Benchmarking Sample Selection Strategies for Batch Reinforcement Learning

Yuwei Fu **Di Wu** **Benoit Boulet**
McGill University
yuwei.fu@mail.mcgill.ca

Abstract

Training sample selection techniques, such as prioritized experience replay (PER), have been recognized as of significant importance for online reinforcement learning algorithms. Efficient sample selection can help further improve the learning efficiency and the final performance. However, the impact of sample selection for batch reinforcement learning (RL) has not been well studied. In this work, we investigate the application of non-uniform sampling techniques in batch RL. In particular, we compare six variants of PER based on various heuristic priority metrics that focus on different aspects of the offline learning setting. These metrics include temporal-difference error, n-step return, self-imitation learning objective, pseudo-count, uncertainty, and likelihood. Through extensive experiments on the standard batch RL datasets, we find that non-uniform sampling is also effective in batch RL settings. Further, there is no single metric that works in all situations. The investigation also shows that it is insufficient to avoid the bootstrapping error in batch reinforcement learning by only changing the sampling scheme.

1 Introduction

A key question in machine learning is to select the suitable training samples [30]. Many prior works showed that an appropriate sample selection strategy usually significantly improves the learning efficiency and final performance [5, 57, 15]. Similarly, sample selection also plays a crucial role in reinforcement learning (RL) [13]. A notable example is the sample selection problem for experience replay (ER) in off-policy RL [16], where an agent reuses stored experiences from a buffer. For example, Prioritized Experience Replay (PER) [57], which samples high error transitions more frequently, is widely used in different *state-of-the-art* (SOTA) off-policy RL algorithms [4, 26].

Batch RL, also known as offline RL, refers to the problem of learning a near-optimal policy from a fixed offline buffer [35]. Due to the wide availability of logged data and the increasing computing power, batch RL holds the promise for successful real-world applications [37]. Especially for the scenarios where collecting online data is time-consuming, dangerous or unethical, *i.e.*, robotics, self-driving cars and medical treatments [25]. While most off-policy RL algorithms are applicable in the offline setting, they usually suffer from the bootstrapping error [22, 33] due to out-of-distribution (OOD) samples. Different solutions have been proposed to mitigate this problem, *i.e.*, adding constraints [22, 66], imitating experts [10, 69], learning dynamics models [68, 31, 3], incorporating uncertainties [67], learning ensembles [1], or learning pessimistic value functions [34, 7, 28].

Unlike the wide application of PER in online off-policy RL, the non-uniform sampling strategy is largely ignored in recent batch RL algorithms. Inspired by the success of PER [57] in the online setting, one natural question to ask is that what is the counterpart of PER in batch RL? Some prior works proposed different sample selection strategies in batch RL. For example, Optimal Sample Selection (OSS) [54] introduced a meta-learning algorithm which selects optimal samples according to a cross entropy search method for tree-based Fitted Q-Iteration (FQI) [14] with a known dynamics model.

Offline Reinforcement Learning Workshop at Neural Information Processing Systems, 2021

Recently, Best-Action Imitation Learning (BAIL) [10] proposed to select high-performing samples with a learned value function in behavior cloning. Another related line of research is to reweight sampled transitions. For example, Advantage-Weighted Regression (AWR) [48] and Advantage-weighted Behavior Model (ABM) [58] used reward-weighted regression [49] to learn the policy. Further, Uncertainty Weighted Actor Critic (UWAC) [67] adopted a dropout-uncertainty estimation method [24] and reweighted samples using the estimated uncertainties. However, it is unclear which sample selection strategy is preferred in batch RL, thereby demanding more investigations.

In this work, we study the sample selection problem in batch RL [13]. We follow the PER framework by assigning samples with different priorities [57]. Crudely, there are two types of metrics to evaluate sample importance. Firstly, we can design a heuristic metric based on our prior knowledge, *i.e.*, temporal-difference (TD) error. Secondly, we can use an end-to-end approach to learn a metric for each sample, for example, we can use off-policy evaluation (OPE) methods [65, 19] to evaluate the goodness of current policy as the metric. However, existing OPE methods usually need to learn a model for each evaluation [36], which makes the learning-based metric approach to be computationally expensive. Therefore, in this paper, we focus on the heuristic metric-based approach and leave the learning-based metric approach for future work.

2 Preliminaries

2.1 Batch Reinforcement Learning

We consider the standard Markov Decision Process (MDP) [53] $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, T, r, \gamma \rangle$. $\mathcal{S}$ and $\mathcal{A}$ denote the state and action spaces. $T(s'|s,a)$ and $r(s,a)$ represent the dynamics and reward function, and $\gamma \in [0,1)$ is the discount factor. A policy $\pi(a|s)$ defines a mapping from state to distributions over actions. The goal of an RL agent is to learn a policy $\pi(a|s)$ that maximizes the expected cumulative discounted reward $J(\pi) := \mathbb{E}_\pi\left[\sum_{t=0}^{\infty} \gamma^t r_t\right]$. The performance of the policy can be defined by the Q-function $Q^\pi(s,a) := \mathbb{E}_\pi\left[\sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s, a_0 = a\right]$ and value function $V^\pi(s) := \mathbb{E}_\pi\left[\sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s\right]$, where $\mathbb{E}_\pi[\cdot]$ is the expected result when following the policy π. Once given the optimal Q-function $Q^*(s,a) = \arg\max_\pi Q^\pi(s,a)$, we can derive an optimal policy as $\pi^*(a|s) = \arg\max_a Q^*(s,a)$ [60].

In (tabular) Q-learning, we solve for the Q^* by iterating the Bellman Optimality Operator $\mathcal{T}^*$, defined as $\mathcal{T}^*Q(s,a) \leftarrow r + \gamma \max_{a'} Q(s',a')$ [6]. To solve problems with large state space, we can use a parameterized Q-function $Q_\theta(s,a)$ to approximate Q^*. In practice, we optimize the parameters by a μ-weighted L2 projection $\Pi_\mu(Q)$ [18], which minimizes the empirical Bellman error loss: $\Pi_\mu(Q) = \min_\theta \mathbb{E}_{(s,a,r,s')\sim\mu}\left[(\mathcal{T}^*Q_\theta(s,a) - Q_\theta(s,a))^2\right]$.

Batch RL, also known as offline RL, aims to learn a near-optimal policy from a fixed dataset [35] $\mathcal{D}$, representing a series of timestep tuples (s_t, a_t, r_t, s_{t+1}). Furthermore, the dataset can be collected by agents with different policies from different control tasks, including non-RL policies, such as human demonstrations [37]. Some early works such as Fitted Q-Iteration (FQI) [14] and Neural Fitted Q-Iteration (NFQ) [55], which formulate the original RL problem as a sequence of supervised regression problem, are shown to be sample efficient in solving various real-world problems [50, 12]. On the other hand, some recent studies show that current deep off-policy RL algorithms usually fail in challenging batch RL problems due to bootstrapping error [22, 33]. That is, the OOD action a' might lead to unrecoverable over-estimation error through max operator in the Bellman backup. The over-estimation problem is particularly detrimental in the offline setting where the agent has no access to interact with the real environment to get the feedback to fix the estimation error [34].

2.2 Non-uniform Sampling with Experience Replay

Experience replay (ER) [40] has been a de facto component for modern deep RL algorithms. By reusing previous collected experiences from the replay buffer, ER helps to reduce sample complexity and stabilize training in off-policy RL [42, 39]. For some real-world problems where collecting online data is expensive or time consuming, *i.e.*, robotics or self-driving cars, the ability to learn good policies from pre-collected data is crucial for successful real-world applications [8].

A number of works [57, 2, 41, 59, 23] show that applying different non-uniform sampling strategies in ER can significantly improve the learning efficiency. Especially for problems where there are many

redundant transitions [57], or the reward signal is sparse [2]. A notable example is the Prioritized Experience Replay (PER) [57], where the probability of sampling a certain transition (s_t, a_t, r_t, s_{t+1}) is proportional to the absolute TD error. However, it is still an open question that which priority metric is optimal to value the importance of samples [13].

3 Related Work

3.1 Sample Selection with Experience Replay

Many prior works have sought to analyze the mechanism of experience replay, both empirically [13, 16] and theoretically [23, 38]. Similar to our work, [13] investigated a number of proxies, *i.e.*, age, TD error, and exploration noise, to decide which experience to store in the replay buffer and how to sample from the replay buffer. Likewise, [18] used a "unit-testing" framework to study Q-learning with function approximators and found that a sampling scheme with wider coverage improves performance. Further, [16] conducted a systematic analysis of experience replay in Q-learning methods and provided two insights – (1) Increasing the buffer capacity is preferable, because it has a broader data coverage. (2) Decreasing the age of the oldest policy improves the performance, because it contains more high-quality on-policy data. While these insights help us to understand the mechanism of experience replay, they are less practical in the batch RL setting, where the given offline dataset is fixed [35].

A number of variants of ER have been introduced to further improve the learning efficiency [57, 2, 44, 41, 59]. One of the most popular variants is the Prioritized Experience Replay (PER) [57], which proposed to use the absolute TD error $|\delta(i)|$ as the priority metric and the probability $p(i)$ of sampling the i-th transition is:

$$p(i) = \frac{p_i^\alpha}{\sum_j p_j^\alpha}, \quad p_i = |\delta(i)| + \epsilon \quad \text{or} \quad p_i = \frac{1}{\text{rank}(i)}, \tag{1}$$

where α is a hyper-parameter, ϵ is a small positive constant to avoid zero priority, priority p_i is could be the value of $|\delta(i)|$ or the inverse rank of $|\delta(i)|$. In addition, Hindsight Experience Replay (HER) [2] proposed to re-label visited state as goal states to overcome hard exploration problems with sparse rewards. Competitive Experience Replay (CER) [41] later introduced an automatic exploratory curriculum by formulating an exploration competition between two agents. On the other hand, Remember and Forget Experience Replay (ReF-ER) [44] classified samples as "near-policy" and "far-policy" by the importance weight $\rho = \pi(a|s)/\mu(a|s)$ between current policy π and the behavior policy μ, and compute gradients only with near-policy samples. Similarly, Attentive Experience Replay (AER) [59] selects samples according to the similarities between the transition state and current state.

Recently, Loss-Adjusted Prioritized (LAP) experience replay [23] built the connection between the non-uniform sampling scheme in PER and loss functions. It shows that any loss function $\mathcal{L}_1$ evaluated with uniform sampling ($i \sim \mathcal{D}_1$) is equivalent to another loss function $\mathcal{L}_2$ that is evaluated with non-uniformly sampled data ($i \sim \mathcal{D}_2$):

$$\mathbb{E}_{i\sim\mathcal{D}_1}\left[\nabla_Q \mathcal{L}_1(\delta(i))\right] = \mathbb{E}_{i\sim\mathcal{D}_2}\left[\frac{p_{\mathcal{D}_1}(i)}{p_{\mathcal{D}_2}(i)}\nabla_Q \mathcal{L}_1(\delta(i))\right] = \mathbb{E}_{i\sim\mathcal{D}_2}\left[\nabla_Q \mathcal{L}_2(\delta(i)),\right] \tag{2}$$

where $\delta(i)$ is the TD error of the i-th sample and the two loss functions follows $\nabla_Q \mathcal{L}_2(\delta(i)) = \frac{p_{\mathcal{D}_1}(i)}{p_{\mathcal{D}_2}(i)}\nabla_Q \mathcal{L}_1(\delta(i))$. Moreover, Valuable Experience Replay (VER) [38] proved that the absolute TD error $|\delta(i)|$ is an upper-bound of different value metrics of experiences in Q-learning.

3.2 Sample Selection in Batch Reinforcement Learning

A pioneer work that applying sample selection strategy in batch RL is the Optimal Sample Selection (OSS) method [54]. More specifically, OSS is a model-based RL (MBRL) approach [29] where a known dynamics model is available to generate Monte Carlo rollouts for policy evaluation. Moreover, OSS introduced a meta-learning algorithm to select optimal samples according to the cross entropy search method [56] for tree-based Fitted Q-Iteration (FQI) [14]. Recently, Best-Action Imitation Learning (BAIL) proposed to learn a special value function $V_\phi(s)$, called upper envelope, that upper

bounds the cumulative discounted return $G_i = \sum_{t=i}^{T} \gamma^{t-i} r_t$ from starting from state s_i to the end of the episode (max horizon T). The learned upper envelope $V_\phi(S)$ is then used to filter high-quality samples to train a behavior cloning policy [52].

Another related line of research is to reweight samples [63, 48, 67]. Unlike previous methods that actively select samples from the buffer, these methods still adopt uniform sampling while assigning different weights to each sample to compute the loss function. For example, Advantage-Weighted Regression (AWR) [48] first formulated the RL problem as a supervised regression problem, and then used a learned value function to train the policy $\pi(a|s)$ via reward-weighted regression [49], which assigns higher weights to samples with large advantage values. Similarly, Advantage-weighted Behavior Model (ABM) [58] adopted reward-weighted regression in policy training to focus more on good actions. On the other hand, Uncertainty Weighted Actor Critic (UWAC) [67] used Monte Carlo Dropout [24] to approximate the epistemic uncertainty [11] for samples in batch RL dataset. The goal of UWAC is to assign lower weights to samples with higher epistemic uncertainty in order to mitigate the bootstrapping error caused by OOD state-action pairs [22, 33].

4 Methodology

4.1 Backbone Algorithms

In this work, we select TD3BC [20] and PER [57] as the backbone algorithms for benchmarking sample selection strategies in batch RL. TD3BC is a minimalist batch RL algorithm which simply adds a behavior cloning term to the TD3 algorithm [21]. While being simple, TD3BC achieves comparable performance w.r.t. other SOTA batch RL algorithms [32, 34] on the standard batch RL benchmark [17]. Moreover, TD3BC is able to run significantly faster than previous methods by removing additional computations overheads.

In terms of the non-uniform sampling strategy, we follow the PER framework [57] in which the probability to sample transition i is $p(i) = p_i^\alpha / \sum_j p_j^\alpha$, where p_i is the priority of transition i and parameter α determines how much prioritization is used. In this paper, we investigate the problem of how the choice of priority metric matters in batch RL. We use both the proportional PER and rank-based PER in the experiment depending on the used priority metric.

4.2 Proposed Metrics

Here, we introduce six different priority metrics that we use in the experiment (Table 1).

Table 1: Depends on if the metric changes during the training, we can devide it as dynamic or static metic. Some metrics are more computationally expensive which require extra computation.

Metric	Type	Motivation	Prioritization	Extra computation
TD-Error	Dynamic	Reducing redundant samples	Proportional PER	-
N-step Return	Static	Selecting good samples	Rank PER	-
GSIL	Dynamic	Selecting good samples	Proportional PER	A second buffer
Pseudo-count	Static	Avoiding OOD samples	Rank PER	Hash table
Uncertainty	Static	Avoiding OOD samples	Rank PER	Probabilistic ensemble
Likelihood	Static	Being more on-policy	Rank PER	Behavior policy

TD error. We first select TD error as our primary baseline, and test how well does the naïve PER [57] perform in batch RL setting. The priority for the i-th transition is $p_i = |\delta(i)| + \epsilon$, where $|\delta(i)|$ is the absolute TD error and ϵ is a small positive constant to avoid zero priority. The motivation is that samples with small absolute TD error may contain less information for our model to learn from [43].

N-step return. We then select the n-step return as the proxy to evaluate the goodness of samples. We hypothesis that samples with higher n-step return is more likely to be high-quality samples. Unlike Monte Carlo return which requires a full trajectory, n-step return is more practical in real-world

problems, where we usually only have partial trajectories without a terminal state. Given the different reward scales across tasks, we use a rank based PER in the experiment.

Generalized SIL. Our third metric is inspired by the Self-Imitation Learning (SIL) [45], which exploits past good experiences. In particular, SIL imitates past good experiences by optimizing following actor-critic loss functions:

$$\mathcal{L}^{sil}_{value} = \frac{1}{2}\| \left[R - V_\theta(s)\right]_+ \|^2, \quad \mathcal{L}^{sil}_{policy} = -\log \pi_\theta(a|s) \left[R - V_\theta(s)\right]_+, \tag{3}$$

where $R = \sum_{t=0}^{\infty} \gamma^t r_t$ is the cumulative discounted return starting from state s after taking action a, and $[x]_+ = \max(0, x)$. The motivation of SIL is intuitive that policy $\pi_\theta(a|s)$ should imitate action a if it is high-performing, such that $R > V_\theta(s)$. Generalized Self-Imitation Learning (GSIL) [62] later extends the original SIL to deterministic actor-critic setting with n-step TD-learning. We follow GSIL to set the priority for the i-th transition to be $p_i = \left[R^N_i - Q_\theta(s_i, a_i)\right]_+ + \epsilon$, where R^N_i is the n-step return and ϵ is a small positive constant.

Pseudo-count. Although batch RL does not concern the exploration problem [46]. We attempt to borrow some insights from an exploration perspective to distinguish useful samples. In particular, we test the efficacy of Pseudo-count [47, 61] for sample selection in batch RL. We follow the #Exploration [61] model to use locality-sensitive hashing (LSH) method, *i.e.*, SimHash [9], to convert continuous state s to discrete hash codes $\phi(s) = sgn(Ag(s))$, where $g(\cdot)$ is a preprocessing function and A is a random matrix. We use a rank-based PER and set the priority of the i-th sample to be $p_i = 1/\text{rank}(N_i)$.

Uncertainty. In addition, we also try to use the epistemic uncertainty to evaluate the sample importance. In the experiment, we adopt the probabilistic ensemble [11] method to evaluate sample uncertainty. We first train an ensemble of M probabilistic dynamic models $\{T_1, \cdots, T_M\}$, and each dynamic model $T_i(s_{t+1}|s_t, a_t) = \mathcal{N}(\mu_{\theta_i}(s_t, a_t), \Sigma_{\theta_i}(s_t, a_t))$ outputs a Gaussian distribution with diagonal covariances. For the i-th transition, we approximate its epistemic uncertainty by the standard deviation of $\sigma_i = \text{std}\left(\{\mu_{\theta_1}(s_i, a_i), \cdots, \mu_{\theta_M}(s_i, a_i)\}\right)$. We use a rank-based PER to assign higher priority to samples with smaller uncertainty, that is $p_i = 1/\text{rank}(\sigma_i^{-1})$.

Likelihood. The last metric we test in the experiment is the likelihood of the behavior model [32]. Similar to previous constrained based batch RL algorithms [22, 66, 33], we want to make the learned policy $\pi(a|s)$ to stay close to the behavior policy $\mu(a|s)$. Therefore, we first learned a behavior policy with a mixture of Gaussian model [32] and used the likelihood as the priority. We use a rank-based PER where $p_i = 1/\text{rank}(\log \mu(a_i|s_i))$ is the priority for the i-th sample.

5 Experiment

In this section, we compare different PER variants with the proposed metrics on standard batch RL benchmarks [17]. We seek to address the following questions in the experiments: (1) Does non-uniform sampling scheme also help to improve the performance in batch RL? (2) Which priority metric is preferred in the batch RL setting? We first introduce the dataset and the experiment setups. Then we present our main experiment result and discuss some limitations of our method.

Datasets. We evaluate different sample selection strategies on the widely-used D4RL gym Mujoco benchmark [64, 17], including three environments (halfcheetah, hopper, and walker2d) and five dataset types (random, medium, medium-replay, medium-expert, expert), yielding a total of 15 datasets. These datasets differ in many aspects, e.g., number of transitions, quality of behavior policy, and data coverage. We seek to validate the robustness of each sample selection strategy in different domains.

Experiment setup. For the backbone algorithm, we use the author-provided implementation for TD3BC. We maintain two replay buffers for the GSIL metric as in the origin paper[62], where the first buffer stores single-step transitions to train TD3 and the second buffer stores n-step transitions to compute the GSIL loss. In addition, we use the MBRL-Lib[51] to train the probabilistic ensemble, and implement the SimHash according to EPG[27]. Parameters for the PER are taken from the original paper [57]. We follow exactly the same experimental setup as [20], in which we train for 1 million time steps and evaluate every 5000 time steps for 10 episodes. More details are in the Appendix.

Results. We report the final performance of different priority metrics in Table 2 and plot the learning curves in Figure 1. We make several observations: (1) Non-uniform sampling strategy is also effective in batch RL, for example, the most performant method in each environment is usually a non-uniform sampling strategy. (2) There is no single metric that is consistently the best performer. (3) In some environments, such as Hopper-Medium and Hopper-Expert, different sampling schemes perform very similar. In light of these results, we conclude that offline datasets are quite complicate and multiple factors can influence the sample priority. In environments with relatively low dimensions, such as Hopper, the learned policy is less affected by the sampling scheme.

Table 2: Performance of different priority metrics in the D4RL datasets. We report the average normalized score over the final 10 evaluations over 3 seeds ($\pm$ standard deviation).

		Uniform	TD-Error	Nstep-Return	GSIL	Pseudo-Count	Uncertainty	Likelihood
Random	HalfCheetah	11.2 $\pm$ 1.3	11.1 $\pm$ 1.1	10.3 $\pm$ 0.6	9.1 $\pm$ 2.0	11.3 $\pm$ 1.3	**11.4 $\pm$ 1.2**	11.0 $\pm$ 0.6
	Hopper	11.0 $\pm$ 0.0	10.9 $\pm$ 0.1	11.0 $\pm$ 0.0	10.9 $\pm$ 0.0	**11.1 $\pm$ 0.0**	10.8 $\pm$ 0.1	11.0 $\pm$ 0.0
	Walker2d	0.9 $\pm$ 0.6	1.7 $\pm$ 1.1	2.6 $\pm$ 0.8	**5.1 $\pm$ 0.3**	2.4 $\pm$ 0.6	2.3 $\pm$ 1.7	1.8 $\pm$ 0.6
Medium	HalfCheetah	42.9 $\pm$ 0.1	42.8 $\pm$ 0.3	**43.9 $\pm$ 0.5**	43.2 $\pm$ 0.2	43.3 $\pm$ 0.4	42.9 $\pm$ 0.4	42.4 $\pm$ 0.1
	Hopper	**99.9 $\pm$ 0.1**	99.6 $\pm$ 0.4	99.4 $\pm$ 0.6	99.8 $\pm$ 0.1	99.7 $\pm$ 0.1	99.8 $\pm$ 0.1	99.8 $\pm$ 0.2
	Walker2d	77.3 $\pm$ 0.9	78.2 $\pm$ 1.0	77.3 $\pm$ 1.2	77.9 $\pm$ 1.3	77.2 $\pm$ 0.7	76.9 $\pm$ 0.6	**79.4 $\pm$ 0.6**
Medium Replay	HalfCheetah	43.1 $\pm$ 0.4	43.3 $\pm$ 0.1	**43.5 $\pm$ 0.5**	42.8 $\pm$ 0.2	43.3 $\pm$ 0.5	43.4 $\pm$ 0.2	43.3 $\pm$ 0.0
	Hopper	**32.1 $\pm$ 1.3**	30.3 $\pm$ 0.8	31.4 $\pm$ 0.7	30.6 $\pm$ 1.9	31.9 $\pm$ 0.3	31.1 $\pm$ 1.8	31.7 $\pm$ 1.6
	Walker2d	24.3 $\pm$ 4.6	23.8 $\pm$ 2.5	17.4 $\pm$ 2.7	15.2 $\pm$ 9.4	**29.0 $\pm$ 3.6**	26.4 $\pm$ 1.0	24.8 $\pm$ 1.1
Medium Expert	HalfCheetah	92.4 $\pm$ 1.5	**96.9 $\pm$ 1.7**	87.8 $\pm$ 3.4	96.5 $\pm$ 2.1	91.3 $\pm$ 1.7	88.0 $\pm$ 3.3	84.4 $\pm$ 4.0
	Hopper	112.0 $\pm$ 0.1	106.2 $\pm$ 1.1	110.7 $\pm$ 1.7	111.6 $\pm$ 0.9	**112.2 $\pm$ 0.0**	109.8 $\pm$ 2.2	111.4 $\pm$ 0.6
	Walker2d	95.7 $\pm$ 4.2	96.9 $\pm$ 3.0	90.8 $\pm$ 2.6	103.1 $\pm$ 4.1	96.9 $\pm$ 4.6	**103.9 $\pm$ 2.1**	81.4 $\pm$ 23.6
Expert	HalfCheetah	**105.9 $\pm$ 0.7**	103.5 $\pm$ 1.5	102.4 $\pm$ 1.6	105.4 $\pm$ 1.0	104.0 $\pm$ 1.2	103.8 $\pm$ 0.2	104.5 $\pm$ 0.9
	Hopper	**112.3 $\pm$ 0.0**	112.2 $\pm$ 0.1	112.2 $\pm$ 0.1	112.1 $\pm$ 0.1	112.2 $\pm$ 0.1	111.8 $\pm$ 0.6	44.6 $\pm$ 47.9
	Walker2d	105.0 $\pm$ 2.0	104.5 $\pm$ 1.8	105.5 $\pm$ 1.7	103.8 $\pm$ 1.3	104.2 $\pm$ 1.1	**105.9 $\pm$ 0.4**	104.1 $\pm$ 3.3
# Beat baseline		-	5	4	5	**8**	6	5

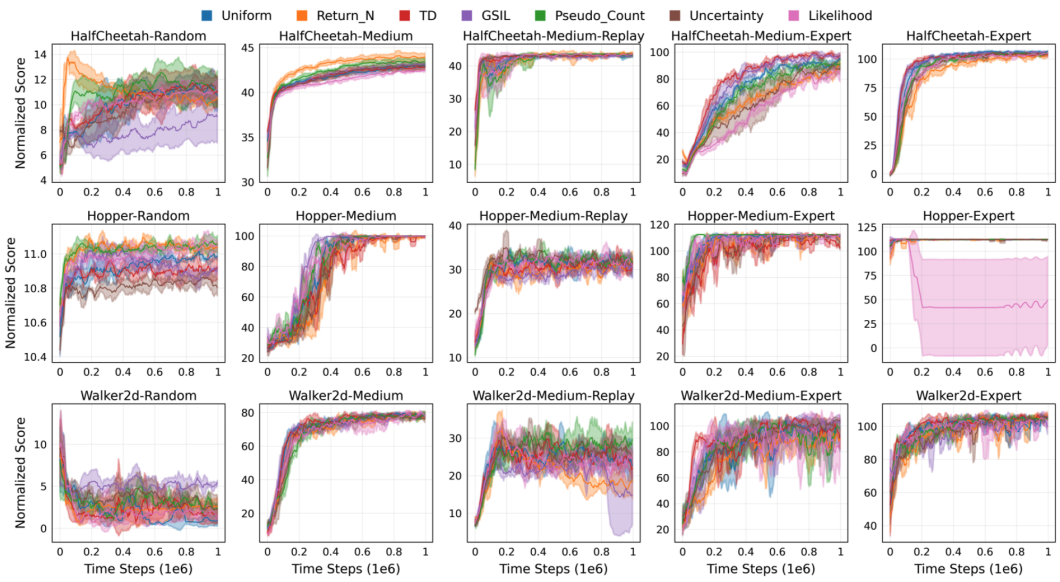

Figure 1: Results are averaged across 3 random seeds with shaded areas representing the standard deviation. We can observe that non-uniform sampling strategies is also effective in batch RL, and there is no priority metric that works in all situations.

Sampling scheme and bootstrapping error. We plot the learned Q_1 function in TD3 (Figure 2). We can see that all the sampling schemes learn explosive Q values in the Walker2d-Random environment, which implies that non-uniform sampling schemes fail to avoid the bootstrapping error.

We also observe that the sampling scheme affects the learned Q function, where the likelihood metric usually learns a relatively smaller Q values and N-step return learns a relatively higher Q values. This corresponds to the inductive bias of each metric. For example, a transition with higher N-step return is more likely to have large single-step return, which leads to higher Q values in the Bellman backup. In addition, we can also observe that the bootstrapping error is not the only problem which prevents us to learn a good offline policy. For example, we did not suffer from severe bootstrapping error in HalfCheetah-Random and Hopper-Random environment, but we still fail to exploit the offline dataset to learn performant policies.

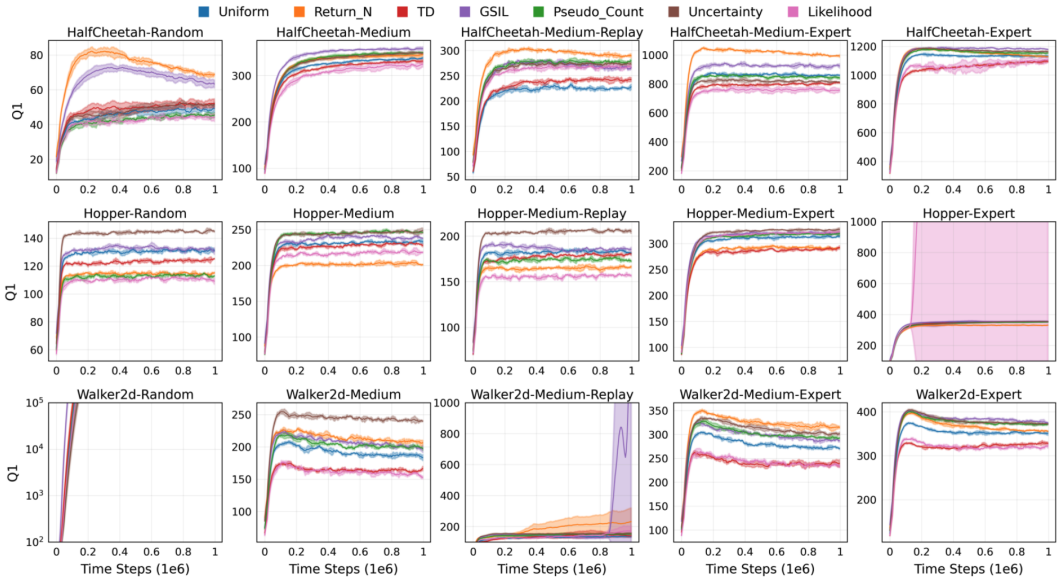

Figure 2: Learned Q_1 function in TD3. We use a discount factor $\gamma = 0.99$.

Some problems of the proposed metrics. We summarize some shortcomings of the heuristic metric-based sample selection strategies. Firstly, we may need extra computations to compute the priority metrics. In addition, these extra models require further parameter tuning which may be problematic in the offline setting. Secondly, another critical problem of the metric-based sample selection method is that how exact is the metric for selecting a good sample. It might be better to use these metrics as thresholds to filter bad samples. For example, a low uncertainty transition may not be a good sample, but a high uncertainty transition is more likely to be a bad one. Thirdly, in the experiment, we compute the priority metric metric for transition (s_i, a_i, r_i, s_{i+1}) based the current state-action pair (s_i, a_i). However, in the batch RL setting, the bootstrapping error comes from the OOD state-action pair (s_{i+1}, a_{i+1}). Therefore, it might be more effective to compute the priority metric based on the next state-action pair (s_{i+1}, a_{i+1}). We leave these shortcomings for future work.

6 Conclusion and Future Work

In this paper, we performed empirical analysis on non-uniform sample selection strategies in batch reinforcement learning (RL). In particular, we compared different variants of Prioritized Experience Replay (PER) based on various heuristic sample priority metrics, and our experiments showed that non-uniform sampling is also effective in the batch RL setting. However, there is no single priority metric that works in all situations. Our preliminary investigation showed that offline dataset is quite complicate and multiple factors can influence the sample priority. With the growing number of available offline datasets, we believe that a better sample selection strategy holds the promise to help us learn more performant batch RL policies. One limitation of our work is that the proposed metric only focuses on current state-action pairs and requires extra computations. In the future, we plan to do more evaluations on different data sets and investigate learning a priority metric end-to-end with off-policy policy evaluation (OPE) methods.

References

[1] Rishabh Agarwal, Dale Schuurmans, and Mohammad Norouzi. An optimistic perspective on offline reinforcement learning. In *International Conference on Machine Learning*, pages 104–114. PMLR, 2020.

[2] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. *arXiv preprint arXiv:1707.01495*, 2017.

[3] Arthur Argenson and Gabriel Dulac-Arnold. Model-based offline planning. *arXiv preprint arXiv:2008.05556*, 2020.

[4] Gabriel Barth-Maron, Matthew W Hoffman, David Budden, Will Dabney, Dan Horgan, Dhruva Tb, Alistair Muldal, Nicolas Heess, and Timothy Lillicrap. Distributed distributional deterministic policy gradients. *arXiv preprint arXiv:1804.08617*, 2018.

[5] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48, 2009.

[6] Dimitri P Bertsekas and John N Tsitsiklis. *Neuro-dynamic programming*. Athena Scientific, 1996.

[7] Jacob Buckman, Carles Gelada, and Marc G Bellemare. The importance of pessimism in fixed-dataset policy optimization. *arXiv preprint arXiv:2009.06799*, 2020.

[8] Serkan Cabi, Sergio Gómez Colmenarejo, Alexander Novikov, Ksenia Konyushkova, Scott Reed, Rae Jeong, Konrad Zolna, Yusuf Aytar, David Budden, Mel Vecerik, et al. Scaling data-driven robotics with reward sketching and batch reinforcement learning. *arXiv preprint arXiv:1909.12200*, 2019.

[9] Moses S Charikar. Similarity estimation techniques from rounding algorithms. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 380–388, 2002.

[10] Xinyue Chen, Zijian Zhou, Zheng Wang, Che Wang, Yanqiu Wu, Qing Deng, and Keith Ross. Bail: Best-action imitation learning for batch deep reinforcement learning. *arXiv preprint arXiv:1910.12179*, 2019.

[11] Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. *arXiv preprint arXiv:1805.12114*, 2018.

[12] Joao Cunha, Rui Serra, Nuno Lau, Luís Seabra Lopes, and Antóio JR Neves. Batch reinforcement learning for robotic soccer using the q-batch update-rule. *Journal of Intelligent & Robotic Systems*, 80(3):385–399, 2015.

[13] Tim De Bruin, Jens Kober, Karl Tuyls, and Robert Babuska. Experience selection in deep reinforcement learning for control. *Journal of Machine Learning Research*, 19, 2018.

[14] Damien Ernst, Pierre Geurts, and Louis Wehenkel. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, 6:503–556, 2005.

[15] Yang Fan, Fei Tian, Tao Qin, Jiang Bian, and Tie-Yan Liu. Learning what data to learn. *arXiv preprint arXiv:1702.08635*, 2017.

[16] William Fedus, Prajit Ramachandran, Rishabh Agarwal, Yoshua Bengio, Hugo Larochelle, Mark Rowland, and Will Dabney. Revisiting fundamentals of experience replay. In *International Conference on Machine Learning*, pages 3061–3071. PMLR, 2020.

[17] Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

[18] Justin Fu, Aviral Kumar, Matthew Soh, and Sergey Levine. Diagnosing bottlenecks in deep q-learning algorithms. In *International Conference on Machine Learning*, pages 2021–2030. PMLR, 2019.

[19] Justin Fu, Mohammad Norouzi, Ofir Nachum, George Tucker, Ziyu Wang, Alexander Novikov, Mengjiao Yang, Michael R Zhang, Yutian Chen, Aviral Kumar, et al. Benchmarks for deep off-policy evaluation. *arXiv preprint arXiv:2103.16596*, 2021.

[20] Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. *arXiv preprint arXiv:2106.06860*, 2021.

[21] Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*, pages 1587–1596. PMLR, 2018.

[22] Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. 2018.

[23] Scott Fujimoto, David Meger, and Doina Precup. An equivalence between loss functions and non-uniform sampling in experience replay. *arXiv preprint arXiv:2007.06049*, 2020.

[24] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059. PMLR, 2016.

[25] Caglar Gulcehre, Ziyu Wang, Alexander Novikov, Tom Le Paine, Sergio Gomez Colmenarejo, Konrad Zolna, Rishabh Agarwal, Josh Merel, Daniel Mankowitz, Cosmin Paduraru, et al. Rl unplugged: Benchmarks for offline reinforcement learning. *arXiv e-prints*, pages arXiv–2006, 2020.

[26] Matteo Hessel, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. In *Thirty-second AAAI conference on artificial intelligence*, 2018.

[27] Rein Houthooft, Richard Y. Chen, Phillip Isola, Bradly C. Stadie, Filip Wolski, Jonathan Ho, and Pieter Abbeel. Evolved policy gradients. *arXiv preprint arXiv:1802.04821*, 2018.

[28] Ying Jin, Zhuoran Yang, and Zhaoran Wang. Is pessimism provably efficient for offline rl? In *International Conference on Machine Learning*, pages 5084–5096. PMLR, 2021.

[29] Lukasz Kaiser, Mohammad Babaeizadeh, Piotr Milos, Blazej Osinski, Roy H Campbell, Konrad Czechowski, Dumitru Erhan, Chelsea Finn, Piotr Kozakowski, Sergey Levine, et al. Model-based reinforcement learning for atari. *arXiv preprint arXiv:1903.00374*, 2019.

[30] Angelos Katharopoulos and François Fleuret. Not all samples are created equal: Deep learning with importance sampling. In *International conference on machine learning*, pages 2525–2534. PMLR, 2018.

[31] Rahul Kidambi, Aravind Rajeswaran, Praneeth Netrapalli, and Thorsten Joachims. Morel: Model-based offline reinforcement learning. *arXiv preprint arXiv:2005.05951*, 2020.

[32] Ilya Kostrikov, Rob Fergus, Jonathan Tompson, and Ofir Nachum. Offline reinforcement learning with fisher divergence critic regularization. In *International Conference on Machine Learning*, pages 5774–5783. PMLR, 2021.

[33] Aviral Kumar, Justin Fu, George Tucker, and Sergey Off-policy deep reinforcement learning without exploration. Stabilizing off-policy q-learning via bootstrapping error reduction. *arXiv preprint arXiv:1906.00949*, 2019.

[34] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *arXiv preprint arXiv:2006.04779*, 2020.

[35] Sascha Lange, Thomas Gabel, and Martin Riedmiller. Batch reinforcement learning. In *Reinforcement learning*, pages 45–73. Springer, 2012.

[36] Hoang Le, Cameron Voloshin, and Yisong Yue. Batch policy learning under constraints. In *International Conference on Machine Learning*, pages 3703–3712. PMLR, 2019.

[37] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

[38] Ang A Li, Zongqing Lu, and Chenglin Miao. Revisiting prioritized experience replay: A value perspective. *arXiv preprint arXiv:2102.03261*, 2021.

[39] 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.

[40] Long-Ji Lin. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine learning*, 8(3-4):293–321, 1992.

[41] Hao Liu, Alexander Trott, Richard Socher, and Caiming Xiong. Competitive experience replay. *arXiv preprint arXiv:1902.00528*, 2019.

[42] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[43] Andrew W Moore and Christopher G Atkeson. Prioritized sweeping: Reinforcement learning with less data and less time. *Machine learning*, 13(1):103–130, 1993.

[44] Guido Novati and Petros Koumoutsakos. Remember and forget for experience replay. In *International Conference on Machine Learning*, pages 4851–4860. PMLR, 2019.

[45] Junhyuk Oh, Yijie Guo, Satinder Singh, and Honglak Lee. Self-imitation learning. In *International Conference on Machine Learning*, pages 3878–3887. PMLR, 2018.

[46] Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep exploration via bootstrapped dqn. *Advances in neural information processing systems*, 29:4026–4034, 2016.

[47] Georg Ostrovski, Marc G Bellemare, Aäron Oord, and Rémi Munos. Count-based exploration with neural density models. In *International conference on machine learning*, pages 2721–2730. PMLR, 2017.

[48] Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*, 2019.

[49] Jan Peters, Katharina Mulling, and Yasemin Altun. Relative entropy policy search. In *Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010.

[50] Olivier Pietquin, Matthieu Geist, Senthilkumar Chandramohan, and Hervé Frezza-Buet. Sample-efficient batch reinforcement learning for dialogue management optimization. *ACM Transactions on Speech and Language Processing (TSLP)*, 7(3):1–21, 2011.

[51] Luis Pineda, Brandon Amos, Amy Zhang, Nathan O Lambert, and Roberto Calandra. Mbrl-lib: A modular library for model-based reinforcement learning. *arXiv preprint arXiv:2104.10159*, 2021.

[52] Dean A Pomerleau. Efficient training of artificial neural networks for autonomous navigation. *Neural computation*, 3(1):88–97, 1991.

[53] Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[54] Emmanuel Rachelson, François Schnitzler, Louis Wehenkel, and Damien Ernst. Optimal sample selection for batch-mode reinforcement learning. In *Proceedings of the 3rd International Conference on Agents and Artificial Intelligence (ICAART 2011)*, 2011.

[55] Martin Riedmiller. Neural fitted q iteration–first experiences with a data efficient neural reinforcement learning method. In *European conference on machine learning*, pages 317–328. Springer, 2005.

[56] Reuven Y Rubinstein and Dirk P Kroese. The cross-entropy method: A unified approach to monte carlo simulation, randomized optimization and machine learning. *Information Science & Statistics, Springer Verlag, NY*, 2004.

[57] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015.

[58] Noah Y Siegel, Jost Tobias Springenberg, Felix Berkenkamp, Abbas Abdolmaleki, Michael Neunert, Thomas Lampe, Roland Hafner, Nicolas Heess, and Martin Riedmiller. Keep doing what worked: Behavioral modelling priors for offline reinforcement learning. *arXiv preprint arXiv:2002.08396*, 2020.

[59] Peiquan Sun, Wengang Zhou, and Houqiang Li. Attentive experience replay. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 5900–5907, 2020.

[60] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[61] Haoran Tang, Rein Houthooft, Davis Foote, Adam Stooke, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. # exploration: A study of count-based exploration for deep reinforcement learning. In *31st Conference on Neural Information Processing Systems (NIPS)*, volume 30, pages 1–18, 2017.

[62] Yunhao Tang. Self-imitation learning via generalized lower bound q-learning. *arXiv preprint arXiv:2006.07442*, 2020.

[63] Andrea Tirinzoni, Andrea Sessa, Matteo Pirotta, and Marcello Restelli. Importance weighted transfer of samples in reinforcement learning. In *International Conference on Machine Learning*, pages 4936–4945. PMLR, 2018.

[64] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012.

[65] Cameron Voloshin, Hoang M Le, Nan Jiang, and Yisong Yue. Empirical study of off-policy policy evaluation for reinforcement learning. *arXiv preprint arXiv:1911.06854*, 2019.

[66] Yifan Wu, George Tucker, and Ofir Nachum. Behavior regularized offline reinforcement learning. *arXiv preprint arXiv:1911.11361*, 2019.

[67] Yue Wu, Shuangfei Zhai, Nitish Srivastava, Joshua Susskind, Jian Zhang, Ruslan Salakhutdinov, and Hanlin Goh. Uncertainty weighted actor-critic for offline reinforcement learning. *arXiv preprint arXiv:2105.08140*, 2021.

[68] Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. Mopo: Model-based offline policy optimization. *arXiv preprint arXiv:2005.13239*, 2020.

[69] Konrad Zolna, Alexander Novikov, Ksenia Konyushkova, Caglar Gulcehre, Ziyu Wang, Yusuf Aytar, Misha Denil, Nando de Freitas, and Scott Reed. Offline learning from demonstrations and unlabeled experience. *arXiv preprint arXiv:2011.13885*, 2020.

A Appendix

Here, we introduce some details of our experiments. For the PER model, we use the hyperparameters recommended in the origin paper [57], and set $\alpha = 0.6$, $\beta = 0.4$. For the N-step return metric, we select the N to be 20. For the GSIL metric, we follow the same experiment setup as in the origin GSIL paper [62]. For the pseudo-count metric, we select the key dimension for Simhash by computing the 25% quantile and 50% quantile number (see Table 3). A small key dimension would lead to too many collisions while a large key dimension would lead to sparse collisions. We highlight the selected parameter for each environment we used in the experiment. For the uncertainty metric, we train an probabilistic ensemble with 7 models with early stopping. We use the default training parameters as in the MBRL-LIB [51] package. For the likelihood metric, we use the official FBRC [32] code to learn the behavior policy.

Table 3: Quantile number of the pseudo-count of each state-action pair in the offline dataset.

Key Dimension		16		24		32		48		64		128	
Quantile		25%	50%	25%	50%	25%	50%	25%	50%	25%	50%	25%	50%
Random	HalfCheetah	29	85	**1**	**3**	1	1	1	1	1	1	1	1
	Hopper	1068	4377	321	1697	62	343	**6**	**42**	2	11	1	1
	Walker2d	56	201	**2**	**10**	1	3	1	1	1	1	1	1
Medium	HalfCheetah	670	3288	112	727	21	188	**3**	**28**	1	4	1	1
	Hopper	562	1931	125	626	41	208	**7**	**43**	2	13	1	1
	Walker2d	99	443	6	40	**1**	**7**	1	2	1	1	1	1
Medium Replay	HalfCheetah	7	29	**1**	**3**	1	1	1	1	1	1	1	1
	Hopper	57	205	8	37	**2**	**10**	1	1	1	1	1	1
	Walker2d	6	18	**1**	**2**	1	1	1	1	1	1	1	1
Medium Expert	HalfCheetah	838	4309	192	1241	23	210	**4**	**43**	1	8	1	1
	Hopper	405	1354	73	326	26	135	**4**	**23**	1	4	1	1
	Walker2d	218	1069	14	90	**3**	**23**	1	2	1	2	1	1
Expert	HalfCheetah	638	3831	149	947	23	198	**2**	**12**	1	2	1	1
	Hopper	1032	3683	175	718	25	123	**4**	**24**	2	8	1	1
	Walker2d	160	655	17	103	**4**	**25**	1	3	1	1	1	1