is to pave the probabilistic models deeply. Deep belief networks [8] may be the earliest one, which has undirected connections between its top two layers (it is actually a RBM[7]) and downward directed connections between all its lower layers. Its advantage is that it needs less time to train the model, but there is no feedback from top to bottom. On the contrary, Deep Bolzman Machine (DBM)[18], which is the multi-layered RBM, requires more time for model training but there is feedback from top to bottom so it is more robust. The hidden units in DBN and DBM are restricted to be binary. To resolve this constraint, Gamma belief networks[29] and deep poisson factor analysis[5] are proposed using Gamma and Negative-binomial distributions, which could build deep structure with nonnegative real hidden units. Deep latent Gaussian model[17] is another deep Bayesian model built by Gaussian distributions. Beyond Gaussian distributions, Gaussian process (GP) is also adopted for constructing Bayesian deep models. For example, deep neural network is used to construct a deep kernel as the covariance function of GP in deep kernel learning[28,27]; the other idea is to pave more Gaussian process on each other known as deep Gaussian process[3].

3. The Proposed Model

RL is often modelled by a Markov decision process as[†]$< S, A, P, R, \gamma >$, where S is a set of environment states, A is a set of agent actions, P defines the state transition of environment, R defines the reward from

[†] https://en.wikipedia.org/wiki/Markov_decision_process

environment, and γ is the discount factor. The goal of RL is the agent policy $\pi(s)$ which is a mapping from environment states to actions aiming to maximise the (discounted) expected long-term reward $\mathbb{E}[\sum_{t=1}^{\infty}\gamma_t r_t]$, where r_t is the reward from t-th interaction.

3.1. Preliminary: Deep Q-Network

Q learning[12] is a methodology to resolve RL by evaluating the value $Q(s,a)$ of action a given state s which is also known as Q-function. Such value could be considered as the expected "reward" from taking this action under this state. To learn this action-value function (a.k.a., Q-function), an effective method is temporal difference learning[21] which minimises the difference between current value of an action and its optimal value according to the next step

$$||Q(s_t,a_t)-(r_t+\gamma_t\max_{a'}Q(s_{t+1},a')||_F \quad (1)$$

where $s_{t+1}=P(s_t,a_t)$ is the next state of environment by taking action a_t. DQN adopts deep neural network (DNN)[20] to model $Q_\theta(s,a)$ where θ denotes the parameters of the deep neural network. With the sequential interaction, θ is continuously optimised using Eq. (1) as the loss function. Another significant contribution of DQN is the experience replay based on a constructed memory which is composed of historical interactions $< s_{t+1},s_t,a_t,r_t >$. When learning the RL algorithm (i.e., training θ), a number of interactions are uniformly sampled from the memory.

3.2. Bayesian deep RL

We propose a Bayesian Deep RL (DBRL) to adopt Gaussian process with deep kernel[28] to model $Q(s,a)$. A Gaussian Process (GP)[16] is defined as a collection of random variables, any finite number of which have (consistent) joint Gaussian distributions

$$f(x)\sim GP(m(x),k_\vartheta(x,x')) \quad (2)$$

where $m(x)$ is the mean function and $k_\vartheta(x,x')$ is a kernel function parameterised by ϑ, e.g., Polynomial kernel and Radial basis function kernel (RBF). GP is often used as the prior for functions and is able to approximate complex and nonlinear functional forms given a number of observations. To further improve the ability on function approximation, deep kernel learning (DKL) incorporates the deep learning idea to the kernel learning

$$k_\vartheta(g_\theta(x),g_\theta(x')) \quad (3)$$

where $g(\cdot)$ is a deep neural network-based data transformation. When applying this deep kernel-GP to RL, we adopt the same framework with DQN and replace the traditional DNN by DKL as illustrated in Fig. 1. The advantage of BDRL comparing DQN is the capability of encoding more uncertainty. That is to say, the prediction from the BDRL is not only the action value but also its variance which could be simple seen as the confidence of this prediction. Next, we will utilise such variance to better take advantage of replay memory.

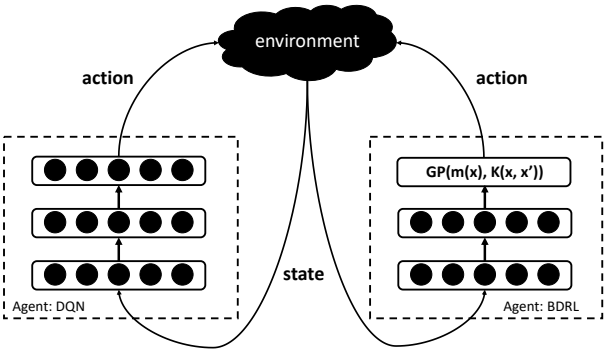

Fig. 1. Interaction between an environment and two agents: DQN and BDRL

Experience replay is another genius in DQN, where historical interactions are saved in memory for Q-function learning. The original memory structure is

$$< s_{t+1},s_t,a_t,r_t > . \quad (4)$$

We add the action uncertainty (i.e., prediction standard deviation) from GP as another column to the memory,

$$< s_{t+1},s_t,a_t,u_t,r_t > . \quad (5)$$

Rather than uniform sampling, weighted sampling is adopted when the historical interactions are retrieved for model training, where each interaction is

weighted by

$$w_i = \frac{(1-\sigma(u_i))^\alpha}{\sum_i (1-\sigma(u_i))^\alpha} \tag{6}$$

where $\sigma()$ denotes the sigmoid function and α controls the contribution of this weight. When $\alpha = 0$, it degenerates to uniform sampling. This weighting schema prefers the interaction with small uncertainty. The whole procedure of the algorithm is summarised in Algorithm 1.

Input: An environment: *env* and n_episode
Output: interactions: $< s_t, a_t, r_t >$ and parameters of Q-function: θ, ϑ
initialization θ, ϑ;
step = 0;
while *n_episode* > 0 **do**
  a, u = agent.chooseAction(s);
  s', r, done = env.step(a);
  agent.store($< s, a, u, r, s' >$);
  agent.updateMemoryWeight() by Eq. (6);
  θ, ϑ = agent.update() using Eq. (1) as loss function;
  $s = s'$;
  if *done* **then**
    s = env.reset();
    n_episode−−;
  end
  step++;
end

Algorithm 1: Bayesian deep RL

4. Experiments

We evaluate BDRL on a open RL testing platform: OpenAI.gym and compare it with DQN in this section.

4.1. Setting

In the following, we choose the simple environment 'CartPole-v1'[‡] in OpenAI.gym. This environment has two actions (i.e., move left or right), its state is a four dimensional vector, and a reward of +1 is provided for every timestep that the pole remains upright. One episode ends when the pole is more than 15 degrees from vertical or the cart moves more than 2.4 units from the center. At first, we take 10,000 steps to explore the state space and construct the initial experience memory, where the action is randomly chosen not from RL algorithms. After the exploring, we take 15,000 steps to train the RL algorithms using ε-greedy strategy with a decreasing ε value and minimum value 0.01. Finally, we test 10,000 steps only using RL algorithms and record the total reward as score for each episode. The higher score means better RL algorithm. The memory size is set as 10,000 and the batch size is 32. Every 150 steps, the fixed target Q-function is updated by the latest evaluation Q-function. The other two hyper-parameters are $\gamma = 0.99$ and learning rate 0.00025 for the optimisor. For DQN, we build a deep neural network that contains two hidden fully connected linear layers with 512 and 128 nodes and activation function is ReLU. For BDRL, we use a deep neural network containing two hidden (linear) layers with 10 nodes in each layer and activation function is tanh, and RBF kernel is used for GP. The implementation of BDRL is based on DKL.[§]

4.2. Results

At first, we compare the performance of DQN and BDRL under same setting where both of them use uniform sampling when retrieving replay memory for deep model update. The result is shown in Fig. 2, where the scores of all episodes in testing stage are plotted. According to this figure, we can observe that: 1) BDRL can reach much higher score than DQN. The highest score from BDRL is 346, which means the car makes 346 moves with the bar upright. 2) The average score of BDRL (113.52) is also better than the one of DQN (98.33). 3) The episode number of BDRL (88) is smaller than the one of DQN (102) within the same number of steps (i.e., 10,000). 4) Although the overall results of BDRL is better than DQN, the line for BDRL is with stronger

[‡] https://gym.openai.com/envs/CartPole-v1/
[§] https://github.com/maka89/Deep-Kernel-GP

fluctuation than the one for DQN, which means the results from BDRL is unstable comparing with the ones from DQN.

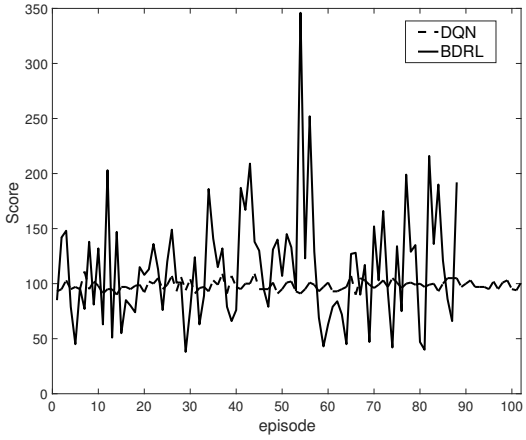

Fig. 2. Evaluation of the efficiency of BDRL through comparing with DQN on 'CartPole-v1'. The scores (the higher is the better) of all episodes in testing stage from both algorithms are plotted. The average scores from BDRL and DQN are 113.52 and 98.33, respectively.

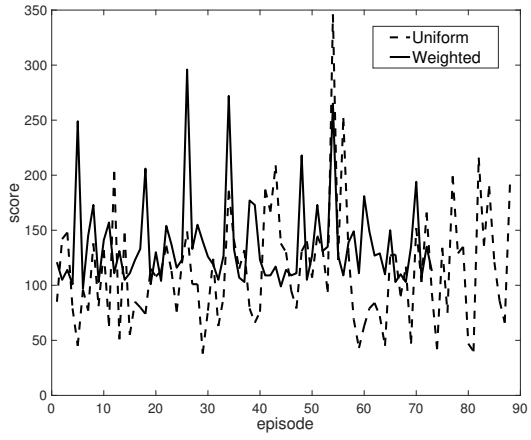

Fig. 3. Evaluation of the efficiency of uncertainty-based weighted sampling strategy for replay memory on 'CartPole-v1'. The scores (the higher is the better) of all episodes in testing stage from BDRL with uniform and weighted strategies are plotted. The average scores from BDRL with weighted and BDRL with uniform are 136.75 and 113.52, respectively.

Then, we evaluate the proposed weighted sampling strategy for replay memory. We run BDRL using two different sampling strategy: uniform and weighted (according to Eq. (6) with $\alpha = 0.1$. Note that during the exploring stage, the action is randomly chosen and their weights are also randomly assigned. The result is shown in Fig. 3, where the scores of all episodes in testing stage are plotted. According to this figure, we can observe that: 1) The episode number of BDRL with weighted (73) is smaller than the one of BDRL with uniform (88) within the same number of steps (i.e., 10,000). 2) The average score of BDRL with weighted (136.75) is also better than the one of BDRL with uniform (113.52). Therefore, we can draw the conclusion that the proposed uncertainty-based weighted sampling strategy could better take advantage of replay memory and improve the efficiency of the BDRL.

5. Conclusions and Further Study

In this study, we have presented a new model-free reinforcement learning (RL) algorithm based on Bayesian deep model. This Bayesian deep RL (BDRL) algorithm has unique advantages compared with deep Q-network (DQN) that is the deep neural network (DNN)-based RL algorithm, i.e., more ability to handle uncertainty and take advantage of past experience. The experiments on standard reinforcement learning testing platform demonstrated its effectiveness on RL tasks, and the comparison with DQN showed the superior performance of the proposed algorithm than the DNN-based one. In this paper, we only use on one simple task to demonstrate the effectiveness of the proposed ideas, and we will test them using more and harder tasks in the future. We believe that this study is only a start point, and there are many interesting following works. One interesting direction for further study would be to add more Gaussian process layers to the deep architecture based on deep Gaussian process[3]; the second interesting direction is to build new Bayesian nonparametric deep models[23] for more flexible Q-function learning with less tunable parameters; the third one is to incorporate the temporal difference error[19] and marginal likelihood from GP into the weights for interactions in replay memory; and the last one is to apply fuzzy systems techniques to the Q-learning process to deal with decision processes[?]

in which the goals and constraints are fuzzy in nature. These future investigations are expected to provide backbones for applying RL techniques to support dynamic decision-making in complex situations that involve massive data domains, agents, and environments.

Acknowledgments

This work is supported by the Australian Research Council (ARC) under Discovery Grant DP170101632.

References

1. David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3(Jan):993–1022, 2003.
2. Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks. *arXiv preprint arXiv:1505.05424*, 2015.
3. Andreas Damianou and Neil Lawrence. Deep gaussian processes. In *Artificial Intelligence and Statistics*, pages 207–215, 2013.
4. Yarin Gal and Zoubin Ghahramani. Bayesian convolutional neural networks with bernoulli approximate variational inference. *arXiv preprint arXiv:1506.02158*, 2015.
5. Ricardo Henao, Zhe Gan, James Lu, and Lawrence Carin. Deep poisson factor modeling. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2800–2808. Curran Associates, Inc., 2015.
6. José Miguel Hernández-Lobato and Ryan Adams. Probabilistic backpropagation for scalable learning of bayesian neural networks. In *International Conference on Machine Learning*, pages 1861–1869, 2015.
7. Geoffrey E Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1771–1800, 2002.
8. Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554, 2006.
9. Abhishek Kumar and Rajneesh Sharma. Linguistic lyapunov reinforcement learning control for robotic manipulators. *Neurocomputing*, 272:84–95, 2018.
10. Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.
11. Ying Liu, Brent Logan, Ning Liu, Zhiyuan Xu, Jian Tang, and Yangzhi Wang. Deep reinforcement learning for dynamic treatment regimes on medical registry data. In *Healthcare Informatics (ICHI), 2017 IEEE International Conference on*, pages 380–385. IEEE, 2017.
12. Francisco S Melo. Convergence of q-learning: A simple proof. *Institute Of Systems and Robotics, Tech. Rep*, pages 1–4, 2001.
13. Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.
14. Yuriy Nevmyvaka, Yi Feng, and Michael Kearns. Reinforcement learning for optimized trade execution. In *Proceedings of the 23rd international conference on Machine learning*, pages 673–680. ACM, 2006.
15. Lawrence R Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.
16. Carl Edward Rasmussen and Christopher KI Williams. *Gaussian process for machine learning*. MIT press, 2006.
17. Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of the 31st International Conference on International Conference on Machine Learning-Volume 32*, pages II–1278. JMLR. org, 2014.
18. Ruslan Salakhutdinov and Geoffrey Hinton. Deep boltzmann machines. In David van Dyk and Max Welling, editors, *Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics*, volume 5 of *Proceedings of Machine Learning Research*, pages 448–455, Hilton Clearwater Beach Resort, Clearwater Beach, Florida USA, 2009. PMLR.
19. Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015.
20. David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484, 2016.
21. Richard S Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3(1):9–44, 1988.
22. Richard S Sutton, Andrew G Barto, et al. *Reinforcement learning: An introduction*. MIT press, 1998.

23. Yee Whye Teh and Michael I Jordan. Hierarchical bayesian nonparametric models with applications. *Bayesian Nonparametrics*, 1:158–207, 2010.
24. Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *AAAI*, volume 2, page 5. Phoenix, AZ, 2016.
25. Hao Wang and Dit-Yan Yeung. Towards bayesian deep learning: A framework and some existing methods. *IEEE Transactions on Knowledge and Data Engineering*, 28(12):3395–3408, 2016.
26. Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Hasselt, Marc Lanctot, and Nando Freitas. Dueling network architectures for deep reinforcement learning. In *International Conference on Machine Learning*, pages 1995–2003, 2016.
27. Andrew G Wilson, Zhiting Hu, Ruslan R Salakhutdinov, and Eric P Xing. Stochastic variational deep kernel learning. In *Advances in Neural Information Processing Systems*, pages 2586–2594, 2016.
28. Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P Xing. Deep kernel learning. In *Artificial Intelligence and Statistics*, pages 370–378, 2016.
29. Mingyuan Zhou, Yulai Cong, and Bo Chen. Augmentable gamma belief networks. *Journal of Machine Learning Research*, 17(1):5656–5699, 2016.