Gaussian Process based Deep Dyna-Q Approach for Dialogue Policy Learning

Guanlin Wu [1,2,*] **Wenqi Fang** [3,*,†] **Ji Wang** [1,*] **Jiang Cao** [2] **Weidong Bao** [1]
Yang Ping [2] **Xiaomin Zhu** [1] **Zheng Wang** [4]
[1]National University of Defense Technology, Changsha, China
[2]Academy of Military Science, Beijing, China
[3]Nanhu Laboratory, Jiaxing, China
[4]Shenzhen Institutes of Advanced Technology, CAS, Shenzhen, China
{wuguanlin16,wangji,wdbao,xmzhu}@nudt.edu.cn
ocean.py@163.com amscaojiang@126.com
wqfang@nanhulab.ac.cn zheng.wang@siat.ac.cn

Abstract

Applying reinforcement learning to dialogue policy learning requires prohibitively large rounds of human-machine interactions. To improve the learning performance, the Deep Dyna-Q framework with a world model that imitates real users is widely used in recent years. Unfortunately, how to build an effective world model and how to evaluate the experiences generated by the world model efficiently have not been well studied. In order to further improve the effectiveness and efficiency of dialogue policy learning, we present a novel Gaussian Process based Deep Dyna-Q approach in this paper. The Gaussian Process model, which is analytically tractable and fits for small-sample problems, is introduced to build the world model. In addition, we design a highly efficient Kullback-Leibler divergence based discriminator to evaluate the quality of experiences generated by the world model. Extensive experiments validate the effectiveness and robustness of our proposed approach. The task-completion success rate can be improved by about 20% with fewer human-machine interactions.

1 Introduction

Task-completion dialogue policy learning aims to build a task-completion dialogue system that can help people complete a specific single task or multi-domain tasks through several rounds of natural language interactions. It has been widely used in chat robots and personal voice assistants, such as Siri of Apple and Cortana of Microsoft.

Reinforcement learning (RL) becomes the mainstream dialogue policy learning method in recent years (Chen et al., 2020; Saha et al., 2020; Li et al., 2020). Based on the RL, the task-completion dialogue system can gradually adjust policy through interacting with real users to improve performance. However, the vanilla RL methods require many rounds of human-machine dialogue interactions before getting a satisfactory dialogue policy, which not only increases the training cost but also deteriorates user experience during the early training phase.

In order to address the above problem and accelerate the learning process of dialogue policy, Deep Dyna-Q (DDQ) (Peng et al., 2018) is proposed based on the Dyna-Q framework where a environment model, known as *world model*, is introduced to generate simulated user experiences in the dynamic environment. The world model is trained by the real user experience to make itself act more like real users. During the dialogue policy learning, the dialogue agent is trained by both real experiences collected from interacting with real users and simulated experiences collected from interacting with the world model.

By introducing the world model, DDQ can promote the learning efficiency effectively during dialogue policy learning. However, it still faces two critical challenges which are crucial to further improve the dialogue policy learning with limited dialogue interactions.

Firstly, the world model in DDQ is built as a deep neural network (DNN) whose performance heavily relies on the amount of training data. In the initial training stage when the real experiences are relatively few, the data-hungry problem caused by DNN may make the world model fail to generate simulated user experiences with enough quality. It requires a lot of real experiences to train a qualified DNN-based world model that can produce high-quality simulated experiences. The world model implemented by the data-hungry model such as DNN erodes the advantage brought by Dyna-Q framework and makes DDQ less effective in reality.

Secondly, it has been pointed in (Peng et al.,

*The three authors contribute equally to this work
†Corresponding author: wqfang@nanhulab.ac.cn

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 1786–1795

2018) that the simulated experience generated by the world model does not necessarily improve performance. Low-quality experiences even hinder the performance seriously. To address this issue, some recent works attempted to control the quality of simulated experiences by using generative adversarial network (GAN) to discriminate the low-quality experiences (Su et al., 2018). Nonetheless, the notorious instability of training GANs may make dialogue policy learning suffer badly from non-convergence and high sensitive to the hyperparameter selections, which is demonstrated in Section 3 of our paper. It is an important yet unsolved problem to efficiently discriminate low-quality experiences during dialogue policy learning.

In order to tackle the above two challenges, we propose a new Gaussian Process based Deep Dyna-Q approach. Compared with the previous works (Peng et al., 2018; Su et al., 2018), the world model in our approach is built as a Gaussian Process (GP) model rather than a DNN model. The GP model is analytically tractable and enjoys the advantage of dealing with small-sample problems (Patacchiola et al., 2019; Gašić et al., 2017; Su et al., 2016), which makes it more competitive than DNN models in this work. In addition, we design a novel method to evaluate the quality of simulated user experiences by comparing them with real user experiences based on Kullback-Leibler (KL) divergence directly without any extra training of discriminator. The main contributions of this work are as follows:

- We present a new GP-based Deep Dyna-Q approach, which can generate high-quality simulated experiences to supplement the limited real user experience. To build the world model as a GP model, we design a Dyna-Q framework that supports regression mode meeting the basic requirements of using GP methods.
- We propose a KL divergence based discriminator which is able to fluently control the quality of simulated experiences. By introducing KL divergence, we can check the distribution of experiences without wasting extra work to design and train a complex discriminator. It is easier to evaluate the quality of simulated experiences in reality, and greatly improve the computational efficiency while ensuring the robustness and effectiveness of the dialogue policy.

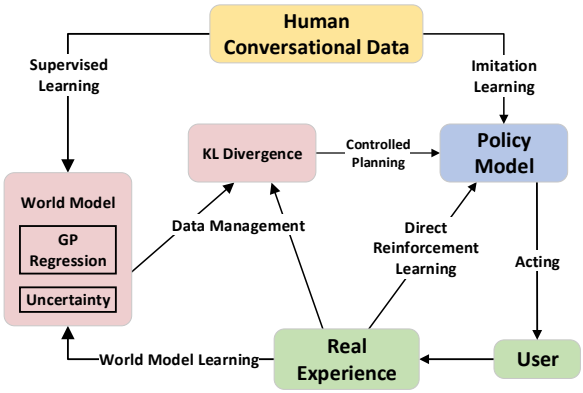


Figure 1: Architecture of the proposed dialogue learning approach.

2 Gaussian Process based Deep Dyna-Q Approach

In this section, we introduce the proposed GP based DDQ approach in detail. Figure 1 shows the architecture of the proposed approach. Our GP based DDQ approach follows the learning process of DDQ, and concentrates on two issues: 1) how to build an effective world model, and 2) how to evaluate simulated experiences efficiently. Accordingly, we build the world model as a GP model and design a novel KL divergence based discriminator to promote the efficiency of dialogue policy learning.

The dialogue policy learning starts with initializing the policy model and the world model by using the human conversational data. In *direct reinforcement learning*, the policy model is trained by interacting with real users to improve the dialogue policy. Meanwhile, the real experiences collected from real users are used to train the world model, which is referred to as *world model learning*. In *data management*, the simulated experiences generated by the world model are evaluated by comparing with the real experiences based on the KL divergence. Then, the qualified ones are pushed into the replay buffer for *controlled planning* to train the policy model without interaction with real users.

2.1 Gaussian Process based World Model

During the planning process, we implement the world model to generate simulated experience that can be used to improve dialogue policy. The world model, denoted by $W(s, a; \theta_w)$, consists of three GP models shown in Figure 2, parameterized by different θ_w. Three GP regression models are used to generate response action a^u, reward r, and variable

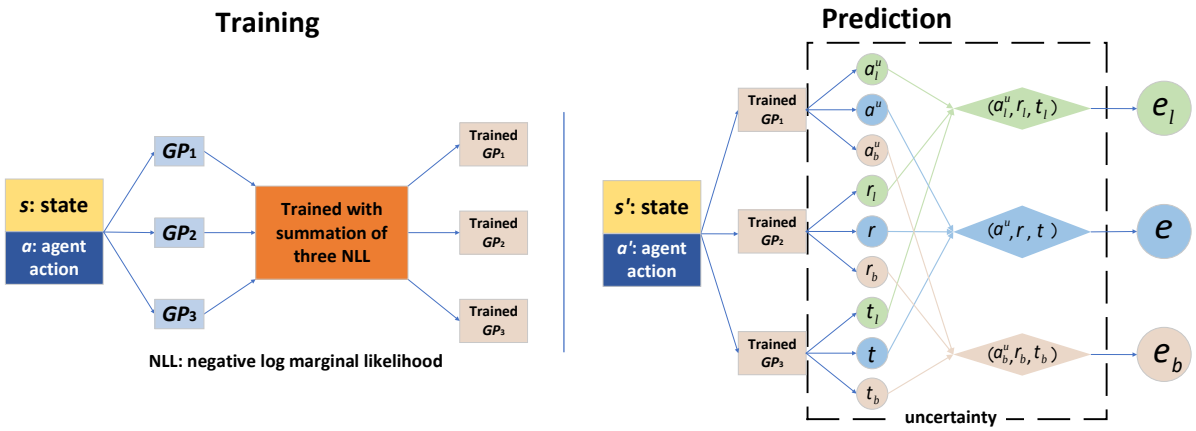


Figure 2: The training and prediction stage of world model.

t indicating whether the dialogue terminates, respectively. We denote the simulated experience as a tuple $\boldsymbol{e} = (a^u, r, t)$. In a practical GP regression problem, the observed targets y are generated from the function $f(x)$ by adding independent Gaussian noise (Williams and Rasmussen, 2006):

$$y = f(x) + \epsilon, \tag{1}$$

where $p(f|x) = N(f|\mu, K(x,x))$ with mean μ and kernel function K, and $\epsilon \sim N(0, \sigma^2 I)$, I is the identity matrix. According to the Bayesian principle, the conditional mean and covariance of posterior distribution, $p(y^*|y, x, x^*)$, with test input x^* is as follows:

$$\mu + K(x^*, x)\Sigma^{-1}(y - \mu) \tag{2}$$

$$K(x^*, x^*) + \sigma^2 I - K(x^*, x)\Sigma^{-1}K(x, x^*), \tag{3}$$

where $\Sigma = K(x,x) + \sigma^2 I$. To accommodate the correlation properties of human dialogue, the stationary kernel function Matérn is used in our case:

$$K_{Mat}(r) = \sigma_f^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu} r}{l} \right)^{\nu} K_\nu \left(\frac{\sqrt{2\nu} r}{l} \right), \tag{4}$$

where σ_f and l are magnitude and lengthscale parameters, respectively. Γ is the gamma function, K_ν is the modified Bessel function of the second kind, and ν are positive parameters of the covariance. The argument r represents distance between observations (Hensman et al., 2017). For the multi-dimensional input case, its automatic relevance determination (ARD) version could be introduced to deal with this situation (Duvenaud, 2014).

In each round of the world model learning, the current dialogue state s and the last agent action a are concatenated as input for the world model. We set all the GP priors with constant mean and the Matérn kernel ($\nu = \frac{7}{2}$) function. The world model $W(s, a; \theta_w)$ is trained to mimic the real dialogue environments. The training data for the world model learning are collected from the real user and are stored in the replay buffer M^w. The loss function is set as the summation of the negative log marginal likelihood (NLL) of three GP models. Because of the conjugate property, each NLL could be analytically solvable, and their general formulas can be written as:

$$-\log p(y|x) = \frac{n}{2}\log(2\pi) + \frac{1}{2}\log|\Sigma| + \frac{1}{2}(y-\mu)^{\mathrm{T}}\Sigma^{-1}(y-\mu), \tag{5}$$

where $|\cdot|$ represents determinant of the matrix and n is the number of the training data. The world model $W(s, a; \theta_w)$ is refined at the end of each epoch via L-BFGS-B algorithm (Zhu et al., 1997) using real experiences.

During prediction, we use these trained GP models to generate simulated experiences. To increase diversity of the simulated experiences, the uncertainty of GP models is taken into account in the prediction stage, shown by the black box frame labeled as "uncertainty" in Figure 2. We calculate the 50% confidence interval[1] of these three variables. The lower bound and the upper bound of the simulated experience are represented by $\boldsymbol{e}_l = (a_l^u, r_l, t_l)$ and $\boldsymbol{e}_b = (a_b^u, r_b, t_b)$, respectively. Then, we have three simulated experiences $\boldsymbol{e}_l$, $\boldsymbol{e}$, and $\boldsymbol{e}_b$ per prediction. The quality of the three simulated experiences will

[1]The usual 95% confidence interval isn't used here to narrow the variable domain.

be measured by KL divergence, which will be detailed in the following subsection. The qualified simulated experiences will be stored in the replay buffer M^p for training the dialogue policy model.

Differing to DDQ where the world model is essentially a classification model to generate user action a^u, the above GP-based world model is a regression model to make it tractable and much easier to handle than classification model. Considering the user action should be an integer and have finite action domain, the user action generated by the proposed world model should be filtered to meet these inherent requirements. The filtering mechanism consists of the following two steps. Firstly, when the user action is not an integer, which is common in regression case, a^u is round to its nearest integer, a^u_l is replace by its ceiling value, and the floor value of a^u_b is chosen, respectively. Secondly, if the user action is beyond the defined action domain, the upper or the lower bound of the domain will be selected. Through the above process, the user action generated by a regression model can achieve the approximately equivalent effect as the task-specific representation in classification models.

2.2 Management of Replay Buffer

As mentioned in the Introduction, low-quality experiences generated by the world model can hinder the learning performance seriously. In this subsection, we evaluate the quality of the simulated experience to determine whether it can be pushed into the replay buffer for training the dialogue policy model. The whole structure is shown in Figure 3.

We give two dictionaries, i.e., *world-dict* and *real-dict*, to record the frequency of all actions generated by the world model and the real user from the beginning of the dialogue policy learning. The key of the dictionary is user action, and the corresponding value is the frequency of this action. A high-quality simulated experience means that its action is similar to the real user. Therefore, we evaluate the quality of simulated experience by measuring the similarity between *world-dict* and *real-dict* based on the KL divergence which is a non-symmetric variable (Raiber and Kurland, 2017).

The evaluation process is shown in Algorithm 1. This algorithm runs repeatedly during the planning (see Line 19 in Algorithm 2). The variable KL_{pre}, which is initialized as a extremely large number, tracks the KL divergence between *world-dict* and

Algorithm 1: Evaluate Simulated Experiences

Input: User actions in the experience generated by the world model a^u_w; Previous action dictionary of the world model *world-dict*; Previous action dictionary of the real user *real-dict*; KL divergence KL_{pre}.

1 Update *world-dict* with the current user actions a^u_w ;
2 **foreach** *a in world-dict.key* **do**
3 **if** *a in real-dict.key* **then**
4 *same-dict*[a] $\leftarrow$ [*world-dict*[a],*real-dict*[a]]
5 $qualified \leftarrow$ **FALSE** ;
6 **if** $\texttt{Length}$*(same-dict)* $\geq$ *cut-off* **then**
7 Calculate current KL divergence KL using *same-dict*;
8 **if** $KL \leq KL_{pre}$ **then** $qualified \leftarrow$ **TRUE**;
9 $KL_{pre} \leftarrow KL$;
10 **else**
11 $qualified \leftarrow$ **TRUE** ;
12 **if** $qualified$ **then**
13 Push current simulated experience into M^p;

real-dict. When evaluating a simulated experience, we first use its user action to update *world-dict*. Then, we use *same-dict* to save the intersection keys of *world-dict* and *real-dict*, and store their frequencies respectively (see Line 2-4). During the initial stage of planning, there is limited actions in *world-dict*, and hence the length of *same-dict* is quite small. To warm up the world model and expand the replay buffer, we regard the simulated experience as a qualified one directly when the length of *same-dict* is smaller than a constant value *cut-off*. Otherwise, we calculate the current KL divergence KL by using *same-dict*. If the current KL divergence is smaller than that of the previous round KL_{pre}, we regard the current experience as a qualified one (see Line 7-8) because it make the world model more similar to the real user. The qualified experience will then be pushed into M^p for training the dialogue policy model.

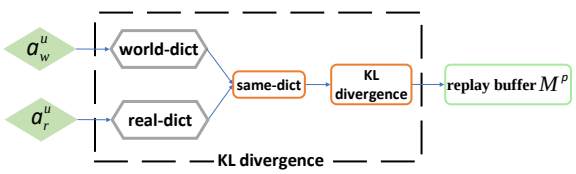


Figure 3: KL divergence calculation.

2.3 Direct and Indirect Reinforcement Learning

For the direct reinforcement learning, the Deep Q-Network (DQN) (Mnih et al., 2015) is adopted to

Algorithm 2: GP based DDQ Approach

Input: Learning epoch *count*; probability ϵ for ϵ-policy; planning step K; period T to update $Q'(s,a;\theta_{Q'})$; user goal $G=(C,R)$, where C is a set of constraints and R is a set of requests; training epoch Z_1 and Z_2.

Initialize $Q(s,a;\theta_Q)$ and $W(s,a;\theta_w)$ via pre-training on human conversational data ;
Initialize $Q'(s,a;\theta_{Q'})$ with $\theta_{Q'}=\theta_Q$;
Initialize the replay buffer M^w for world model and M^p for policy model using Reply Buffer Spiking (Lipton et al., 2016) ;
for $i \leftarrow 1$ **to** *count* **do**
  #Direct Reinforcement Learning:
  User starts a dialogue with action a^u ;
  Generate an initial dialogue state s ;
  while t *is not terminal state* **do**
    Policy model selects and executes action a based on ϵ-policy ;
    User responses a^u_r, reward r, and terminal state t;
    Update *real-dict* ;
    Store (s,a,r,a^u_r,t) to M^p and M^w ;
    $s \leftarrow s'$;
  #Controlled Planning:
  for $k \leftarrow 1$ **to** K **do**
    Sample user action a^u from G ;
    while t' *is not terminal state* **do**
      Policy model selects and executes action a based on ϵ-policy ;
      World model responds a^u_w, r and t' ;
      Store (s,a,a^u_w,r,t') to M^p based on $qualified$ from Algorithm 1 ;
      $s \leftarrow s'$;
  Sample random mini-batch samples from M^p ;
  Update θ_Q via Z_1-step Q-learning ;
  $\theta_{Q'} \leftarrow \theta_Q$ every T steps;
  #World Model Learning:
  Sample random mini-batch samples from M^w ;
  Update θ_w via Z_2-step L-BFGS-B algorithm ;

improve the dialogue policy based on real experiences. The dialogue agent interacts with the user and uses a DNN to approximate the non-linear Q function. In each step, the agent chooses the corresponding action a to execute using an ϵ-greedy policy (Watkins and Dayan, 1992) according to the observed dialogue state s. In ϵ-greedy policy, a threshold ϵ is set for logical selection, i.e., a random action or a action chosen by the greedy policy $a = \text{argmax}_{a'} Q(s,a';\theta_Q)$ where $Q(\cdot)$ is the value function. Then, the agent receives the reward r. The real user responses a^u_r based on the current environment. The next state s' is updated in the state tracker module. Before we store the experience (s,a,r,a^u_r,t) in the replay buffer M^p, the statistical distribution of a^u_r, denoted as *real-dict*, is updated for further KL divergence inspection.

The value function $Q(s,a;\theta_Q)$, approximated by a DNN, is updated by optimizing θ_Q to minimize the mean-squared loss function as below:

$$\begin{aligned} \mathcal{L}(\theta_Q) &= \mathbb{E}_{(s,a,r,s')\sim\mathcal{M}^p}[(y_i - Q(s,a;\theta_Q))^2] \\ y_i &= r + \gamma \max_{a'} Q'(s',a';\theta_{Q'}), \end{aligned} \tag{6}$$

where $\gamma \in [0,1]$ is a discount factor, and $Q'(\cdot)$ is a separate network that is only updated periodically for generating the targets value y_i. In each iteration, we improve $Q(\cdot)$ by using mini-batch deep Q-learning. We can use several optimization algorithms such as Adam (Kingma and Ba, 2014), Stochastic gradient descent (Sutskever et al., 2013) and RMSprop (Ruder, 2016) to train the deep Q network.

During the indirect reinforcement learning, also known as planning, the dialogue agent improves its dialogue policy by interacting with the world model rather than the real user to reduce the training cost. The frequency of planning is controlled by the parameter K, which means that the planning is performed K steps per step of the direct reinforcement learning. The value of K tends to be large when the world model is able to capture the feature of the real environment accurately. In each step of planning, the world model responses a^u_w based on the current environment. As mentioned in the last subsection, the experience (s,a,r,a^u_w,t') generated during planning will be evaluated by the KL divergence inspection before pushing it into the replay buffer M^p to ensure the quality of experiences.

Algorithm 2 gives the whole process of our proposed approach. Each epoch of dialogue policy learning consists of direct reinforcement learning, controlled planning, and world model learning.

3 Experiment

To illustrate the effectiveness and superiority of our method, we test it in the movie ticket booking task, and compare it with the other methods from two aspects : 1) the change of performance in different hyperparameters; 2) the performance comparison. The source codes and the implementation details are packed in the supplementary materials for reproduction.

3.1 Dataset

We use the same raw data as original DDQ method. It is collected via Amazon Mechanical Turk. The dataset has been manually labeled based on a

schema defined by domain experts, which consists of 11 dialogue acts and 16 slots (Peng et al., 2018). In total, the dataset contains 280 annotated dialogues, the average length of which is approximately 11 turns.

3.2 Dialogue Agents for Comparison

We develop different versions of task-completion dialogue agents to benchmark the performance of our proposed method and its variants.

- The **GPDDQ**(M, K, N) agents are learned by our GPDDQ method, where M is the buffer size, K is the number of planning steps and N is the batch size. The initial world model is pre-trained on human conversational data. Note that we **do not** utilize uncertainty attribute and KL divergence inspection in this agent.
- The **UN-GPDDQ**(M, K, N) agents are very similar to GPDDQ(M, K, N) agents, but the uncertainty is considered in this case. Currently, $\boldsymbol{e}_l$, $\boldsymbol{e}$ and $\boldsymbol{e}_b$ are returned in the world model prediction stage.
- The **KL-GPDDQ**(M, K, N) agents are the same to the UN-GPDDQ(M, K, N) agents, except that the KL divergence inspection is also considered.
- The **GPDDQ**(M, K, N, **rand-init** θ_w) agents are learned by the GPDDQ method with a randomly initialized world model. The reward r and terminal variable t are randomly sampled from their corresponding GP models. And for action a^u, we uniformly sample it from its defined action domain.
- The **GPDDQ**(M, K, N, **fixed** θ_w) agents are only refined during warm-up stage on human conversational data. After that, the world model will not be trained any more.
- The **GPDQN**(M, K, N) agents are learned by direct reinforcement learning. Its performance can be viewed as the upper bound of its GPDDQ(M, K, N) counterpart, assuming that the world model perfectly matches real users.
- For other agents which are not mentioned above, please refer to (Peng et al., 2018; Su et al., 2018).

3.3 Parameter Analysis

To illustrate the advantages of our model in terms of sensitivity to hyperparameter changing, we conduct a series of experiments by changing the corresponding parameters such as batch size, planning step, parameter update policy, and buffer size.

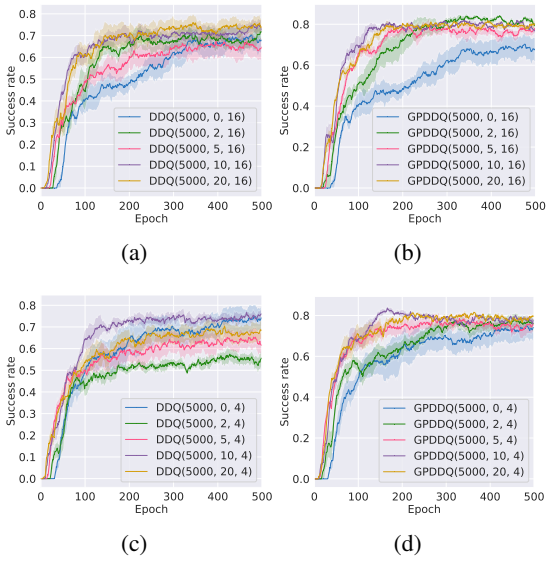
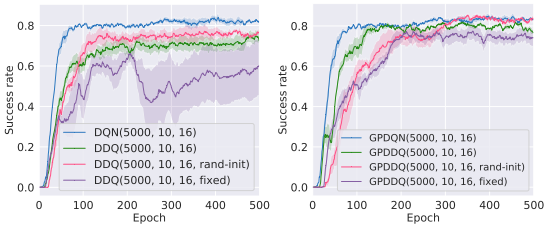


Figure 4: Learning curve for DDQ and GPDDQ with different parameter settings: $M = 5000$, $N = 4, 16$ and $K = 0, 2, 5, 10, 20$.

3.3.1 Batch Size and Planning Step

In this group of experiments, we use the 16 and 4 as the batch size to train the policy network $Q(\cdot)$ and world model $W(\cdot)$ with different planning steps K. The main results are shown in Figure 4 which indicates that the GPDDQ agents consistently outperform the DDQ agents in a statistical sense. In Figure 4 (a) and (b), it can be found that the convergence value of the success rate of GPDDQ agent is much better than that of DDQ agent with the same planning step K. The converged success rate oscillates around 0.8 in our proposed method, however, the corresponding value is about 0.74 in the DDQ method. As the planning steps increase, the learning speeds generally become faster. This phenomenon is consistent with intuition that a large planning step brings faster learning speed. Nonetheless, we can notice that there is no significant difference between the learning curve of $K = 20$ and $K = 10$. This is due to the reduction of the quality of simulated experiences caused by a too large K. To find the optimal value of K, the trade-off between the amount of simulated experience and the quality of simulated experience should be considered seriously.

Since GP method is more robust to hyperparameters (Kuss, 2006), we speculate that it still has better performance with a small batch size. In Figure 4 (c) and (d), we shrink our batch size to 4, and keep the other parameters the same as previ-



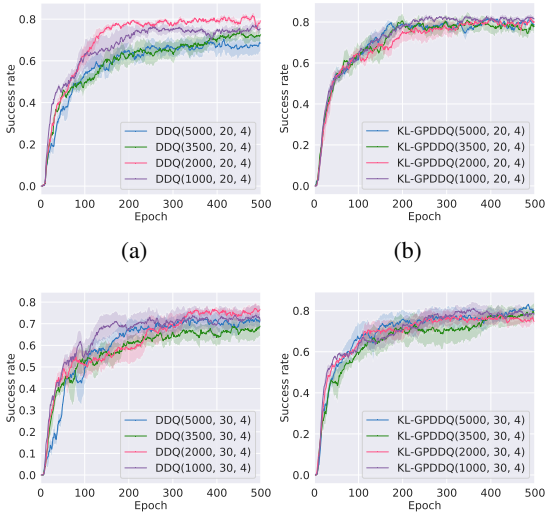
(a) (b)

Figure 5: Learning curve for DDQ and GPDDQ with $M = 5000$, $K = 10$ and $N = 16$, but with different parameter update strategies.

ous experiments. For GPDDQs with $K > 0$, their performances still outperform the DDQ(5000, 0, 4) agent. Moreover, compared to the results when batch size is 16, there is no obvious performance degradation. Besides, since the matrix inversion operation costs more time during training when the batch size is large, the training time consumption can be greatly reduced if the batch size becomes smaller. On the contrary, for DDQ methods, only when $K = 10$, the learning curve is better than DDQ(5000, 0, 4) method in terms of the stable success rate. When increasing the planning step to $K = 20$, its performance degrades dramatically. This may be caused by the insufficient training of DNN when the batch size is too small.

3.3.2 Parameter Update Policy

In this group of experiments, we set $M = 5000$, $K = 10$, $N = 16$, and change its parameter update policy. The results are given in Figure 5. The results indicate that the quality of the world model has a significant impact on the performance of these agents. The DQN and GPDQN method is the completely model free method with K times training data larger than other methods in Figure 5. Due to randomness, the two rising curves are slightly different, but basically the same. Obviously, if the world model is fixed after the warm-up stage, it will produce the worst results. The huge drop in the learning curve of DDQ(5000, 10, 16, fixed) at about 250 epoch may be the result of insufficient training data. For each learning curve of GPDDQ, the proposed GPDDQ method can achieve almost the same maximum value as DQN. In addition, the final sucees rates of GPDDQ are always larger than those of DDQ methods. Even if we use different parameter update policies, the final success rates do not fluctuate too much.



(a) (b)

(c) (d)

Figure 6: Learning curve for DDQ and GPDDQ with different parameter settings: $M = 5000, 3500, 2000, 1000$, $K = 20, 30$ and $N = 4$.

3.3.3 Buffer Size

In this subsection, we evaluate our KL-GPDDQ method, ignoring the other simplified methods, by changing the buffer size. From the perspective of overall performance shown in Figure 6, our proposed method is more stable in different conditions, i.e., different buffer sizes and planning steps. After reducing the buffer size from 5000 to 1000, the learning curve does not change much in our methods. However, for DDQ methods, their performances are still poor. These phenomena make us suspect that the world model in DDQ, built by DNN, may generate many low-quality experiences during planning. Nevertheless, when the buffer size becomes smaller, high-quality experiences become the dominant part of the replay buffer. In terms of convergence, the success rate of KL-GPDDQ method stabilizes around 0.8 after 200 epochs when planning step is 20, and slightly smaller when $K = 30$. On the contrary, the DDQ methods do not converge after 200 epochs. Their success rates are basically lower than those of our proposed methods when converging. This result argues that our method can achieve better and more robust performance with relatively small buffer sizes.

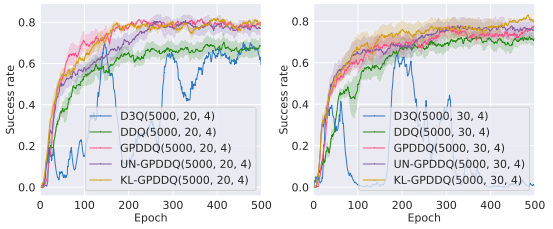
3.4 Performance Comparison

To demonstrate the performance of our proposed method, we compare it with other baseline algorithms. We can find from Table 1 that the DDQ

Agent	Epoch = 100			Epoch = 200			Epoch = 300		
	Success	Reward	Turns	Success	Reward	Turns	Success	Reward	Turns
D3Q(10)*	.6333	28.99	**16.01**	.7000	37.24	**15.52**	.6667	33.09	**15.83**
DDQ(5000, 20, 4)	**.5379**	**12.60**	25.90	**.6466**	**26.79**	**23.60**	**.6612**	**29.14**	**22.41**
GPDDQ(5000, 20, 4)	**.7069**	**35.09**	21.48	.7706	43.65	19.60	.7874	45.72	19.54
UN-GPDDQ(5000, 20, 4)	.5800	17.61	**25.98**	.7050	34.32	22.57	.7726	43.84	19.75
KL-GPDDQ(5000, 20, 4)	.6138	22.57	24.17	**.7915**	**46.39**	19.16	**.7985**	**47.34**	18.97

*The result for D3Q method is borrowed from its original paper (Su et al., 2018).

Table 1: Results of different agents with buffer size 5000 at training epoch = {100, 200, 300}. The best and the worst agents for planning step $K = 20$ are highlighted in blue and red, respectively. (Success: success rate, Turns: dialogue turns)



(a) (b)

Figure 7: Learning curve for DDQ, GPDDQ, UN-GPDDQ and KL-GPDDQ agent with $M = 5000$, $K = 20, 30$ and $N = 4$.

methods are still the worst among the five methods. Due to its extremely large training time consumption and high sensitivity, for D3Q method, we only calculate it once in Figure 7 and borrow its performance from its original paper (Su et al., 2018) in Table 1. From the results of GPDDQ, UN-GPDDQ, and KL-GPDDQ agents, it is obvious that the KL divergence inspection we design is helpful for performance improvement, which can be concluded based on the clear increase of success rate and reward shown in Table 1. Compared with DDQ, our proposed method can improve the success rate by about 20% with fewer user interactions.

Figure 7 shows that the learning speeds of our proposed methods are much faster than those of DDQ and D3Q. It should be noted that the learning curve of D3Q vibrates violently. Especially, when $K = 30$, D3Q even cannot converge to the optimal value. Although D3Q can discriminate low-quality experiences, it is very hard to implement D3Q in reality due to the instability of GANs.

4 Related Work

Most of the works on task-completion dialogue policy learning focus on how to use fewer conversation rounds to complete a specific task (Lu et al., 2019). There are four typical methods, including rule based method (Weizenbaum, 1966), retrieval based method (Mikolov et al., 2013; Pennington et al., 2014; Serban et al., 2017), supervised learning based method (Sukhbaatar et al., 2015; Weston et al., 2016), and reinforcement learning based method (Levin et al., 2002). Since the reinforcement learning based method can fine-tune the current dialogue strategy based on users' feedback to promote user satisfaction, it has been the mainstream of dialogue policy learning method in recent years (Chen et al., 2020; Saha et al., 2020; Li et al., 2020).

However, the vanilla RL methods require prohibitively many rounds of human-machine dialogue interactions before getting a usable dialogue policy. Deep Dyna-Q (DDQ) (Peng et al., 2018) is proposed based on the Dyna-Q framework which introduces an environment model, known as *world model*, to generate simulated user experiences in the dynamic environment to decrease the heman-machine interactions. Based on (Peng et al., 2018), (Su et al., 2018) attempted to control the quality of simulated experiences by using GANs to discriminate the low-quality experiences. (Zhao et al., 2020) proposed a method called DR-D3Q to learn policies in noise robustly by combining dynamic reward and Dueling DQN. Based on human demonstrations, (Wang et al., 2020) presented how to efficiently learn dialogue policy through policy shaping and reward shaping, in which the world model is replaced by an imitation model.

5 Conclusion

In this paper, we propose a Gaussian Process based Deep Dyna-Q approach. The world model is built as a GP model, and a novel KL divergence based discriminator is designed to evaluate simulated experiences. Extensive experiments demonstrate the superiority of our proposed method thanks to the newly-designed world model and discriminator. Compared with existing DDQ framework based methods, both efficiency and robustness are pro-

moted by our proposed method. With this satisfactory result, it is potential to develop more valuable algorithms based on our method. In the future work, we will try to incorporate other strategy, such as tree-based search algorithms (Schrittwieser et al., 2020), to further improve the learning performance.

Acknowledgments

This work was supported by the National Natural Science Foundation of China under grant 62002369, the Scientific Research Project of National University of Defense Technology under grant ZK19-03, Key-Area Research and Development Program of Guangdong Province (Grant No. 2019B010155003).

References

Zhi Chen, Lu Chen, Xiang Zhou, and Kai Yu. 2020. Deep reinforcement learning for on-line dialogue state tracking. *arXiv preprint arXiv:2009.10321*.

David Duvenaud. 2014. *Automatic model construction with Gaussian processes*. Ph.D. thesis, University of Cambridge.

Milica Gašić, Nikola Mrkšić, Lina M Rojas-Barahona, Pei-Hao Su, Stefan Ultes, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2017. Dialogue manager domain adaptation using gaussian process reinforcement learning. *Computer Speech & Language*, 45:552–569.

James Hensman, Nicolas Durrande, and Arno Solin. 2017. Variational fourier features for gaussian processes. *The Journal of Machine Learning Research*, 18(1):5537–5588.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Malte Kuss. 2006. *Gaussian process models for robust regression, classification, and reinforcement learning*. Ph.D. thesis, echnische Universität Darmstadt Darmstadt, Germany.

E. Levin, R. Pieraccini, and W. Eckert. 2002. Learning dialogue strategies within the markov decision process framework. In *IEEE Workshop on Automatic Speech Recognition Understanding*.

Ziming Li, Julia Kiseleva, and Maarten de Rijke. 2020. Rethinking supervised learning and reinforcement learning in task-oriented dialogue systems. *arXiv preprint arXiv:2009.09781*.

Zachary C Lipton, Jianfeng Gao, Lihong Li, Xiujun Li, Faisal Ahmed, and Li Deng. 2016. Efficient exploration for dialogue policy learning with bbq networks & replay buffer spiking. *arXiv preprint arXiv:1608.05081*, 3.

Keting Lu, Shiqi Zhang, and Xiaoping Chen. 2019. Goal-oriented dialogue policy learning from failures. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 2596–2603.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26.

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. 2015. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533.

Massimiliano Patacchiola, Jack Turner, Elliot J Crowley, Michael O'Boyle, and Amos Storkey. 2019. Deep kernel transfer in gaussian processes for few-shot learning. *arXiv preprint arXiv:1910.05199*.

Baolin Peng, Xiujun Li, Jianfeng Gao, Jingjing Liu, and Kam-Fai Wong. 2018. Deep dyna-q: Integrating planning for task-completion dialogue policy learning. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2182–2192.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Conference on Empirical Methods in Natural Language Processing*.

Fiana Raiber and Oren Kurland. 2017. Kullback-leibler divergence revisited. In *Proceedings of the ACM SIGIR International Conference on Theory of Information Retrieval*, pages 117–124.

Sebastian Ruder. 2016. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*.

Tulika Saha, Sriparna Saha, and Pushpak Bhattacharyya. 2020. Towards sentiment aided dialogue policy learning for multi-intent conversations using hierarchical reinforcement learning. *PloS one*, 15(7):e0235367.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. 2020. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609.

Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron Courville, and Yoshua Bengio. 2017. A hierarchical latent variable encoder-decoder model for generating dialogues. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, page 3295–3301. AAAI Press.

Pei-Hao Su, Milica Gasic, Nikola Mrkšić, Lina M Rojas Barahona, Stefan Ultes, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2016. On-line active reward learning for policy optimisation in spoken dialogue systems. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2431–2441.

Shang-Yu Su, Xiujun Li, Jianfeng Gao, Jingjing Liu, and Yun-Nung Chen. 2018. Discriminative deep dyna-q: Robust planning for dialogue policy learning. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3813–3823.

Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. 2015. End-to-end memory networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'15, page 2440–2448, Cambridge, MA, USA. MIT Press.

Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. 2013. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pages 1139–1147.

Huimin Wang, Baolin Peng, and Kam Fai Wong. 2020. Learning efficient dialogue policy from demonstrations through shaping. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*.

Christopher JCH Watkins and Peter Dayan. 1992. Q-learning. *Machine learning*, 8(3-4):279–292.

Joseph Weizenbaum. 1966. Eliza—a computer program for the study of natural language communication between man and machine. *Communications of the ACM*, 9(1):36–45.

Jason Weston, Antoine Bordes, Sumit Chopra, and Tomás Mikolov. 2016. Towards ai-complete question answering: A set of prerequisite toy tasks. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

Christopher KI Williams and Carl Edward Rasmussen. 2006. *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA.

Yangyang Zhao, Zhenyu Wang, Kai Yin, Rui Zhang, Zhenhua Huang, and Pei Wang. 2020. Dynamic reward-based dueling deep dyna-q: Robust policy learning in noisy environments. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34:9676–9684.

Ciyou Zhu, Richard H Byrd, Peihuang Lu, and Jorge Nocedal. 1997. Algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound-constrained optimization. *ACM Transactions on Mathematical Software (TOMS)*, 23(4):550–560.

A Implementation Details

We implement our experiment on Thinkstation-P520 with Intel W-2223 CPU, 64G memory and two Nvidia GeForce RTX 2080 cards. And the average runtime for each DDQ and GPDDQQ approach are about from 2 to 3 hours and from 3 to 4.5 hours, respectively. For D3Q method, it takes about 2 days to run. The policy network $Q(s, a; \theta_Q)$ of direct reinforcement leaning in these models are approximated by deep neural network with `tanh` activations. It has one hidden layer with 80 hidden nodes. And the discount factor γ introduced in loss function is set to be 0.9. In our each GP model, there are 4 parameters need to be optimized. We limit the maximum length of a simulated dialogue to 40. In all our experiments, we only train the dialogue agents by interacting with user simulator which is publicly available. Only if the movie ticket is successfully booked and the information provided by the agent satisfies the constraints, the dialogue is considered successfully. If the dialogue is successful, the agent receives a positive reward of $2L$, otherwise, the reward value will be $-L$, where L is the defined maximum number of dialogue turns. Furthermore, shorter conversations are encouraged in this dialogue system since the agent will receive a reward of -1 per round. If there are no other instructions, in order to eliminate errors, each experiment are conducted five times to average.