Guided Adaptive Credit Assignment for Sample Efficient Policy Optimization

Anonymous authors
Paper under double-blind review

Abstract

Policy gradient methods have achieved remarkable successes in solving challenging reinforcement learning problems. However, it still often suffers from sparse reward tasks, which leads to poor sample efficiency during training. In this work, we propose a guided adaptive credit assignment method to do effectively credit assignment for policy gradient methods. Motivated by entropy regularized policy optimization, our method extends the previous credit assignment methods by introducing a more general credit assignment named guided adaptive credit assignment(GACA). The benefit of GACA is a principled way of utilizing off-policy samples. The effectiveness of proposed algorithm is demonstrated on the challenging WikiTableQuestions and WikiSQL benchmarks and an instruction following environment. The task is generating action sequences or program sequences from natural language questions or instructions, where only final binary success-failure execution feedback is available. Empirical studies show that our method significantly improves the sample efficiency of the state-of-the-art policy optimization approaches.

1 Introduction

Deep reinforcement learning (RL) provides a general framework for solving challenging goal-oriented sequential decision-making problems, it has recently achieved remarkable successes in advancing the frontier of AI technologies (Silver et al., 2016; Mnih & Kavukcuoglu, 2013; Silver et al., 2017; Andrychowicz et al., 2017). Policy gradient (PG) (Kakade, 2002; Mnih et al., 2016; Schulman et al., 2015) is one of the most successful model-free RL approaches that has been widely applied to high dimensional continuous control, vision-based robotics, playing video games, and program synthesis (Liang et al., 2018; Guu et al., 2017; Bunel et al., 2018).

Despite these successes, a key problem of policy gradient methods is that it often suffers from high sample complexity in sparse reward tasks. In sparse reward tasks, there is only a binary signal which indicate successful task completion but without carefully shaped reward function to properly guide the policy optimization. A naive yet effective solution to address this challenge is by exploring many diverse samples and re-labelling visited states as goal states during training (see e.g. Andrychowicz et al., 2017; Pong et al., 2019). Regardless of the cost of generating large samples and the bias introduced during comparison, in many practical applications like program synthesis, it may not even be possible to compare between different states. A variety of credit assignment techniques have been proposed for policy gradient methods in settings where comparison of states is not available (See e.g. Liang et al. 2018, Agarwal et al. 2019, and Norouzi et al. 2016).

In this work, we focus on entropy regularized reinforcement learning. Instead of directly optimizing the RL objective, which is hard in sparse reward tasks, we sort to optimize policy to approximate a learnable prior distribution called guiding prior distribution. By using so-called f-divergence (Csiszár et al., 2004; Liese & Vajda, 2006; Nowozin et al., 2016; Wang et al., 2018) which defines a broad class of divergence(*e.g.*, KL and reverse KL divergence) that are sufficient to fully characterize the distributions under consideration, we construct a class of gradient estimator that allow us to generalize previous credit assignment methods. The neat property is that the gradient estimator can adaptively optimize policy based on divergence between itself and the prior distribution. It is natural to expect this more flexible gradient estimator provide an adaptive trade-off between different credit assignment methods, in addition, it also has a good property such that all off-policy samples are utilized to compute gradient, which can yield powerful credit assignment. Our approach tremendously extends the existing credit assignment used including REINFORCE (Sutton et al., 2000; Williams, 1992), maximum marginal likelihood(MML) (Dempster et al., 1977; Guu et al., 2017), MAPO (Liang et al., 2018), iterative maximum likelihood(IML) (Liang et al., 2017; Abolafia et al., 2018), and RAML (Norouzi et al., 2016).

We evaluate our method on a variety of tasks, including the challenging WikiSQL (Zhong et al., 2017) and WikiTableQuestions (Pasupat & Liang, 2015) program synthesis benchmarks, and an instruction following navigation task TextWorld (Agarwal et al., 2019). Our experiments show that GACA greatly improves the sample efficiency of the entire policy optimization, and leads to significant higher asymptotic performance over previous state-of-the-art methods.

2 Background

2.1 Reinforcement Learning and Policy Optimization

Reinforcement learning(RL) considers the problem of finding an optimal policy for an agent that interacts with an uncertain environment and collects reward per action. The goal of the agent is to maximize its cumulative reward. Formally, this problem can be viewed as a Markov decision process over the environment states $s \in S$ and agent actions $z \in Z$, with the environment dynamics defined by the transition probability $T(s'|s, z)$ and reward function $r(s_t, z_t)$, which yields a reward immediately following the action z_t performed in state s_t. The agent's action z is selected by a conditional probability distribution $\pi(z|s)$ called policy.

In policy gradient methods, we consider a set of candidate policies $\pi_\theta(z|s)$ parameterized by θ and obtain the optimal policy by maximizing the expected cumulative reward or return

$$J(\theta) = \mathbb{E}_{s\sim\rho_\pi, z\sim\pi(z|s)}\left[r(s, z)\right],$$

where $\rho_\pi(s) = \mathbb{E}_{t=1}^{\infty}\gamma^{t-1}\mathrm{Pr}(s_t = s)$ is the normalized discounted state visitation distribution with discount factor $\gamma \in [0, 1)$.

2.2 Sparse Reward Reinforcement Learning and Credit Assignment

Auto-regressive model is often used as a policy in many real world applications including program synthesis and combinational optimization (Liang et al., 2018; Guu et al., 2017). In this work, we consider the following form of policy distribution.

$$\pi_\theta(\mathbf{z}|s_0) = \prod_{i=t}^{|\mathbf{z}|} \pi(z_t \mid \mathbf{z}_{<t}, s_0), \tag{1}$$

where $\mathbf{z}_{<t} = (z_1, \ldots, z_{t-1})$ denotes a prefix of the action sequence $\mathbf{z}$, $s_0 \in \mathcal{Z}$ denotes some context information about the task, such as initial state or goal state (Andrychowicz et al., 2017). And $\pi_\theta(\mathbf{z}|s_0)$ satisfy $\forall \mathbf{z} \in \mathcal{Z} : \ \pi_\theta(\mathbf{z}|s_0) \geq 0$ and $\mathbb{E}_{\mathbf{z}\in\mathcal{Z}}\pi_\theta(\mathbf{z}|s_0) = 1$.

In environments where dense reward function is not available, only a small fraction of the agents' experiences will be useful to compute gradient to optimize policy, leading to substantial high sample complexity. Therefore, it is of great practical importance to develop algorithms which can learn from binary signal indicating successful task completion or other unshaped reward signal. In Section 3, we will describe a method to efficiently utilize high-reward and zero-reward trajectories to address this challenge. We will evaluate the method on program synthesis and instructions following navigation, both are particular sparse reward tasks. Figure 1 shows an example of sparse reward program synthesis. The model needs to discover the programs that can generate the correct answer in a given context and generalizes over unseen context.

Rank	Player	County
1	Nicky English	Tipperary
2	Mark Corrigan	Offaly
3	Joe Hennessy	Kerry
3	Finbarr Delaney	Cork
5	Nicky English	Tipperary
5	Adrian Ronan	Kilkenny
7	Nicky English	Tipperary

x = "Which player ranked the most?" $R(\mathbf{z}) = \mathbb{I}\{\texttt{Execute}(\mathbf{z}) == \text{"Nicky English"}\}$

Figure 1: An example of the program synthesis task, where an agent is presented with a context s_0 consists of a natural language question and a table, and is asked to generate a program $\mathbf{z} = (z_1, z_2, .., z_n)$. The agent receives a reward of 1 if execution of $\mathbf{z}$ on the relevant data table leads to the correct answer g (e.g., "Nicky English").

We consider goal-conditioned reinforcement learning from sparse rewards. This constitutes a modification to the reward function such that it depends on a goal $g \in G$, such that $r(\mathbf{z}, g, s) : S \times Z \times G \to R$. Every episode starts with sampling a state-goal pair from some distribution $p(s_0, g)$. Unlike the state, the goal stays fixed for the whole episode. At every time step, an action is chosen according to some policy π, which is expressed as a function of the state and the goal, $\pi : S \times G \to Z$. Therefore, we apply the following sparse reward function:

$$r(\mathbf{z}, g, s) = \begin{cases} 1, \text{ if } F(\mathbf{z}) = g \\ 0, \text{ otherwise} \end{cases} \tag{2}$$

where g is a goal and $F(\mathbf{z})$ denotes evaluating action sequence $\mathbf{z}$ on the task that controls when the goal is considered completed. The objective is given by

$$J(\theta) = \mathbb{E}_{s_0,g\sim p(s_0,g),\mathbf{z}\sim\mathcal{Z}}[r(\mathbf{z}, g, s_0)] = \mathbb{E}_{s_0,g\sim p(s_0,g)}\mathbb{E}_{z\sim\mathcal{Z}}[r(\mathbf{z}, g, s_0)\pi_\theta(\mathbf{z}|s_0)] \tag{3}$$

$$= \mathbb{E}_{s_0,g\sim p(s_0,g)}\mathbb{E}_{\mathbf{z}\sim\mathcal{Z}}[r(\mathbf{z}, g, s_0)\prod_{t=1}^{H}\pi(z_t \mid \mathbf{z}_{<t}, s_0)], \tag{4}$$

where H is the length of the trajectory. We can calculate the gradient of Equation 4 with REINFORCE (Williams, 1992) and estimate it using Monte Carlo samples.

$$\nabla_\theta J(\theta) = \mathbb{E}_{s_0,g\sim p(s_0,g)}\mathbb{E}_{\mathbf{z}\sim\mathcal{Z}}[\nabla_\theta \log \pi_\theta(\mathbf{z}|s_0) r(\mathbf{z}, g, s_0)], \tag{5}$$

Unfortunately, since the search space of programs is very large, most samples $\mathbf{z}$ have reward $R(\mathbf{z}) = 0$, thus have no contribution to the gradient estimation in Equation 5. Besides, because the variance of score function estimators is very high, it is challenging to estimate the gradient in Equation 5 with a small number of successful programs. Previous method Liang et al. (2018) propose to estimate gradient as a combination of expectations inside and outside successful programs buffer, however it's still restricted to use successful programs only, and suffers from high sample complexity.

3 Method

In this section, we fist introduce entropy regularized reinforcement learning and describe optimizing policy via minimizing a discrepancy between itself and a prior in Section 3.1, and then introduce learnable prior to guide policy optimization in Section 3.2, finally we introduce a class of flexible adaptive gradient estimator Section 3.3.

3.1 Entropy Regularized Reinforcement Learning.

We consider a general entropy regularized objective (Ziebart et al., 2008) which favors stochastic policies by augmenting the objective with the relative entropy of the policy,

$$J(\theta) = \mathbb{E}_{s_0,g\sim p(s_0,g)}\mathbb{E}_{\mathbf{z}\sim\mathcal{Z}}[\pi_\theta(\mathbf{z}|s_0)r(\mathbf{z},g,s_0) + \lambda H(\pi_\theta(\mathbf{z}|s_0))], \tag{6}$$

where λ is a regularization weight, $H(\pi_\theta(\mathbf{z}|s_0))$ is the entropy regularization term. Entropy based policy optimization is a general framework that has gained many successes in a variety of tasks (see e.g., Haarnoja et al., 2018; Teh et al., 2017). Maximizing Equation 6 is equivalent to minimizing the Kullback–Leibler discrepancy between policy $\pi_\theta(\mathbf{z}|s_0)$ and an energy based prior distribution.

Lemma 1. *Maximizing Equation 6 is equivalent to minimizing the following objective,*

$$L(\theta) = \mathbb{E}_{s_0,g\sim p(s_0,g)}\mathbb{E}_{\mathbf{z}\sim\mathcal{Z}}[\lambda D_{\mathrm{KL}}\left(\pi_\theta(\mathbf{z}|s_0)\|\bar{\pi}(\mathbf{z})\right), \quad \bar{\pi}(\mathbf{z})] = \exp\left(\frac{1}{\lambda}(r(\mathbf{z},g,s_0) - V(s_0))\right) \tag{7}$$

where $V(s_0) = \lambda\log\int_{\mathbf{z}\sim\mathcal{Z}}\exp(R(\mathbf{z},g,s_0)/\lambda)$ is a 'soft-version' of value function, serving as a normalization constant here. From Equation 7, we aim to approximate the distribution $\bar{\pi}(\mathbf{z})$ with a distribution from a family $\{\pi_\theta(\mathbf{z}|s_0)\colon \theta\in\Theta\}$, where θ is the parameter that we want to optimize, and $\pi_\theta(\mathbf{z}|s_0)$ is represented as an autoregressive policy in Equation 1. In environments where only sparse reward function is available, only a small fraction of the agent's samples will be useful to compute gradient to optimize policy, thus Equation 6 often leads to a substantial sample complexity. Equation 7 seems would be a better objective since all of the agent's samples can contribute to the minimization of KL-divergence, however, for a given s_0, the prior distribution is simply a binary value function over $\mathbf{z}$ which is not suitable. Intuitively, we would like $\bar{\pi}(\mathbf{z})$ weighs higher on 'almost success' action sequences $\mathbf{z}$ and weighs lower on 'far from success' action sequences $\mathbf{z}$.

3.2 Guiding Prior Distribution.

In this part, we will describe how to learn the prior distribution $\bar{\pi}(\mathbf{z})$ to guide policy optimization.

Proposition 1. *Given a policy $\pi_\theta(\mathbf{z}|s_0)$, new guiding prior distribution $\bar{\pi}(\mathbf{z})$ that minimizes the discrepancy in Equation 7 is given by,*

$$\bar{\pi}(\mathbf{z}) = \mathbb{E}_{s_0,g\sim p(s_0,g)}\left[\pi_\theta(\mathbf{z}|s_0)\right], \tag{8}$$

and the minimization of Equation 7 equals to mutual information between s_0 and $\mathbf{z}$:

$$\mathbb{E}_{s_0,g\sim p(s_0,g)}[D_{\mathrm{KL}}\left(\pi_\theta(\mathbf{z}|s_0)\ \|\ \bar{\pi}(\mathbf{z})\right)] = I(s_0;\mathbf{z}) \tag{9}$$

Proof. See Appendix C for details. □

Proposition 1 indicates that alternatively optimizing $\pi_\theta(\mathbf{z}|s_0)$ and $\bar{\pi}(\mathbf{z})$ leads to a complex mixture distribution of $\bar{\pi}(\mathbf{z})$, increasing the expressive power of prior for credit assignment. Since Equation 8 minimizes $D_{\mathrm{KL}}\left(\pi_\theta(\mathbf{z}|s_0)\ \|\ \bar{\pi}(\mathbf{z})\right)$ and leads to a mutual information between s_0 and $\mathbf{z}$, therefore the entropy regularized objective becomes the following mutual information regularized objective,

$$J(\theta) = \mathbb{E}_{s_0,g\sim p(s_0,g)}\mathbb{E}_{\mathbf{z}\sim\mathcal{Z}}\pi_\theta(\mathbf{z}|s_0)r(\mathbf{z},g,s_0) - \lambda I(s_0;\mathbf{z}), \tag{10}$$

Equation 10 draws connection with rate distortion theory (Shannon, 1959; Cover & Thomas, 2012), intuitively, the policy $\pi_\theta(\mathbf{z}|s_0)$ is encouraged to discard reward-irrelevant information in context s_0 subject to a limited channel capacity given by $I(s_0;\mathbf{z})$. In the next section, we will present a class of gradient estimator that can adaptively update policy distribution to approximate the guiding prior.

3.3 Adaptive Gradient Estimation.

While $D_{\text{KL}}(\pi_\theta(\mathbf{z}|s_0) \mid\mid \bar{\pi}(\mathbf{z}))$ is the typical divergence measure widely used in variational inference and reinforcement learning (see e.g. Wainwright et al., 2008; Abdolmaleki et al., 2018; Hoffman et al., 2013), it often leads to model collapse because of its mode-seeking property. Therefore, directly optimizing Equation 7 often gives a suboptimal model $\pi_\theta(\mathbf{z}|s_0)$. It is therefore natural to consider alternative divergence measures. We approach this problem by minimizing the general f-divergence (Ali & Silvey, 1966; Morimoto, 1963) between $\bar{\pi}(\mathbf{z})$ and $\pi_\theta(\mathbf{z}|s_0)$. f-divergence includes a large spectrum of divergences (*e.g.*, KL and reverse KL divergence) and is shown to be powerful in various settings (Nowozin et al., 2016; Wang et al., 2018; Ghasemipour et al., 2019),

$$D_{\text{F}}(\bar{\pi}(\mathbf{z}) \mid\mid \pi_\theta(\mathbf{z}|s_0)) = \mathbb{E}_{\mathbf{z}\sim\pi_\theta(\mathbf{z}|s_0)}\left[f\left(\frac{\bar{\pi}(\mathbf{z})}{\pi_\theta(\mathbf{z}|s_0)}\right) - f(1)\right], \tag{11}$$

where $f\colon \mathbb{R}_+ \to \mathbb{R}$ is any twice-differentiable convex function. It can be shown by Jensen's inequality that $D_{\text{F}}(p \mid\mid q) \geq 0$ for any p and q. Further, if $f(t)$ is strictly convex at $t = 1$, then $D_{\text{F}}(\bar{\pi}(\mathbf{z}) \mid\mid \pi_\theta(\mathbf{z}|s_0)) = 0$ implies $\bar{\pi}(\mathbf{z}) = \pi_\theta(\mathbf{z}|s_0)$. We use stochastic optimization to minimizing Equation 11, then gradient of Equation 11 is given by:

Lemma 2. *Assume f is a differentiable convex function and $\log \pi_\theta(\mathbf{z}|s_0)$ is differentiable w.r.t. θ. For f-divergence defined in equation 11, we have*

$$\nabla_\theta D_{\text{F}}(\bar{\pi}(\mathbf{z}) \mid\mid \pi_\theta(\mathbf{z}|s_0)) = -\mathbb{E}_{\mathbf{z}\sim\pi_\theta(\mathbf{z}|s_0)}\left[\rho_f\left(\frac{\pi_\theta(\mathbf{z}|s_0)}{\bar{\pi}(\mathbf{z})}\right)\nabla_\theta \log \pi_\theta(\mathbf{z}|s_0)\right], \tag{12}$$

where $\rho_f(t) = f'(t)t - f(t)$.

Proof. See Appendix B for details or Wang et al. (2018). □

Equation 12 shows that the gradient of f-divergence between $\pi_\theta(\mathbf{z}|s_0)$ and $\bar{\pi}(\mathbf{z})$ can be specified through ρ_f or f. In next section, we will describe how to adaptive choose ρ_f or f based on the discrepancy between $\pi_\theta(\mathbf{z}|s_0)$ and $\bar{\pi}(\mathbf{z})$. Since the space of Z is enumerable and the environment is deterministic, the expectation over $\mathbf{z} \sim \pi_\theta(\mathbf{z}|s_0)$ can be efficiently computed through sampling in replay buffer. We proceed to describe how to estimate this gradient with samples.

3.4 Final Algorithm.

Given Equation 12, it's natural to ask how to estimate the gradient, a naive way is simply store past trajectories in a replay buffer and sample random mini-batch from it to compute the gradient. However this approach suffers from the fact that a large fraction of sampled trajectories have zero-reward, which leads to high sample complexity. We propose to save high-reward trajectories and zero-reward trajectories into two separated replay buffers and estimate the following gradient:

Proposition 2. *Given replay buffers $\mathcal{B}$ and $\mathcal{C}$ for saving high-reward and zero-reward trajectories, an unbiased and low variance estimation is given by,*

$$\nabla_\theta \hat{D}_F(\bar{\pi}(\mathbf{z}) \mid\mid \pi_\theta(\mathbf{z}|s_0)) =$$
$$w_{\mathcal{B}}\mathbb{E}_{\mathbf{z}\sim\pi_\theta^+(\mathbf{z}|x)}\rho_f\left(\frac{\pi_\theta(\mathbf{z}|s_0)}{\bar{\pi}(\mathbf{z})}\right)\nabla_\theta \log \pi_\theta(\mathbf{z}|s_0) + w_{\mathcal{C}}\mathbb{E}_{\mathbf{z}\sim\pi_\theta^-(\mathbf{z}|x)}\rho_f\left(\frac{\pi_\theta(\mathbf{z}|s_0)}{\bar{\pi}(\mathbf{z})}\right)\nabla_\theta \log \pi_\theta(\mathbf{z}|s_0) \tag{13}$$

where $w_{\mathcal{B}}$ and $w_{\mathcal{B}}$ represent the total probability of trajectories in replay buffers $\mathcal{B}$ and $\mathcal{C}$ respectively, $w_{\mathcal{B}} + w_{\mathcal{C}} = 1$, and

$$\pi_\theta^+(\mathbf{z} \mid x) = \begin{cases} \pi_\theta(\mathbf{z}|s_0)/w_{\mathcal{B}} & \textit{if } \mathbf{z} \in \mathcal{B} \\ 0 & \textit{if } \mathbf{z} \in \mathcal{C} \end{cases}, \ \pi_\theta^-(\mathbf{z} \mid x) = \begin{cases} 0 & \textit{if } \mathbf{z} \in \mathcal{B} \\ \pi_\theta(\mathbf{z}|s_0)/w_{\mathcal{C}} & \textit{if } \mathbf{z} \in \mathcal{C} \end{cases} \tag{14}$$

Proof. See Appendix D for details. □

The gradient estimation uses high-reward trajectories thus $\pi_\theta(\mathbf{z}|s_0)$ will not forget them, the estimation also utilize zero-reward trajectories in the past, which improves sample efficiency. The corresponding framework is shown in Figure 2. Note that different from MAPO where they also use a buffer to save successful programs, Equation 13 differs in that all off-policy samples can be used to estimate gradient, which leads to a higher sample efficiency.

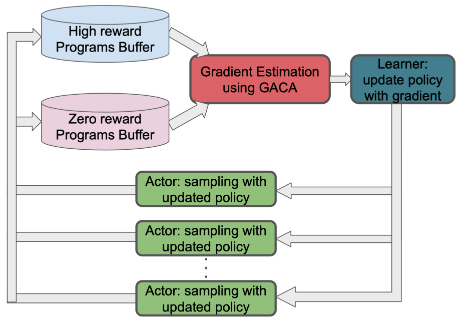
High reward Programs Buffer
Zero reward Programs Buffer
Gradient Estimation using GACA
Learner: update policy with gradient
Actor: sampling with updated policy
Actor: sampling with updated policy
Actor: sampling with updated policy

Figure 2: Overview of GACA: it consists of multiple actors for sampling and storing high reward episodes into buffer $\mathcal{B}$ and zero reward episodes into buffer $\mathcal{C}$, gradient is estimated at central learner periodically using samples from both $\mathcal{B}$, $\mathcal{C}$ based on Equation 13.

We follow Wang et al. (2018) in choosing f-divergence such that it achieve a trade-off between exploration and exploitation, specifically, let $\{\mathbf{z}_i\}$ be drawn from buffers $\mathcal{B}$ and $\mathcal{C}$ and $w_i = \pi(\mathbf{z}_i|s_0)/\bar{\pi}(\mathbf{z}_i)$, then we substitute $\rho_f(\pi_\theta(\mathbf{z}|s_0)/\bar{\pi}(\mathbf{z}))$ with the inverse of approximate tail probability given by $\frac{1}{n}\mathbb{E}_{i=1}^{n}\mathbb{I}(w_i \geq t)$. The benefit of doing this is policy distribution $\pi_\theta(\mathbf{z}|s_0)$ can adaptively coverage and approximate prior distribution $\bar{\pi}(\mathbf{z})$.

In practice, to overcome cold start problem in sparse reward policy optimization, we follow Liang et al. (2018) to clip $w_{\mathcal{B}}$ to a given range such that $w_{\mathcal{B}} = \max(w_{\mathcal{B}}, w_l)$ and $w_{\mathcal{B}} = \min(w_{\mathcal{B}}, w_u)$ where $w_l \leq w_u$. Note that Equation 13 generalizes previous work in credit assignment including REINFORCE, MML (Dempster et al., 1977; Berant et al., 2013; Guu et al., 2017), MAPO (Liang et al., 2018), RAML (Norouzi et al., 2016), and IML (Liang et al., 2017; Abolafia et al., 2018). It is natural to expect this more flexible gradient estimator provide an adaptive trade-off between different credit assignment methods and can yield powerful credit assignment. Due to page limit, we leave discussions and proofs around generalization in Appendix E. Combining Proportion 1 and Proportion 2 together, we summarize the main algorithm in Algorithm 1.

4 Experiment

We first introduce the set up of experiments, then evaluate GACA on two sparse reward program synthesis benchmarks WikiTableQuestions and WikiSQL, and an instruction following sparse reward navigation task.

4.1 Experimental setup

WikiTableQuestions (Pasupat & Liang, 2015) contains 2,108 tables and 18,496 question-answer pairs build from tables extracted from Wikipedia. WikiSQL (Zhong et al., 2017) is a recent

Algorithm 1 Guided Adaptive Credit Assignment for Sample Efficient Policy Optimization

```
Require: Training data $p(s_0, g)$, random initialized policy $\pi_\theta(\mathbf{z}|s_0)$, uniform initialized prior $\bar{\pi}(\mathbf{z})$, high-reward and zero-reward trajectory buffers $\mathcal{B}$ and $\mathcal{C}$, and clipping thresholds $w_l$ and $w_u$.
  repeat
    Sample initial states and goals $\{s_0, g\}$ from data distribution $p(s_0, g)$
    Collect trajectories with $\pi_\theta(\mathbf{z}|s_0)$ given $\{s_0, g\}$ and push trajectories into replay buffers $\mathcal{B}$ and $\mathcal{C}$ according to their rewards.
    Draw $\{\mathbf{z}_i\}$ from buffers $\mathcal{B}$ and $\mathcal{C}$ through stratified sampling, compute $w_\mathcal{B}$ and $w_\mathcal{C}$
    Compute tail probability $\frac{1}{n}\mathbb{E}_{i=1}^{n}\mathbb{I}(w_i \geq t)$, where $w_i = \pi(\mathbf{z}_i \mid x)/\bar{\pi}(\mathbf{z}_i)$
    Update policy distribution $\pi_\theta(\mathbf{z}|s_0)$ with Equation 13 by substituting $\rho_f(\pi_\theta(\mathbf{z}|s_0)/\bar{\pi}(\mathbf{z}))$ with the inverse of tail probability
    Compute new guiding prior distribution $\bar{\pi}(\mathbf{z}) = \mathbb{E}_{s_0, g \sim p(s_0, g)}\left[\pi_\theta(\mathbf{z}|s_0)\right]$
  until converge or early stop
```

large scale dataset on learning natural language interfaces for databases. It contains 24,241 tables extracted from Wikipedia and 80,654 question-program pairs. It is annotated with programs (SQL). In both datasets, question-answers are split into train, validation, and test sets. Examples of generated programs are shown in Figure 5. The policy needs to generate correct programs from binary feedback, making these two benchmarks very challenging. We also evaluate GACA on an instruction following navigation environment TextWorld (Agarwal et al., 2019). The task is an instruction following navigation in a maze of size $N \times N$ with K deadly traps distributed randomly over the maze. An agent is given need a language instruction which outlines an optimal path that the agent can take to reach the goal, the agent needs to generate a sequence of actions and the agent receives a reward of 1 if it succeeds in reaching the goal within a certain number of steps, otherwise 0. An example of this task is shown in Figure 5. For details in experiments, refer to Appendix F.

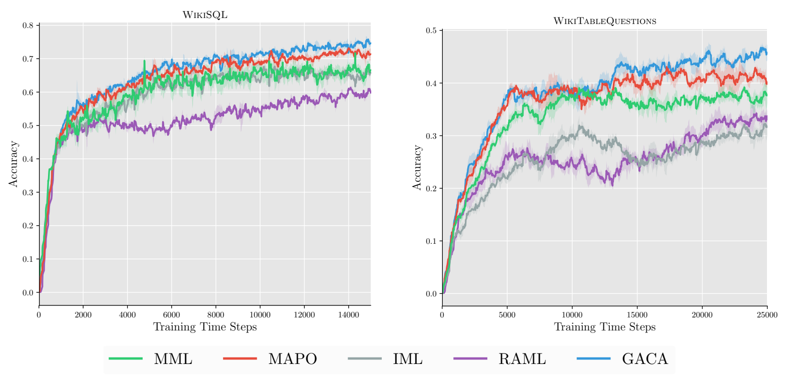

Figure 3: Comparing GACA and baselines on benchmarks. The plot is average of 5 runs with a bar of one standard deviation.

4.2 Comparing GACA with baselines

Firstly, we compare GACA with several baseline methods that are special cases of GACA, to show the effectiveness of guiding prior and adaptive gradient estimation. We briefly introduce each baseline method here and leave the detailed discussion and proof of generalization in Appendix E.
REINFORCE: REINFORCE maximizes expected return, we use on-policy samples to estimate the gradient $\nabla_\theta J_{\text{RL}} = \mathbb{E}_{s_0, g \sim p(s_0, g)}\mathbb{E}_{\mathbf{z} \sim \pi_\theta(\mathbf{z}|s_0)},[\nabla_\theta \log \pi_\theta(\mathbf{z}|s_0) r(\mathbf{z}, s_0, g)]$.
IML: Iterative maximize likelihood (Liang et al., 2017; Abolafia et al., 2018) uniformly maximizes the likelihood of all the high-reward trajectories in past experience, the gradient is given by

$\nabla_\theta J_{\text{IML}} = \mathbb{E}_{s_0,g\sim p(s_0,g)} \sum_{\mathbf{z}\in\mathcal{B}} \nabla_\theta \log \pi_\theta(\mathbf{z}|s_0) r(\mathbf{z}, s_0, g)$.

RAML: Reward Augmented Maximum Likelihood (Norouzi et al., 2016) is a more general variant of IML, which weights off-policy samples with an energy based prior distribution in Equation 7, $J_{\text{RAML}} = \mathbb{E}_{s_0,g\sim p(s_0,g)} \mathbb{E}_{\mathbf{z}\sim \mathrm{Z}} \bar{\pi}(\mathbf{z}) \log \pi_\theta(\mathbf{z}|s_0) r(\mathbf{z}, s_0, g)$, where $\bar{\pi}(\mathbf{z}) = \exp\left(\frac{1}{\lambda}(r(\mathbf{z}, s_0, g) - V(x))\right)$.

MML: Maximize Marginal Likelihood (Dempster et al., 1977; Berant et al., 2013) maximizes the marginal probability of the replay buffer $\mathcal{B}$. The gradient of J_{MML} is given by $\nabla_\theta J_{\text{MML}} = \mathbb{E}_{s_0,g\sim p(s_0,g)} \sum_{\mathbf{z}\in\mathcal{B}} \frac{\pi_\theta(\mathbf{z}|s_0)}{\sum_{\hat{\mathbf{z}}\in\mathcal{B}} \pi_\theta(\hat{\mathbf{z}}|s_0)} r(\mathbf{z}, s_0, g) \nabla_\theta \log \pi_\theta(\mathbf{z}|s_0)$.

MAPO/MAPOX: Memory Augmented Policy Optimization (Liang et al., 2018) is a recent method for reusing high-reward trajectories, it maximizes the expected reward and estimate the gradient with off-policy high-reward trajectories: $\nabla_\theta J_{\text{MAPO}} = (1 - \alpha)\mathbb{E}_{s_0,g\sim p(s_0,g)} \mathbb{E}_{\mathbf{z}\sim\pi(\mathbf{z}|x)}[\nabla_\theta \log \pi_\theta(\mathbf{z}|s_0) r(\mathbf{z}, s_0, g)] + \alpha \mathbb{E}_{\mathbf{z}\sim\mathcal{B}}[\nabla_\theta \log \pi_\theta(\mathbf{z}|s_0) r(\mathbf{z}, s_0, g)]$, where α is a weight equals to the total probability of high-reward trajectory $\mathbf{z}$ in buffer $\mathcal{B}$. MAPOX (Agarwal et al., 2019) improves MAPO by running MAPO on data collected by IML.

Method \ Dataset	WikiTableQuestions Val	WikiTableQuestions Test	WikiTableQuestions Improvement	WikiSQL Val	WikiSQL Test	WikiSQL Improvement
REINFORCE	< 10	< 10		< 10	< 10	
IML	35.4 ± 0.6	36.8 ± 0.4	$+7.8$	69.3 ± 0.5	70.1 ± 0.2	$+6.2$
RAML	35.4 ± 0.7	35.9 ± 0.6	$+8.7$	57.5 ± 0.3	61.4 ± 0.3	$+14.9$
MML	37.8 ± 0.7	39.7 ± 0.3	$+4.9$	68.7 ± 0.2	70.7 ± 0.1	$+5.6$
MAPO	42.3 ± 0.1	42.8 ± 0.3	$+1.8$	71.9 ± 0.6	72.5 ± 0.1	$+3.8$
MAPOX	42.5 ± 0.4	43.6 ± 0.4	$+1.1$	74.4 ± 0.5	74.9 ± 0.3	$+1.4$
GACA w/o GP	43.8 ± 0.4	44.2 ± 0.2		74.6 ± 0.3	75.3 ± 0.2	
GACA w/o AG	44.1 ± 0.5	44.1 ± 0.3		74.9 ± 0.3	75.2 ± 0.1	
GACA	45.1 ± 0.3	$\mathbf{44.6} \pm 0.2$		75.7 ± 0.3	$\mathbf{76.3} \pm 0.1$	

Table 1: Comparison to various credit assignment methods on WikiTableQuestions and WikiSQL benchmarks. We report mean accuracy, and its standard deviation on test sets based on 5 runs.

Method	Val.	Test
Oracle	95.7(±1.3)	92.6(±1.0)
MAPO	73.1(±2.1)	68.5(±2.6)
MeRL	75.3(±1.6)	72.3(±2.2)
BoRL	83.0(±3.6)	74.5(±2.5)
GACA	87.3(±4.1)	80.1(±2.8)

Figure 4: Evaluation of GACA on TextWorld with state-of-the-art.

We first compare GACA with its various special cases on sparse reward program synthesis benchmarks, results are shown in Figure 3 and Table 1, the comparison shows that noticeably improves upon previous methods in terms of both sample efficiency and asymptotic performance significantly by performing better credit assignment. The ablation study shows that both adaptive gradient estimation(AG) and guiding prior(GP) improve performance. The improvement over MAPO demonstrates that reusing all off-policy trajectories can greatly improve policy, the improvement over IML and MAPOX shows that encouraging exploration with reverse KL-divergence is not enough because it's simply impossible to explore such a large state space, while guiding prior and adaptive gradient estimation provide an efficient way of exploration and exploitation. We also analyzed a trained model qualitatively on program synthesis tasks and see that it can generate fairly complex programs, see Appendix G for some examples of generated programs. Our experiments follow the settings of MAPO (Liang et al., 2018) and MeRL (Agarwal et al., 2019), refer to Appendix F for more details in experiments.

4.3 Comparing GACA with state-of-the-art

We present the results on sparse reward program synthesis in Table 3 and Table 2. The results of TextWorld are shown in Table 4. GACA outperforms most recent state-of-the-art methods BoRL and MeRL proposed in Agarwal et al. (2019) by a large margin. The results demonstrate the efficacy of the proposed credit assignment compared to previous SOTA credit assignment methods. We would like to point out that GACA is a general method and can be combined with these techniques to further boost performance.

Method	E.S.	Val.	Test	
MAPO	1	42.3	42.8	+5.8
MeRL	1	44.1	43.2	+5.4
BoRL	1	42.9	43.8	+4.8
MAPO (ensemble)	10	-	46.3	+2.3
MeRL (ensemble)	10	-	46.9	+1.7
GACA	1	45.1	**44.6**	
GACA (ensemble)	10	-	**48.6**	

Table 2: Evaluation of GACA with state-of-the-art on WikiTableQuestions.

Method	E.S.	Val.	Test	
MAPO	1	71.9	72.5	+6.3
MeRL	1	74.9	74.8	+4.0
BoRL	1	74.6	74.2	+4.6
MAPO (ensemble)	10	-	74.9	+3.9
MeRL (ensemble)	10	-	77.1	+1.7
GACA	1	75.7	**76.3**	
GACA (ensemble)	10	-	**78.8**	

Table 3: Evaluation of GACA with state-of-the-art on WikiSQL.

5 Related Work

Credit assignment is a critical part of various sequential decision making methods. Guu et al. (2017) builds connection between REINFORCE and MML by proposing hybrid approaches to take advantages of both MML and REINFORCE. Entropy based policy optimization is widely used in reinforcement learning (Ziebart et al., 2008; Schulman et al., 2017), recently entropy based off-policy policy optimization is also proposed to approximate optimal policy distribution by minimizing the Kullback–Leibler(KL) divergence between policy and optimal distribution (Haarnoja et al., 2018), Norouzi et al. (2016) considers an alternative direction of the KL divergence, where samples from exponential payoff distribution are used to estimate gradient. Recent work Grau-Moya et al. (2019) also propose to learn the prior distribution for Q-learning and show that this leads to a mutual information regularization. Experience replay is widely used in sparse reward reinforcement learning in order to exploit past high reward trajectories (Gangwani et al., 2019; Liang et al., 2018; Oh et al., 2018; Abolafia et al., 2018). Andrychowicz et al. (2017) proposes to re-label visited states as goal states during training. More recent progress includes meta-learning the reward(such as discount factor) (Xu et al., 2018). Weber et al. 2019 provides a comprehensive review of credit assignment methods in stochastic computation graph. Recently, there are a surge of interest in applying policy optimization in program synthesis through sparse supervision (Krishnamurthy et al., 2017; Guu et al., 2017; Liang et al., 2017; 2018; Agarwal et al., 2019). GACA differs from previous methods by enabling reusing off-policy samples through learned prior and generalized gradient estimation.

6 Conclusion

We developed the Guided Adaptive Credit Assignment(GACA), a new and general credit assignment method for obtaining sample efficiency of policy optimization in sparse reward setting. Our method generalizes several previous approaches. We demonstrated its practical advantages over existing methods, including MML, IML, REINFORCE, etc, on several challenging sparse reward tasks. In the future, we will investigate how to extend GACA to stochastic environments and apply it to robot learning from binary reward feedback. We would also like to point out that our method can be useful in other challenging tasks with deterministic environments such as combinational optimization and structural prediction where credit assignment from binary feedback remains a major challenge.

References

Abbas Abdolmaleki, Jost Tobias Springenberg, Yuval Tassa, Remi Munos, Nicolas Heess, and Martin Riedmiller. Maximum a posteriori policy optimisation. In *International Conference on Learning Representations*, 2018.

Daniel A Abolafia, Mohammad Norouzi, Jonathan Shen, Rui Zhao, and Quoc V Le. Neural program synthesis with priority queue training. *arXiv preprint arXiv:1801.03526*, 2018.

Rishabh Agarwal, Chen Liang, Dale Schuurmans, and Mohammad Norouzi. Learning to generalize from sparse and underspecified rewards. *ICML*, 2019.

Syed Mumtaz Ali and Samuel D Silvey. A general class of coefficients of divergence of one distribution from another. *Journal of the Royal Statistical Society: Series B (Methodological)*, 28(1):131–142, 1966.

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In *Advances in Neural Information Processing Systems*, pp. 5048–5058, 2017.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pp. 1533–1544, 2013.

Rudy Bunel, Matthew Hausknecht, Jacob Devlin, Rishabh Singh, and Pushmeet Kohli. Leveraging grammar and reinforcement learning for neural program synthesis. *arXiv preprint arXiv:1805.04276*, 2018.

Thomas M Cover and Joy A Thomas. *Elements of information theory.* John Wiley-Sons, 2012.

Imre Csiszár, Paul C Shields, et al. Information theory and statistics: A tutorial. *Foundations and Trends® in Communications and Information Theory*, 1(4):417–528, 2004.

Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1): 1–22, 1977.

Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Volodymyr Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, et al. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. In *International Conference on Machine Learning*, pp. 1406–1415, 2018.

Tanmay Gangwani, Qiang Liu, and Jian Peng. Learning self-imitating diverse policies. In *International Conference on Learning Representations*, 2019.

Seyed Kamyar Seyed Ghasemipour, Richard Zemel, and Shixiang Gu. A divergence minimization perspective on imitation learning methods. *arXiv preprint arXiv:1911.02256*, 2019.

Jordi Grau-Moya, Felix Leibfried, and Peter Vrancx. Soft q-learning with mutual-information regularization. In *International Conference on Learning Representations*, 2019.

Kelvin Guu, Panupong Pasupat, Evan Liu, and Percy Liang. From language to programs: Bridging reinforcement learning and maximum marginal likelihood. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1051–1062, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1097.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *ICML*. PMLR, 2018.

Matthew D Hoffman, David M Blei, Chong Wang, and John Paisley. Stochastic variational inference. *The Journal of Machine Learning Research*, 14(1):1303–1347, 2013.

Sham M Kakade. A natural policy gradient. In *Advances in Neural Information Processing Systems*, pp. 1531–1538, 2002.

Diederick P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

Jayant Krishnamurthy, Pradeep Dasigi, and Matt Gardner. Neural semantic parsing with type constraints for semi-structured tables. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 1516–1526, 2017.

Chen Liang, Jonathan Berant, Quoc Le, Kenneth D. Forbus, and Ni Lao. Neural symbolic machines: Learning semantic parsers on freebase with weak supervision. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 23–33. Association for Computational Linguistics, 2017. doi: 10.18653/v1/P17-1003.

Chen Liang, Mohammad Norouzi, Jonathan Berant, Quoc Le, and Ni Lao. Memory augmented policy optimization for program synthesis with generalization. *Advances in Neural Information Processing Systems(NeurIPS)*, 2018.

Friedrich Liese and Igor Vajda. On divergences and informations in statistics and information theory. *IEEE Transactions on Information Theory*, 52(10):4394–4412, 2006.

Andriy Mnih and Koray Kavukcuoglu. Learning word embeddings efficiently with noise-contrastive estimation. In *Advances in neural information processing systems*, pp. 2265–2273, 2013.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pp. 1928–1937, 2016.

Tetsuzo Morimoto. Markov processes and the h-theorem. *Journal of the Physical Society of Japan*, 18(3):328–331, 1963.

Mohammad Norouzi, Samy Bengio, Navdeep Jaitly, Mike Schuster, Yonghui Wu, Dale Schuurmans, et al. Reward augmented maximum likelihood for neural structured prediction. In *Advances In Neural Information Processing Systems*, pp. 1723–1731, 2016.

Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. f-gan: Training generative neural samplers using variational divergence minimization. In *Advances in neural information processing systems*, pp. 271–279, 2016.

Junhyuk Oh, Yijie Guo, Satinder Singh, and Honglak Lee. Self-imitation learning. In *ICML*, 2018.

Panupong Pasupat and Percy Liang. Compositional semantic parsing on semi-structured tables. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 1470–1480. Association for Computational Linguistics, 2015. doi: 10.3115/v1/P15-1142.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. In *NIPS Autodiff Workshop*, 2017.

Vitchyr H Pong, Murtaza Dalal, Steven Lin, Ashvin Nair, Shikhar Bahl, and Sergey Levine. Skew-fit: State-covering self-supervised reinforcement learning. *arXiv preprint arXiv:1903.03698*, 2019.

John Schulman, Sergey Levine, Philipp Moritz, Michael I. Jordan, and Pieter Abbeel. Trust region policy optimization. In *International Conference on Machine Learning*, 2015.

John Schulman, Xi Chen, and Pieter Abbeel. Equivalence between policy gradients and soft q-learning. *arXiv preprint arXiv:1704.06440*, 2017.

Claude E Shannon. Coding theorems for a discrete source with a fidelity criterion. *IRE Nat. Conv. Rec*, 4(142-163):1, 1959.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354, 2017.

Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pp. 1057–1063, 2000.

Yee Teh, Victor Bapst, Wojciech M Czarnecki, John Quan, James Kirkpatrick, Raia Hadsell, Nicolas Heess, and Razvan Pascanu. Distral: Robust multitask reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 4496–4506, 2017.

Martin J Wainwright, Michael I Jordan, et al. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1–2):1–305, 2008.

Dilin Wang, Hao Liu, and Qiang Liu. Variational inference with tail-adaptive f-divergence. In *Advances in Neural Information Processing Systems*, pp. 5742–5752, 2018.

Théophane Weber, Nicolas Heess, Lars Buesing, and David Silver. Credit assignment techniques in stochastic computation graphs. *arXiv preprint arXiv:1901.01761*, 2019.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

Zhongwen Xu, Hado P van Hasselt, and David Silver. Meta-gradient reinforcement learning. In *Advances in Neural Information Processing Systems 31*, pp. 2396–2407. 2018.

Victor Zhong, Caiming Xiong, and Richard Socher. Seq2sql: Generating structured queries from natural language using reinforcement learning. *CoRR*, abs/1709.00103, 2017.

Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. Maximum entropy inverse reinforcement learning. In *Aaai*, volume 8, pp. 1433–1438. Chicago, IL, USA, 2008.

A Proof of Lemma 1

Proof. To derive Lemma 1, consider the KL divergence between $\pi_\theta(\mathbf{z}|s_0)$ and $\bar{\pi}(\mathbf{z}) = \exp\left(\frac{1}{\lambda}(r(\mathbf{z}, g, s_0) - V(s_0))\right)$, where $V(s_0) = \lambda \log \int_{\mathbf{z}\sim\mathcal{Z}} \exp(r(\mathbf{z}, g, s_0)/\lambda)$ is a 'soft-version' of value function, serving as a normalization constant here.

$$\begin{aligned}
&D_{\mathrm{KL}}\left(\pi_\theta(\mathbf{z}|s_0) \parallel \bar{\pi}(\mathbf{z})\right) \\
&= \mathbb{E}_{\mathbf{z}\sim\pi_\theta(\mathbf{z}|s_0)}\left[\log \pi_\theta(\mathbf{z}|s_0) - \log \bar{\pi}(\mathbf{z})\right] \\
&= \mathbb{E}_{\mathbf{z}\sim\pi_\theta(\mathbf{z}|s_0)}\Big[\log \pi_\theta(\mathbf{z}|s_0) - r(\mathbf{z}, g, s_0)/\lambda + \log V(s_0)\Big] \\
&= \mathbb{E}_{\mathbf{z}\sim\pi_\theta(\mathbf{z}|s_0)}\left[\log \pi_\theta(\mathbf{z}|s_0) - r(\mathbf{z}, g, s_0)/\lambda\right] + \log V(s_0),
\end{aligned}$$

Rearranging,

$$\begin{aligned}
&\mathbb{E}_{\mathbf{z}\sim\pi_\theta(\mathbf{z}|s_0)}\left[r(\mathbf{z}, g, s_0)\right] + \lambda H(\pi_\theta(\mathbf{z}|s_0)) \\
&= -\lambda D_{\mathrm{KL}}\left(\pi_\theta(\mathbf{z}|s_0) \parallel \bar{\pi}(\mathbf{z})\right) + \lambda \log V(s_0),
\end{aligned}$$

thus maximizing left hand side $\mathbb{E}_{\mathbf{z}\sim\pi_\theta(\mathbf{z}|s_0)}\left[r(\mathbf{z}, g, s_0)\right] + \lambda H(\pi_\theta(\mathbf{z}|s_0))$ is equivalent to minimizing $D_{\mathrm{KL}}\left(\pi_\theta(\mathbf{z}|s_0) \parallel \bar{\pi}(\mathbf{z})\right)$. $\square$

B Proof of Lemma 2

Proof. To derive Lemma 2, consider that $\nabla_\theta \pi_\theta(\mathbf{z}|s_0) = \pi_\theta(\mathbf{z}|s_0)\nabla_\theta \log \pi_\theta(\mathbf{z}|s_0)$, then we have

$$\begin{aligned}
&\nabla_\theta D_f(\bar{\pi}(\mathbf{z}) \parallel \pi_\theta(\mathbf{z}|s_0)) \\
&= \mathbb{E}_{\pi_\theta(\mathbf{z}|s_0)}\left[\nabla_\theta f\left(\frac{\bar{\pi}(\mathbf{z})}{\pi_\theta(\mathbf{z}|s_0)}\right) + f\left(\frac{\bar{\pi}(\mathbf{z})}{\pi_\theta(\mathbf{z}|s_0)}\right)\nabla_\theta \log \pi_\theta(\mathbf{z}|s_0)\right] \\
&= \mathbb{E}_{\pi_\theta(\mathbf{z}|s_0)}\left[f'\left(\frac{\bar{\pi}(\mathbf{z})}{\pi_\theta(\mathbf{z}|s_0)}\right)\nabla_\theta\left(\frac{\bar{\pi}(\mathbf{z})}{\pi_\theta(\mathbf{z}|s_0)}\right) + f\left(\frac{\bar{\pi}(\mathbf{z})}{\pi_\theta(\mathbf{z}|s_0)}\right)\nabla_\theta \log \pi_\theta(\mathbf{z}|s_0)\right] \\
&= \mathbb{E}_{\pi_\theta(\mathbf{z}|s_0)}\left[-f'\left(\frac{\bar{\pi}(\mathbf{z})}{\pi_\theta(\mathbf{z}|s_0)}\right)\left(\frac{\bar{\pi}(\mathbf{z})}{\pi_\theta(\mathbf{z}|s_0)}\right)\nabla_\theta \log \pi_\theta(\mathbf{z}|s_0) + f\left(\frac{\bar{\pi}(\mathbf{z})}{\pi_\theta(\mathbf{z}|s_0)}\right)\nabla_\theta \log \pi_\theta(\mathbf{z}|s_0)\right] \\
&= -\mathbb{E}_{\pi_\theta(\mathbf{z}|s_0)}\left[\rho_f\left(\frac{\bar{\pi}(\mathbf{z})}{\pi_\theta(\mathbf{z}|s_0)}\right)\log \pi_\theta(\mathbf{z}|s_0)\right],
\end{aligned}$$

where $\rho_f(t) = f'(t)t - f(t)$.

For convex function f, we have $f''(t) \geq 0$, which implies $\rho'_f(t) = f''(t)t \geq 0$ on $t \in \mathbb{R}_+$, thus ρ_f is a monotonically increasing function on $\mathbb{R}_+$. If ρ_t is strictly increasing at $t = 1$, we have f is strictly convex at $t = 1$, which guarantees $D_{\mathrm{F}}(p \parallel q) = 0$ imply $p = q$. $\square$

C Proof of Proposition 1

Let $p(s_0)$ and $p(\mathbf{z})$ denote the distribution of s_0, and $\mathbf{z}$ respectively, for notation simplicity, we omit g in the following derivation and simply use $s_0 \in p(s_0)$ to represent $s_0, g \in p(s_0, g)$, and denote

$\bar{\pi}(\mathbf{z}) = \mathbb{E}_{s_0 \sim p(s_0)} [\pi_\theta(\mathbf{z}|s_0)]$, then we have

$$\begin{aligned}
& D_{\mathrm{KL}}(p(s_0)\pi_\theta(\mathbf{z}|s_0) \mid\mid p(s_0)p(\mathbf{z})) - D_{\mathrm{KL}}(p(s_0)\pi_\theta(\mathbf{z}|s_0) \mid\mid p(s_0)\bar{\pi}(\mathbf{z})) \\
&= \mathbb{E}_{s_0 \sim p(s_0)} \mathbb{E}_{\mathbf{z} \sim \mathcal{Z}} [p(s_0)\pi_\theta(\mathbf{z}|s_0) \log \frac{p(s_0)\pi_\theta(\mathbf{z}|s_0)}{p(s_0)p(\mathbf{z})}] \\
&\quad - \mathbb{E}_{s_0 \sim p(s_0)} \mathbb{E}_{\mathbf{z} \sim \mathcal{Z}} [p(s_0)\pi_\theta(\mathbf{z}|s_0) \log \frac{p(s_0)p(\mathbf{z}|s_0)}{p(s_0)\bar{\pi}(\mathbf{z})}] \\
&= \mathbb{E}_{s_0 \sim p(s_0)} \mathbb{E}_{\mathbf{z} \sim \mathcal{Z}} [p(s_0)\pi_\theta(\mathbf{z}|s_0) \log \frac{\bar{\pi}(\mathbf{z})}{p(\mathbf{z})}] \\
&= \mathbb{E}_{s_0 \sim p(s_0)} \mathbb{E}_{\mathbf{z} \sim \mathcal{Z}} [\bar{\pi}(\mathbf{z}) \log \frac{\bar{\pi}(\mathbf{z})}{p(\mathbf{z})}] \\
&= \mathbb{E}_{s_0 \sim p(s_0)} [D_{\mathrm{KL}} (\bar{\pi}(\mathbf{z}) \mid\mid p(\mathbf{z}))] \\
&\geq 0,
\end{aligned}$$

thus $\bar{\pi}(\mathbf{z}) = \mathbb{E}_{s_0 \sim p(s_0)} [\pi_\theta(\mathbf{z}|s_0)] = \arg\min_{p(\mathbf{z})} D_{\mathrm{KL}}(p(s_0)\pi_\theta(\mathbf{z}|s_0) \mid\mid p(s_0)p(\mathbf{z}))$. Substituting $\bar{\pi}(\mathbf{z})$ in $\mathbb{E}_{s_0 \sim p(s_0)} [D_{\mathrm{KL}}(p(s_0)\pi_\theta(\mathbf{z}|s_0) \mid\mid p(s_0)p(\mathbf{z}))]$, we have

$$\begin{aligned}
& D_{\mathrm{KL}}(p(s_0)\pi_\theta(\mathbf{z}|s_0) \mid\mid p(s_0)\bar{\pi}(\mathbf{z})) \\
&= \mathbb{E}_{s_0 \sim p(s_0)} [p(s_0, \mathbf{z}) \log \frac{p(s_0, \mathbf{z})}{p(s_0)p(\mathbf{z})}] \\
&= I(s_0; \mathbf{z})
\end{aligned}$$

Thus $\mathbb{E}_{s_0 \sim p(s_0)} [\pi_\theta(\mathbf{z}|s_0)]$ is the solution of the minimization objective, and $D_{\mathrm{KL}}(p(s_0)\pi_\theta(\mathbf{z}|s_0) \mid\mid p(s_0)\bar{\pi}(\mathbf{z}))$ equals mutual information between state and action.

D Proof of Proposition 2

Proof. To prove Equation 13 is an unbiased estimation of Equation 12, note that we can either enumerate replay buffers $\mathcal{B}$ and $\mathcal{C}$ when the size of buffers are small or approximate sampling from both buffers according to the specified ratio. In any case, this gives us a stratified sampling estimator of Equation 12, which is unbiased and low variance. □

E Proof of generalization of previous credit assignment methods

In this section, we discuss the connection between GACA and each credit assignment method, we will show that GACA is a unified form of existing credit assignment method. Firstly, we summarize existing method in Table 4. Then we describe each method and give a proof of how to reduce GACA to it.

E.1 REINFORCE:

REINFORCE maximizes the expected reward and estimate the gradient with on-policy samples $J_{\mathrm{RL}} = \mathbb{E}_{s_0, g \sim p(s_0, g)} \mathbb{E}_{\mathbf{z} \sim \pi_\theta(\mathbf{z}|s_0)} r(\mathbf{z}, s_0, g)$, the gradient of REINFORCE objective is given by $\nabla_\theta J_{\mathrm{RL}} = \mathbb{E}_{s_0, g \sim p(s_0, g)} \mathbb{E}_{\mathbf{z} \sim \pi_\theta(\mathbf{z}|s_0)} \nabla_\theta \log \pi_\theta(\mathbf{z}|s_0) r(\mathbf{z}, s_0, g)$. Apart from high variance issue in REINFORCE, it also suffers from sparse reward because reward $r(\mathbf{z}, s_0, g)$ is low for most trajectories $\mathbf{z}$. In contrast, GACA utilizes off-policy samples and still maintain unbiased gradient estimate. GACA reduces to REINFORCE by simply choosing ρ_f as constant 1.

Method	Optimization gradient
REINFORCE	$\nabla_\theta J_{\mathrm{RL}} = \mathbb{E}_{s_0,g\sim p(s_0,g)} \mathbb{E}_{\mathbf{z}\sim\pi_\theta(\mathbf{z}\mid s_0)} [\nabla_\theta \log \pi_\theta(\mathbf{z}\mid s_0) r(\mathbf{z}, s_0, g)]$
MML	$\nabla_\theta J_{\mathrm{MML}} = \mathbb{E}_{s_0,g\sim p(s_0,g)} \sum_{\mathbf{z}\in\mathcal{B}} \frac{\pi_\theta(\mathbf{z}\mid s_0)}{\sum_{\hat{\mathbf{z}}\in\mathcal{B}} \pi_\theta(\hat{\mathbf{z}}\mid s_0)} r(\mathbf{z}, s_0, g) \nabla_\theta \log \pi_\theta(\mathbf{z}\mid s_0)$
IML	$\nabla_\theta J_{\mathrm{IML}} = \mathbb{E}_{s_0,g\sim p(s_0,g)} \sum_{\mathbf{z}\in\mathcal{B}} \nabla_\theta \log \pi_\theta(\mathbf{z}\mid s_0) r(\mathbf{z}, s_0, g)$
MAPO, MAPOX	$\nabla_\theta J_{\mathrm{MAPO}} = \mathbb{E}_{s_0,g\sim p(s_0,g)} [(1-\alpha)\mathbb{E}_{\mathbf{z}\sim\pi_\theta(\mathbf{z}\mid s_0)} [\nabla_\theta \log \pi_\theta(\mathbf{z}\mid s_0) r(\mathbf{z}, s_0, g)]$ $+ \alpha \sum_{\mathbf{z}\sim\mathcal{B}} [\nabla_\theta \log \pi_\theta(\mathbf{z}\mid s_0) r(\mathbf{z}, s_0, g)]], \alpha \in [0,1]$
RAML	$\nabla_\theta J_{\mathrm{RAML}} = \mathbb{E}_{s_0,g\sim p(s_0,g)} \mathbb{E}_{\mathbf{z}\sim \mathrm{Z}} [\log \pi_\theta(\mathbf{z}\mid s_0) r(\mathbf{z}, s_0, g) \bar{\pi}(\mathbf{z})]$
GACA	$\nabla_\theta \hat{D}_F(\bar{\pi}(\mathbf{z}) \mid\mid \pi_\theta(\mathbf{z}\mid s_0)) = w_{\mathcal{B}} \mathbb{E}_{\mathbf{z}\sim\pi_\theta^+(\mathbf{z}\mid x)} \rho_f \left(\pi_\theta(\mathbf{z}\mid s_0)/\bar{\pi}(\mathbf{z})\right) \nabla_\theta \log \pi_\theta(\mathbf{z}\mid s_0)$ $+ w_{\mathcal{C}} \mathbb{E}_{\mathbf{z}\sim\pi_\theta^-(\mathbf{z}\mid x)} \rho_f \left(\pi_\theta(\mathbf{z}\mid s_0)/\bar{\pi}(\mathbf{z})\right) \nabla_\theta \log \pi_\theta(\mathbf{z}\mid s_0)$

Table 4: The optimization gradient for various credit assignment algorithms. GACA generalizes existing methods by f-divergence based adaptive gradient estimation.

E.2 MML:

Maximize Marginal Likelihood(MML) (Dempster et al., 1977; Berant et al., 2013) maximizes the marginal probability of the replay buffer $\mathcal{B}$, the objective of MML is given by $J_{\mathrm{MML}} = \mathbb{E}_{s_0,g\sim p(s_0,g)} \log \sum_{\mathbf{z}\in\mathcal{B}} \pi_\theta(\mathbf{z}\mid s_0) r(\mathbf{z}, s_0, g)$. The gradient of J_{MML} has the form:

$$\nabla_\theta J_{\mathrm{MML}} = \mathbb{E}_{s_0,g\sim p(s_0,g)} \sum_{\mathbf{z}\in\mathcal{B}} \frac{\pi_\theta(\mathbf{z}\mid s_0)}{\sum_{\hat{\mathbf{z}}\in\mathcal{B}} \pi_\theta(\hat{\mathbf{z}}\mid s_0)} \nabla_\theta \log \pi_\theta(\mathbf{z}\mid s_0) \tag{15}$$

Taking a step in the direction of J_{MML} up-weights the probability of high-reward trajectory $\mathbf{z}$ and thus attempts to up-weight each reward-earning trajectory. More discussion of this objective can be found in (Guu et al., 2017; Liang et al., 2018).

Choosing $w_l = 1$ in Equation 13, and clearly there exists a monotonically increasing function ρ_f satisfy $\rho_f(\frac{\pi_\theta(\mathbf{z}\mid s_0)}{\bar{\pi}(\mathbf{z})}) = \frac{\pi_\theta(\mathbf{z}\mid s_0)}{\sum_{\hat{\mathbf{z}}\in\mathcal{B}} \pi_\theta(\hat{\mathbf{z}}\mid s_0)}$. Choosing such ρ_f, GACA reduces to MML.

E.3 IML:

Iterative maximize likelihood(IML) (Liang et al., 2017; Abolafia et al., 2018) uniformly maximizes the likelihood of all the high-reward trajectories in past experience. The objective is given by $J_{\mathrm{IML}} = \mathbb{E}_{s_0,g\sim p(s_0,g)} \mathbb{E}_{\mathbf{z}\sim B} [\log \pi_\theta(\mathbf{z}\mid s_0) r(\mathbf{z}, s_0, g)]$. The gradient of IML is given by

$$\nabla_\theta J_{\mathrm{IML}} = \mathbb{E}_{s_0,g\sim p(s_0,g)} [\sum_{\mathbf{z}\in B} \nabla_\theta \log \pi_\theta(\mathbf{z}\mid s_0) r(\mathbf{z}, s_0, g)] \tag{16}$$

Choosing $\rho_f = 1$ and $w_{\mathcal{B}} = 1$ in Equation 13, GACA reduces to IML. For each given s_0, g, IML can be expressed as optimizing policy distribution by minimizing the reverse KL divergence between the parameterized policy distribution and an optimal policy distribution, *i.e.*, $D_{\mathrm{KL}}(\pi^\star \mid\mid \pi)$ where $\pi^\star$ is the optimal distribution. It's well-known that reverse KL-divergence promotes model-covering, thus IML seeks to explore diverse samples thus will have a higher chance of collecting high-reward trajectories. Recent work MAPOX (Agarwal et al., 2019) exploits this property of IML by running IML to collect diverse samples for training.

E.4 MAPO, MAPOX:

Memory Augmented Policy Optimization(MAPO) (Liang et al., 2018) is a recent method for reusing high-reward trajectories, it maximizes the expected reward and estimate the gradient with off-policy high-reward trajectories. The gradient of MAPO is

$$\begin{aligned}\nabla_\theta J_{\text{MAPO}} = \mathbb{E}_{s_0,g\sim p(s_0,g)}[&(1-\alpha)\mathbb{E}_{\mathbf{z}\sim\pi(\mathbf{z}|x)}\nabla_\theta \log \pi_\theta(\mathbf{z}|s_0) r(\mathbf{z}, s_0, g) \\ &+ \alpha \sum_{\mathbf{z}\in\mathcal{B}} \nabla_\theta \log \pi_\theta(\mathbf{z}|s_0) r(\mathbf{z}, s_0, g)]\end{aligned} \tag{17}$$

where α is a weight equals to the total probability of high-reward trajectory $\mathbf{z}$ in buffer $\mathcal{B}$.

MAPOX (Agarwal et al., 2019) improves MAPO by running MAPO on trajectories collected with IML for exploration. As shown previously, IML can be viewed as minimizing 'reverse' KL divergence between policy distribution $\pi_\theta(\mathbf{z}|s_0)$ and the prior distribution, thus IML promotes exploration. When choosing $\rho_f(\frac{\bar{\pi}(\mathbf{z})}{\pi_\theta(\mathbf{z}|s_0)}) = \log(\frac{\bar{\pi}(\mathbf{z})}{\pi_\theta(\mathbf{z}|s_0)}) - 1$ and setting $w_{\mathcal{B}} = 1$, GACA reduces to MAPO.

E.5 RAML:

Reward Augmented Maximum Likelihood(RAML) (Norouzi et al., 2016) is a more general variant of IML, which weights off-policy samples with an energy based prior,

$$\nabla_\theta J_{\text{RAML}} = \mathbb{E}_{s_0,g\sim p(s_0,g)}\mathbb{E}_{\mathbf{z}\sim \text{Z}}[\bar{\pi}(\mathbf{z}) \log \pi_\theta(\mathbf{z}|s_0) r(\mathbf{z}, s_0, g)] \tag{18}$$

where $\bar{\pi}(\mathbf{z}) = \exp\left(\frac{1}{\lambda}(r(\mathbf{z}, s_0, g) - V(x))\right)$ is the energy based prior distribution defined in Equation 7. Similar to IML, for each given s_0, g, RAML can be expressed as optimizing the policy distribution by minimizing the KL divergence between the parameterized policy distribution and an energy based optimal policy distribution defined as $\exp\left(\frac{1}{\lambda}(r(\mathbf{z}, s_0, g) - V(x))\right)$. The gradient estimation is over possible trajectories $\mathbf{z} \sim \text{Z}$, only few of them are high-reward and most trajectories cannot guide the policy learn good behavior, thus RAML suffers from high sample complexity. Choosing $\rho_f(\frac{\bar{\pi}(\mathbf{z})}{\pi_\theta(\mathbf{z}|s_0)}) = \frac{\bar{\pi}(\mathbf{z})}{\pi_\theta(\mathbf{z}|s_0)}$ and $w_{\mathcal{B}} = 1$ in Equation 13, GACA reduces to RAML.

F Experiments Details

For WikiTableQuestions, we follow the construction in Pasupat & Liang (2015) for converting a table into a directed graph that can be queried. The rows and cells are converted to graph nodes while column names become labeled directed edges. Each batch includes samples from 25 examples For WikiSQL, we follow the setting in Liang et al. (2018) for choosing the sampling batch size. Our model use a seq2seq model as $\pi_\theta(\mathbf{z}|s_0)$, and two key-variable memory as high-reward buffer B and zero-reward buffer C, and associated with a domain specific language interpreter (Liang et al., 2017). Table 1 shows a comparison GACA with various baselines on two challenging sparse reward program synthesis tasks, where we also present an ablation study of each technique in GACA. Specifically, we studied the performance of GACA w/o AG which represents GACA without adaptive gradient estimation(Section 3.3), and GACA w/o GP which represents GACA without guiding prior(Section 3.2). In detail, GACA w/o AG means use standard KL-divergence to calculate gradient in Eq 13 instead of f-divergence, while GACA w/o GP means don't learn the prior policy distribution as in 8, but fix prior policy distribution to the energy based distribution as in 7. Our code is based on open source implementation of MAPO (Liang et al., 2018) which implements a distributed actor-learner architecture Espeholt et al. (2018) to accelerate sampling through distributed actors. We also use open source code from MeRL (Agarwal et al., 2019), tail-adapted variational inference (Wang et al., 2018). Our experiments follow the settings of MAPO (Liang et al., 2018) and MeRL (Agarwal et al., 2019).

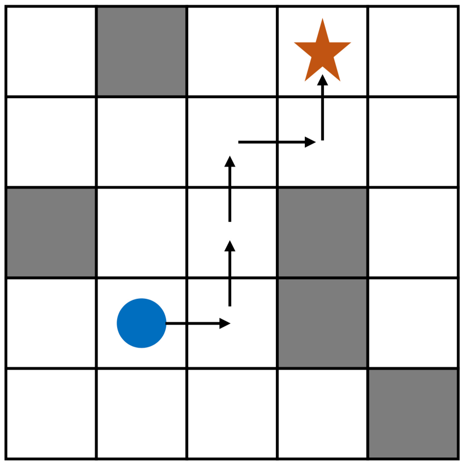

Figure 5: Instruction following navigation in maze. An agent is presented with a sequence of (Left, Right, Up, Down) instructions. Given the input text, the agent on the blue dot need to perform a sequence of actions, and only receives a reward of 1 if it reaches the goal at the orange star.

We port their code to PyTorch (Paszke et al., 2017) and implement GACA on top of them to conduct experiments. We will release the code later. Gradients are estimated and periodically updated through a central learner (Espeholt et al., 2018). For TextWorld[1], we use a set of 300 randomly generated environments with training and validation splits of 80% and 20% respectively following Agarwal et al. (2019). The agent is evaluated on 300 unseen test environments from the same distribution. An example of TextWorld is shown in Figure 5. We used the Adam Optimizer (Kingma & Ba, 2015) for WikiSQL, WikiTABLE, and TextWorld. We performed hyper-parameter sweeps via random search over the interval $\left(10^{-4}, 10^{-2}\right)$ for learning rate. All the hyperparameters are tuned on the evaluation set.

G Qualitative Results

In order to evaluate the qualitative quality of the proposed method, we compare GACA with the recent state-of-the-art MAPO on WIKITABLE-QUESTIONS. Figure 5 shows examples of generated programs from natural language queries using model trained with GACA or MAPO, the difference between generated programs show that sometimes GACA is capable of generating correct programs that capture the meaning of the natural language queries while MAPO generates either wrong answer programs or spurious programs.

[1]https://github.com/google-research/google-research/tree/master/meta_reward_learning/textworld

Query/Generated Programs	Comment
Query nu-1147: **Which opponent of the kansas city chiefs in 1987 saw a total of more than 70,000 in attendance?** MAPO: r_0 = (argmax all_rows r.attendance-number); r_{res} = (hop r_0 r.opponent-string) GACA: r_0 = (filter$_>$ all_rows [70,000] r.attendance-number); r_{res} = (hop r_0 r.opponent-string)	The program generated by MAPO leads to wrong answer, because it wrongly assume that the one with largest number of attendance number is the opponent of kansas city. Instead GACA captures the semantic meaning of the query and generates correct program.
Query nu-1167: **Who was the first oldest living president?** MAPO: r_0 = (first all_rows); r_{ans} = (hop r_0 r.president) GACA: r_0 = (filter_str_contain_any all_rows [oldest living president'] r.became_oldest_living_president-string); r_{res} =(hop r_0 r.president-string)	MAPO gets correct answer by chance because it is spurious program, MAPO wrongly assumes the data is ordered, in contrast, GACA generates programs that capture underlying semantic information.
Query nu-3733: **At least how many more people attended gamestorm 15 than gamestrom 13?** MAPO: r_0 = (filter_str_contain_any all_rows [13] r.dates-string); r_{res} = (hop r_0 r.attendance-number) GACA: r_0 = (filter_str_contain_any all_rows [13] r.dates-string); r_1 = (filter_str_contain_any all_rows [13] r.dates-string); (diff r_0 r_1 r.attendance-number)	MAPO gets correct answer by chance because it is spurious program, MAPO wrongly assumes the data is ordered, in contrast, GACA generates programs that capture underlying semantic information.

Table 5: Example of generated programs from models trained using MAPO and GACA on WikiTableQuestions.