Reinforcement Learning project

ENSAE 2021/2022

Professeur: Sofiane Medjkoune-Capgemini

Mastère Spécialisé - Data Science

Hajar Arrazki & Prisca Bahi & Jeffrey Verdière

VIME: Variational Information Maximizing Exploration

Contents

1 Main contribution of each member

We all chose the article we worked on for this project. We splited the work as follows:

- **Hajar Arrazki :** Explanation of the model and implementation of an example code
- **Prisca Bahi :** Introduction and litterature review
- **Jeffrey Verdière :** Explanation of the model and illustration of the model in an example

2 Introduction

Reinforcement learning is a field for solving problems that we can present with Markov decision processes (MDP). It relies on the interaction of a learning agent, which is trying to achieve a goal, with an environment that has certain states. To do so, the agent performs many actions that change the state of the environment and lead to changes in the outcome. The reactions can be rewards or punishments through which the agent learns which actions will allow it to develop a policy. In other words, the goal is to study how an agent can maximize its cumulative reward in a previously unknown environment, which it learns about through experience.

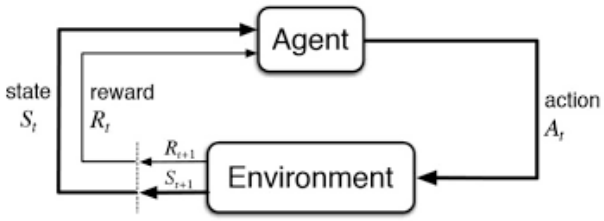


Figure 1: Reinforcement Learning procedure

A basic reinforcement learning agent AI interacts with its environment in discrete time steps.
At each time t, the agent receives the current state s_t and reward r_t. It then chooses an action a_t from the set of available actions, which is subsequently sent to the environment. The environment moves to a new state s_{t+1} and the reward r_{t+1} associated with the transition(s_t, a_t, s_{t+1}) is determined. The goal of a reinforcement learning agent is to learn a policy: $\pi : A \times S \rightarrow [0, 1]$, $\pi(a, s) = \Pr(a_t = a \mid s_t = s)$ which maximizes the expected cumulative reward.

One of the main challenges of reinforcement learning agents is to manage exploration even if there is no reception of valuable information. For example, the agent has some observation of the environment, but has not yet reached the goal. Only from the observation, it cannot tell if the goal is found. However, if the agent never experiments, it cannot learn from it either. Moreover, the agent has to decide when to explore and when to exploit the knowledge.

It could simply decrease the exploration over time but it would be very likely that during an execution, something in the environment would change, which would then need to be re-explored.

- Environment state: $s \in S$
- Agent actions: $a \in A$
- State transactions: $P(S_{t+1}|St, a_t)$
- Reward: $r_t = r(S_t, a_t)$

Optimization of the optimal policy: the objective is to optimize the total reward which is define by [10] :

$$R_t = r_t + \gamma * r_{t+1} +^2 *r_{t+2} + ... + \gamma^n r_{t+n} \tag{1}$$

which could be generalized by

$$R_t = \sum_i \gamma^i r_{t+i} \tag{2}$$

Reinforcement learning mainly consists in finding the optimal policy that maximizes the expected rewards.

$$\pi(s) = P(a|s) \cdot E[R] \tag{3}$$

γ is called [7] the patience of RL process. We consider if we prefer the current action or the future one.

Q-Learning algorithm

We want to find the best rewards. Therefore we maximize [16]:

$$Q(S_t, a_t) = r_t + \gamma \cdot \max_a Q(s_{t+1}, a') \tag{4}$$

Therefore, we need to remember the Q-value for each step of the process

$$\pi(s) = \arg\max_a Q(s, a) \tag{5}$$

Exploration strategies

[12] explains the two main strategies to explore our environment:

ϵ greedy

With a probability ϵ, we take a uniformly random action; otherwise we can take optimal action.

Softmax

We can pick action proportionally to transformed Q-values:

$$P(a) = softmax(\frac{Q(a)}{std}) \tag{6}$$

Some methods have a built-in exploration strategy.
[12] explains that this method needs a lot of try to find before finding the optimal policy.

3 Limits of Q-Learning and the introduction of varational models

[9] explains that scalable and effective exploration remains a key challenge in reinforcement learning (RL). While there are methods with optimality guarantees in the setting of discrete state and action spaces, these methods cannot be applied in high-dimensional deep RL scenarios. As such, most contemporary RL relies on simple heuristics such as -greedy exploration or adding Gaussian noise to the controls. We will explain in the next part the possibility introduced by the varational Information Maximizing Exploration strategy based on maximization of information gain about the agent's beliefs on environment dynamics. We will see that VIME modifies the MDP reward function and can be applied with several different underlying RL algorithms.

4 VIME main idea

VIME uses the information gain about the agent's internal belief or "curiosity" of the dynamics model as a driving force [1]. While exploring, the agent chooses the action that will result in surprising states that create larger updates in the dynamics model distribution. Usually, [9] this maximizes the reduction in entropy which represent unexplored states.

[9] explains the different concept to solve the scalability issue:

Curiosity
Taking actions that increase your knowledge about the world (the environment).

Knowledge about the world
Whatever allows you to predict how world works depending on your behaviors.

Add curiosity to the rewards

$$r(z, a, s') = r(s, a, s') + \beta r_{vime}(z, a, s') \tag{7}$$

where :

z : is the the vector of our last actions
s' : is the next step and s the last step.

We want our rewards to be a sum of the actual reward we get and something called the "curiosity".

Definition of curiosity:

$$r_{vime}(z, a, s') = I(\theta; s'|z, a) \tag{8}$$

It is actually the information between the state that we just observed given history and previous actions.

Environment model

$$P(s'|s, a, \theta) \tag{9}$$

θ can be trained with our favorite machine learning model.

Session
z is the history of all states

$$z_t = (s_0, a_0, s_1, a_1, ..., s_t) \tag{10}$$

Surrogate rewards

$$(z, a, s') = r(s, a, s') + \beta I(\theta, s'|z, a) \tag{11}$$

Curiosity

The aim is to maximize the information gain. This information is computed through the entropy of the states before and after performing an action. We try to maximize the sum of reductions in entropy.

$$\begin{aligned} I(\theta; s'|z, a) &= H(\theta|z, a) - H(\theta|z, a, s') \\ &= \mathbb{E}_{KL}[P(\theta|z, a, s')\|P(\theta|z)] \end{aligned}$$

For example:

$$\begin{aligned} I(x, y) &= \int_x \int_y P(x, y) \log\left(\frac{P(x, y)}{P(x)P(y)}\right) dxdy \\ &= \int_x \int_y P(x|y)P(y) \left[\log(P(x|y).P(y) - \log(P(x)\right] dxdy \end{aligned}$$

We can rewrite our last equation with the following equation.

$$I(x, y) = \int_x P(y) \int_y P(x|y) \log\left(\frac{P(x|y)}{P(x)}\right) dxdy \tag{12}$$

x is the input(the information given) while y is the output of the model.

4.1 Naive objective

To calculate the above expressions, we use the expectation of the Kullback divergence [8]. The Kullback divergence is a way of measuring how a probability distribution differs from another distribution.

$$\mathbb{E}[D_{KL}[P(\theta|z,a,s')\|P(\theta|z)]] = \int_{s'} P(s'|s,a) \int_{\theta} P(\theta|z,a,s') \cdot \log\left(\frac{P(\theta|z,a,s')}{P(\theta|z)}\right) d\theta ds' \quad (13)$$

Where:

- $P(s'|s,a)$ can be sample from an MDP process
- $P(s'|s,a) \int_{\theta} P(\theta|z,a,s') log(\frac{P(\theta|z,a,s')}{P(\theta|z)}$ sample somehow...

With the Bayes formula we can write:

$$P(\theta|z) = \frac{P(z|\theta)P(\theta)}{P(z)} \quad (14)$$

We can simplify our last equation again with the Bayes formula:

$$P(\theta|z) = \frac{\prod_t P(S_{t+1}|S_t,a_t,\theta)P(\theta)}{\int_{\theta} P(z|\theta)P(\theta)d\theta} \quad (15)$$

where:

- $P(s_{t+1}|s_t,a_t,\theta)$ is the model
- $P(\theta)$ is the prior
- $\int_{\theta} P(z|\theta)P(\theta)d\theta$ is the output of our neural networks

The problem is the denominator of our last expression which is maximized over θ. Indeed it is better to avoid computing $P(\theta|z)$ directly.

Illustration of the problem of the naive concept with an exemple with two actions:
Let's illustrate our problem with an example with 2 events:

If we conditionalize

$$p(\theta|s_1,a_2) = \frac{p(s_2|\theta_3 a_1)p(\theta)}{\int p(s_1|\theta_3,a_1)p(\theta)d\theta} \quad (16)$$

Let's also assume that we have:

- $(x_1, ..., x_n) \sim p(x|\theta)$
- $p(\theta|x_1) = \frac{p(x_1|\theta)p(\theta)}{\int p(x_1|\theta p(\theta) d\theta}$

We can see on this simple example that we cannot compute the Kullback leibler divergence easily with this analytical expression.

5 The model required

We want a model that:

- predicts $P(s'|z, a, \theta)$
- allows to estimate $P(\theta|D)$
- from which we can sample from

5.1 The necessity of bayesian neural networks

To calculate the distribution shown during last part, some methods were explored by the bayesian scientists.

Bayesian neural networks were used and described for the first time by [11] where it was needed to have a probabilistic interpretation of neural networks that were considered as "black-box" till then. Instead of considering each weight as deterministic, we settle a normal distribution on each weight as shown on Figure 2. This allows to prioritize few data samples using the weights. A variational Bayesian method is applied to approximate the integrals so as to provide an analytical approximation to the posterior probability of the unobserved variables and to derive a lower bound for the marginal likelihood of the observed data.

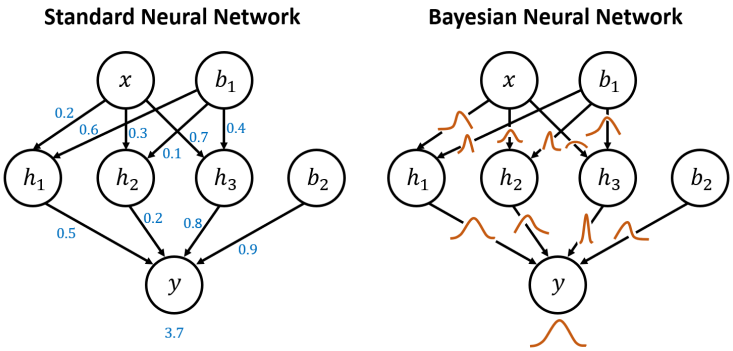


Figure 2: Regular Neural Networks vs Bayesian Neural Networks

In the general case, the idea is the following:

- No explicit weights
- Maintain parametric distribution on the instead!
- Pratical: fully-factorized normal or similar

$$q(\theta|\phi : [\mu, \sigma]) = \prod_i N(\theta_i|\mu_i, \sigma_i) \tag{17}$$

$$P(s'|s, a) = \mathbb{E}[P(s'|s, a, \theta)] \tag{18}$$

How does a bayesian neural networks learn?

- Learn parameters of that distribution (reparametrization trick)
- Less variance: local reparameterization trick

We do not give in our review the explicit formula:

$$\phi = \arg\max_\theta \mathbb{E}[P(s'|s, a, \theta)] \tag{19}$$

[1] explains that we do not want to maximize it four all of our data but only for what we call the "lower bound". We want to minimize the real distribution and the simulated distribution. For that we want to minimize the Kullback-Leibler divergence.

$$\phi_t = arg\max_\theta(-D_{KL}[q(\theta|\phi)\|p(\theta|z_t)] \tag{20}$$

$$\phi_t = arg\max_\theta(\ \mathbb{E}[\ \log(p(z_t|\theta)) - D_{KL}[\ q(\theta|\phi)\|p(\theta)]) \tag{21}$$

Unfortunatly, we cannot calculate $\mathbb{E}log(p(z_t|\theta))$ because of what we showed before with the integrals. Therefore we need to use a reparametrization trick with Bayesian Neural Networks.

$$\mathbb{E}[\ \log(p(z|\theta))] = \mathbb{E}[\log(p(z|(\mu_\phi + \sigma_\phi * \psi)))] \tag{22}$$

This last term enable a better reparametrization trick.

6 VIME objectives thanks to Bayesian Neural Networks

6.1 Main VIME theory

[3] explain that VIME main objectives is to calculate:

$$\mathbb{E}[D_{KL}[P(\theta|z,a,s')||P(\theta|z)] \approx \int_{s'} P(s'|s,a) \int_{\theta} q(\theta|z,a,s') \log\left(\frac{q(\theta|z,a,s')}{q(\theta|z)}\right) d\theta ds' \quad (23)$$

where:

- we sample from the environment $\int_{s'} P(s'|s,a)$
- we sample from Bayesian Neural Networks $\int_{\theta} q(\theta|z,a,s')$
- Prediction from the neural networks (inference): $q(\theta|z,a,s')$ and $q(\theta|z)$

6.2 The algorithm released thanks to VIME algorithm

- Step 1: Interact with environment, get (s,a,r,s')
- Step 2: Compute curiosity reward: $\tilde{r}(z,a,s') = r(s,a,s') + \beta D_{KL}[q(\theta|\phi')\|q(\theta|\phi)]$
- Step 3: train agents (with any RL algorithm) (s,a,$\tilde{r}$,s')
- Step 4: train BNN(s,a,s') (maximize lower bound)

7 VIME in practice and AI in production

[17] explain that in practice it is not easy to seattle VIME but some technical aspects have to be considered:

- **Use batches of many** (s,a,r,s') for CPU/GPU efficiency and greatly improves RL stability
- **Simple formula for KL**

For the simple formula for KL, we need to assume fully-factorized normal distribution:

$$D_{KL}[\ q(\theta|\phi')\ \|\ q(\theta|\phi)\ ] = \frac{1}{2}\sum_{i<|\theta|}[(\frac{\sigma_i'}{\sigma_i}) + 2\log(\sigma_i) - 2\log(\sigma_i') + \frac{(\mu_i' - \mu_i)^2}{\sigma_i^2}] \quad (24)$$

We could have done even simple with a second order Tayler approximation. Also, [17] explains that we need to divide KL by its running average over past iterations.

8 Limits of VIME and the future possibilites

VIME is based on the bayesian neural networks. However, some new sampling has occured since then. Indeed, the Kullback leibler divergence can be estimated through different sampling such as MCMC Monte Carlo Droupout.[6] explains that Monte Carlo Dropout which consists in settling a Bernoulli distrubition on weight can enable to calculate the Kullback-Leibler divergence in neural networks such as shown on next figures:

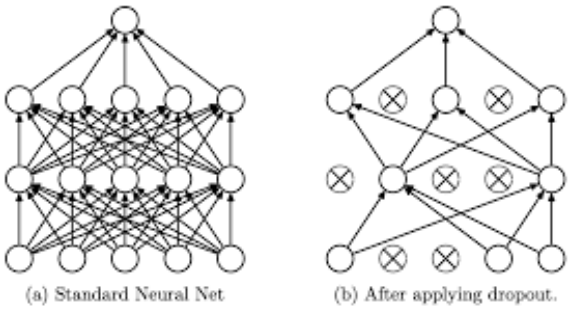


Figure 3: Monte Carlo Dropout in the context of neural networks

We can imagine calculating the Kullback-Leibler divergence with Monte Carlo drop out simulations.

One can also question the idea behind the maximization of gain instead of the expected rewards as the Bayesian RNN induces a distribution over rewards. The idea behind curiosity driven Reinforcement learning is more relevant when there are no rewards. Using the curiosity approach is likely to add unnecessary exploration. For instance, suppose that an action yields a reward of exactly 10, while a second action yields uncertain rewards of at most 9. Curiosity driven RL would explore the second action in order to resolve the uncertainty even though this action will never be optimal. Bayesian RL that optimizes rewards only will explore the environment systematically, but will explore only what is necessary.

The estimated model is not used to find an optimal policy as model free reinforcement learning such as TRPO, REINFORCE and ERWR are used. Since the model is estimated, model-based reinforcement learning could be more appropriate for this setting. Maybe using a model-based setup could be further explored.

In the implementation section of the article, the KL divergence is divided by the median but the idea behind does not seem too obvious as it suggests an underlying instability but no further explanation was provided on that matter. Moreover, no code was provided with the article which makes it complicated to reproduce the results.

Finally, Figure 2 of the article shows that for the cart pole swing-up task there is no significant difference between the reinforcement learning agent with and without VIME exploration. However, no interpretation of this result was provided.

9 Basis for other work

This paper was cited 596 times according to Google Scholar. Therefore, it has become a reference regarding the exploration aspect in reinforcement learning. For instance, it was used in [4] and [5] which enumerate the main algorithms used in the Reinforcement Learning field.

VIME still outperforms models that were investigated after it was published for some setups [15]. However, similar performances have been reached at a lower computational cost [2]. Some works have also been able to reach better performances [14] [13].

10 Implementation and illustration on the mountain car environment

No code was provided with the article. Our implementation was inspired by this github repository: `https://github.com/justheuristic/vime`

Our notebook runs with two python files bnn.py and curiosity.py: bnn.py creates the functions necessary for the creation of the Bayesian Neural Network.

However, we had to alter some functions because it would not run as it was. There were bugs when we wanted to run the gym environment. There were bugs in the definition of the Bayesian Neural Network as well and we had to change some functions from the curiosity.py file.

We also had to change the code for visualizing the results of the experiment.

10.1 Mountain car environment

The article states that using VIME with TRPO algorithm gives better results than only doing TRPO on the mountain car environment. Therefore we decided to try VIME on this algorithm to see if we could reproduce the results from the article.

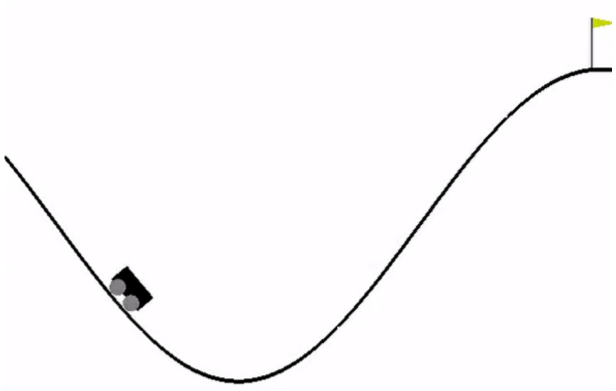


Figure 4: Mountain Car setup

The car starts in between two hills. The goal is for the car to reach the top of the hill on the right. However, the car does not have enough engine power to reach the top of the

hill by just accelerating forward. To win, the car must build momentum by swinging itself until it has enough speed to reach the flag.

10.2 Our Results

We managed to make our algorithm work however the results we got could be improved.

We ran our algorithm only on 100 iterations because it took very long (4 hours).

Ideally, we would like to train it on more iterations but this could not be done with acceptable times.

The long computation time might come from the way the algorithm was written and maybe it could be optimized in order to reduce it. This also might come from our local machines which do not perform very well. The very long running time also suggests a stability issue that needs fixing.

Either way, it would have been more interesting to see how using a larger number of iterations affects the results. Unfortunately we did not have the time to dive into this aspect.

For only 100 iterations, we see on Figure 5 that the rewards do not change on the few number of iterations that we have used our model on. Maybe using more iterations would have allowed to see the rewards change and improve them.

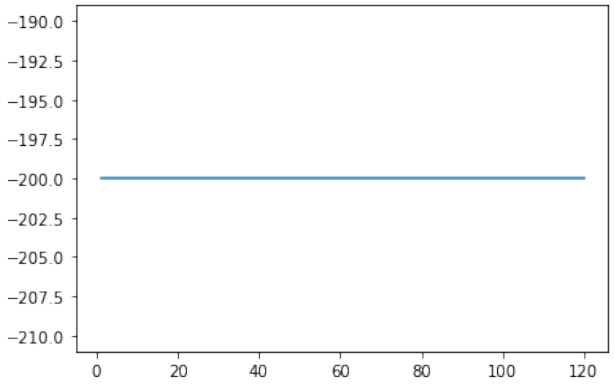


Figure 5: Reward with number for 100 iterations

When looking at the predicted state values Figure 6, we see that the algorithm takes on a wide range of positions that it wants to explore. Using more iterations could have allowed to reach convergence. However, we were able to visualize the idea behind the curiosity approach which leads to a wide range of visitation spaces.

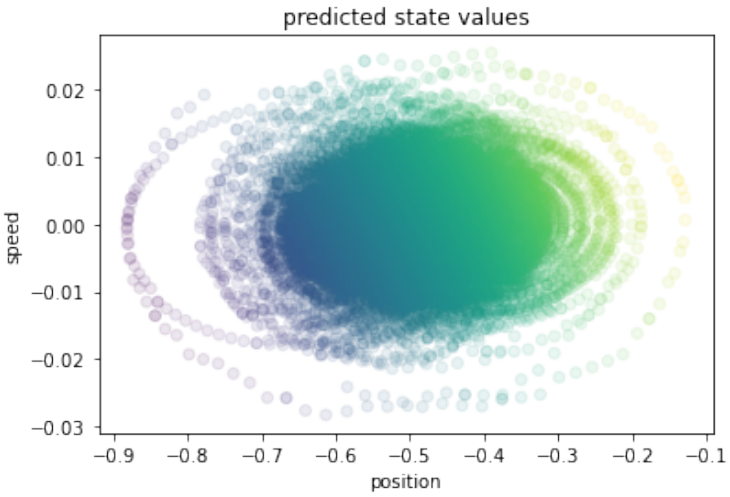
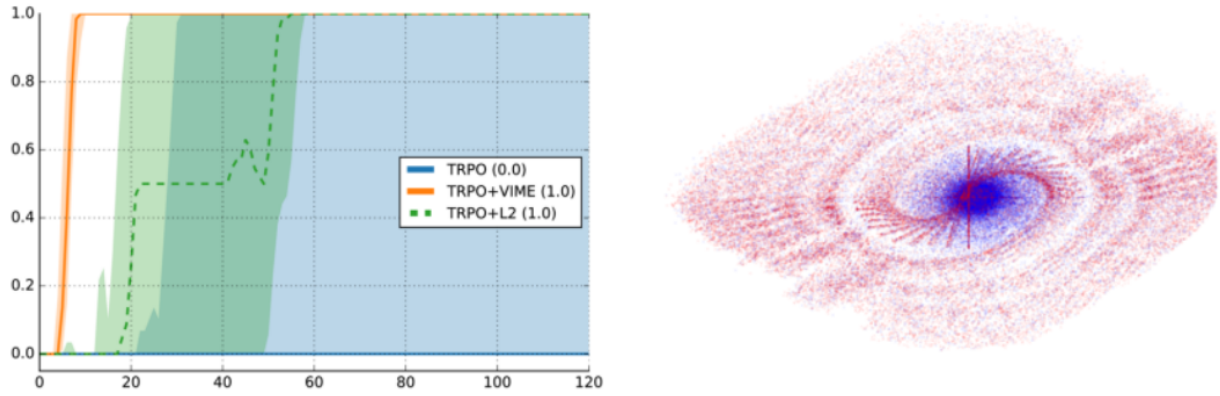


Figure 6: Predicted state values with VIME

10.3 The article's results on mountain car

The article shows Figure 7 that VIME performs much better than naïve approaches, achieving the goal in most cases. It showed that curiosity drives the agent to explore, even in the absence of any initial reward, where naïve exploration completely breaks down. As for the state space, VIME leads to a more diffused visitation pattern, exploring the states more efficiently, reaching the goal more quickly.



(a) reward

(b) state space

Figure 7: Reward and Predicted state values with VIME in the article on mountain car

Therefore, we were able to get a visualization around the curiosity strategy but our rewards did not show improvement fast enough.

11 Conclusion

We have explored Variational Information Maximizing Exploration (VIME), a curiosity-driven exploration strategy for continuous control tasks. Variational inference is used to approximate the posterior distribution of a Bayesian neural network that represents the environment dynamics. Using information gain in this learned dynamics model as intrinsic rewards allows the agent to optimize for both external reward and intrinsic surprise simultaneously. Empirical results show that VIME performs significantly better than heuristic exploration methods across various continuous control tasks and algorithms.

As future work, we would like to investigate measuring surprise in the value function and using the learned dynamics model for planning.

References

[1] Houthooft Chen Duan Schulman Turck Abbeel. "VIME: Variational Information Maximizing Exploration". In: ().

[2] Joshua Achiam and Shankar Sastry. "Surprise-Based Intrinsic Motivation for Deep Reinforcement Learning". In: *CoRR* abs/1703.01732 (2017). arXiv: 1703.01732. URL: http://arxiv.org/abs/1703.01732.

[3] Azizzadenesheli Anandkumar. "Efficient Exploration through Bayesian Deep Q-Networks". In: ().

[4] Kai Arulkumaran et al. "Deep Reinforcement Learning: A Brief Survey". In: *IEEE Signal Processing Magazine* 34.6 (2017), pp. 26–38. DOI: 10.1109/MSP.2017.2743240.

[5] Vincent François-Lavet et al. "An Introduction to Deep Reinforcement Learning". In: *CoRR* abs/1811.12560 (2018). arXiv: 1811.12560. URL: http://arxiv.org/abs/1811.12560.

[6] Gal. "Uncertainty in Deep Learning". In: ().

[7] Dewanto Dunn Eshragh Gallagher. "Average-reward model-free reinforcement learning: a systematic review and literature mapping". In: ().

[8] Filippi ćappe n Garivier. "Optimism in Reinforcement Learning and Kullback-Leibler Divergence". In: ().

[9] Arnold Mankowitz Hester. "Challenges of Real-World Reinforcement Learning". In: ().

[10] Mu Peng Gu Li. "Mixed Reinforcement Learning for Efficient Policy Optimization in Stochastic Environments". In: ().

[11] MacKay. "Bayesian neural networks and density neural networks". In: ().

[12] McFarlane. "A Survey of Exploration Strategies in Reinforcement Learning". In: ().

[13] Deepak Pathak, Dhiraj Gandhi, and Abhinav Gupta. "Self-Supervised Exploration via Disagreement". In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, Sept. 2019, pp. 5062–5071. URL: https://proceedings.mlr.press/v97/pathak19a.html.

[14] Deepak Pathak et al. “Curiosity-driven Exploration by Self-supervised Prediction”. In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, June 2017, pp. 2778–2787. URL: https://proceedings.mlr.press/v70/pathak17a.html.

[15] Haoran Tang et al. “#Exploration: A Study of Count-Based Exploration for Deep Reinforcement Learning”. In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper/2017/file/3a20f62a0af1aa152670bab3c602feed-Paper.pdf.

[16] Violante. “Simple Reinforcement Learning: Q-learning”. In: ().

[17] Igl Zintgraf Anh Wood Whiteson. “Deep Variational Reinforcement Learning for POMDPs”. In: ().