Notes on On-Policy Distillation

Qinqing Zheng

July 2026

1 On-Policy Distillation in MDPs

Let $s_0 \sim \rho$ and τ be a trajectory, π_E be the teacher policy. We have the induced trajectory distribution[1]

$$P_{\pi_\theta}(\tau) = \rho(s_0) \prod_{t=0}^{|\tau|-1} \pi_\theta(a_t|s_t) P(s_{t+1}|s_t, a_t) \tag{1.1}$$

and similarly

$$P_{\pi_E}(\tau) = \rho(s_0) \prod_{t=0}^{|\tau|-1} \pi_E(a_t|s_t) P(s_{t+1}|s_t, a_t). \tag{1.2}$$

The objective function for OPD to minimize under MDP setting is sequence-level reverse KL

$$\begin{aligned} J_{\text{opd}}(\pi_\theta) = &\quad KL[P_{\pi_\theta} \| P_{\pi_E}] \\ = &\quad \mathbb{E}_{\tau \sim P_{\pi_\theta}} \left[\log \frac{P_{\pi_\theta}(\tau)}{P_{\pi_E}(\tau)} \right] \\ = &\quad \mathbb{E}_{\tau \sim P_{\pi_\theta}} \left[\textstyle\sum_{t=0}^{|\tau|-1} \log \frac{\pi_\theta(a_t|s_t)}{\pi_E(a_t|s_t)} \right]. \end{aligned} \tag{1.3}$$

Let $F_\theta(\tau) = \log \frac{P_{\pi_\theta}(\tau)}{P_{\pi_E}(\tau)}$. Analogous to the policy gradient derivation, the OPD gradient can be written as

$$\begin{aligned} \nabla_\theta J_{\text{opd}} = &\quad \nabla_\theta \mathbb{E}_{\tau \sim P_{\pi_\theta}} \big[F_\theta(\tau) \big] \\ = &\quad \sum_\tau \nabla_\theta P_{\pi_\theta}(\tau) F_\theta(\tau) + \sum_\tau P_{\pi_\theta}(\tau) \nabla_\theta F_\theta(\tau) \\ \overset{(i)}{=} &\quad \sum_\tau \nabla_\theta P_{\pi_\theta}(\tau) F_\theta(\tau) \\ = &\quad \sum_\tau P_{\pi_\theta}(\tau) \nabla_\theta \log P_{\pi_\theta}(\tau) F_\theta(\tau) \\ = &\quad \mathbb{E}_{\tau \sim P_{\pi_\theta}} \left[F_\theta(\tau) \textstyle\sum_{t=0}^{|\tau|-1} \nabla \log \pi_\theta(a_t|s_t) \right], \end{aligned} \tag{1.4}$$

[1] RL/OPD objective can be formulated in several equivalent ways, using trajectory distribution or state-visitation distributions. Here, the trajectory-distribution formulation leads to a simpler derivation.

where (i) follows from $\sum_\tau P_{\pi_\theta}(\tau)\nabla_\theta F_\theta(\tau) = 0$. Next, define cost $c_t = \log \frac{\pi_\theta(a_t|s_t)}{\pi_E(a_t|s_t)}$ and let $W_t = \sum_{k=t}^{|\tau|} c_t$ be the cost-to-go. We then have

$$\begin{aligned}
\nabla_\theta J_{\text{opd}} = \quad & \mathbb{E}_{\tau\sim P_{\pi_\theta}}\left[\left(\sum_{t=0}^{|\tau|-1} c_t\right)\cdot\sum_{t=0}^{|\tau|-1}\nabla\log\pi_\theta(a_t|s_t)\right] \\
= \quad & \mathbb{E}_{\tau\sim P_{\pi_\theta}}\left[\sum_{t=0}^{|\tau|-1}\nabla\log\pi_\theta(a_t|s_t)\cdot\left(\sum_{k<t} c_k + \sum_{k\geqslant t} c_k\right)\right] \\
= \quad & \underbrace{\mathbb{E}_{\tau\sim P_{\pi_\theta}}\left[\sum_{t=0}^{|\tau|-1}\left(\nabla\log\pi_\theta(a_t|s_t)\cdot\left(\sum_{k<t} c_k\right)\right)\right]}_{T_1} + \mathbb{E}_{\tau\sim P_{\pi_\theta}}\left[\sum_{t=0}^{|\tau|-1} W_t\nabla\log\pi_\theta(a_t|s_t)\right] \\
\overset{(ii)}{=} \quad & \mathbb{E}_{\tau\sim P_{\pi_\theta}}\left[\sum_{t=0}^{|\tau|-1} W_t\nabla\log\pi_\theta(a_t|s_t)\right],
\end{aligned} \tag{1.5}$$

where in (ii) we used

$$\begin{aligned}
T_1 =& \mathbb{E}_{\tau\sim P_{\pi_\theta}}\left[\sum_{t=0}^{|\tau|-1}\left(\nabla\log\pi_\theta(a_t|s_t)\cdot\left(\sum_{k<t} c_k\right)\right)\right] \\
=& \sum_{t=0}^{|\tau|-1}\mathbb{E}_{\tau\sim P_{\pi_\theta}}\left[\nabla\log\pi_\theta(a_t|s_t)\cdot\left(\sum_{k<t} c_k\right)\right] \\
=& \sum_{t=0}^{|\tau|-1}\mathbb{E}_{(s_0,a_0,\ldots,s_t)\sim P_{\pi_\theta}^{<t}}\mathbb{E}_{a_t\sim\pi_\theta(\cdot|s_t)}\left[\nabla\log\pi_\theta(a_t|s_t)\cdot\left(\sum_{k<t} c_k\right)\right] \\
=& \sum_{t=0}^{|\tau|-1}\mathbb{E}_{(s_0,a_0,\ldots,s_t)\sim P_{\pi_\theta}}\left[\left(\sum_{k<t} c_k\right)\cdot\underbrace{\mathbb{E}_{a_t\sim\pi_\theta(\cdot|s_t)}\left[\nabla\log\pi_\theta(a_t|s_t)\right]}_{0}\right] \\
=& 0.
\end{aligned} \tag{1.6}$$

By comparing Equation (1.3), (1.5) with standard policy optimization objective and its gradient, we obtain an direct analogy between OPD and RL. Specifically,

> *OPD – minimizing trajectory-level reverse KL – can be interpreted as solving an undiscounted RL problem under policy-dependent reward function, whose per-step reward is sampled log-likelihood ratio* $\log\frac{\pi_E(a_t|s_t)}{\pi_\theta(a_t|s_t)}$.

	reward	return	discount
RL	r_t	$\sum_{t=0}^{|\tau|-1}\gamma^t r_t$	γ
OPD	$-c_t = \log\frac{\pi_E(a_t\|s_t)}{\pi_\theta(a_t\|s_t)}$	$-\sum_{t=0}^{|\tau|-1} c_t$	1

Table 1: Analogy between OPD and RL.

Remarks. Unlike standard RL where r_t is soledy defined by the environment (i.e., the MDP), the reward of OPD under this interpretation is policy-dependent. This leads to the following properties:

1. RL optimizes a fixed MDP. Under this interpretation, the analogous MDP of OPD changes over the course of training. Therefore, the analogy is limited to optimization formulation.

2. Simiarly, since the reward cannot be separated from policy, simply reduce $\pi_\theta(a_t|s_t)$ does not neccesarrily increase the expected return even though it increases the per-step reward, because π_θ also determines the sampling distribution under which the expected return is computed.

2 On-Policy Distillation for LLM

Let x be the prompt and y be the completion. The results from Section 1 directly extend to the autoregressive setting by identifying $s_t \equiv (x, y_{<t})$, $a_t \equiv y_t$, $\pi_\theta(a_t|s_t) \equiv \pi_\theta(y_t|x, y_{<t})$ with trajectory distribution

$$P_{\pi_\theta}(\tau) = \rho(x) \prod_{t=1}^{|y|} \pi(y_t|x, y_{<t}). \tag{2.1}$$

The OPD objective becomes

$$J_{\text{opd}}(\pi_\theta) = \mathbb{E}_{\tau \sim P_{\pi_\theta}} \left[\textstyle\sum_{t=1}^{|y|} \log \frac{\pi_\theta(y_t|x,y_{<t})}{\pi_E(y_t|x,y_{<t})} \right], \tag{2.2}$$

and its gradient can be written as

$$\nabla_\theta J_{\text{opd}} = \mathbb{E}_{\tau \sim P_{\pi_\theta}} \left[\textstyle\sum_{t=1}^{|y|} \left(\sum_{k=t}^{|y|} \log \frac{\pi_E(y_k|x, y_{<k})}{\pi_\theta(y_k|x, y_{<k})} \right) \cdot \nabla \log \pi_\theta(y_t|x, y_{<t}) \right]. \tag{2.3}$$

3 Practical Optimization of OPD

In practice, computing J_{opd} exactly often poses non-trivial implementation challenges in distributed training system. Besides, it also incurs substaintial memory consumption. Consequently, existing OPD implementation mainly follow one of two approaches:

1. *Construct RL surrogate objective and use policy-gradient optimization.*

   Interpret the reverse KL objective as an RL objective. In each iteration, construct a surrogate loss whose gradient matches (2.3), while applying `StopGrad` to the sampled log-likelihood ratio.

2. *Optimizing approximated reverse KL loss.*

However, each approach comes with important caveats.

1. For the RL surrogate approach, practical open-source implementations often use token-level log-likelihood ratio $\log \frac{\pi_E(y_t|x,y_{<t})}{\pi_\theta(y_t|x,y_{<t})}$ instead of the proper cost-to-go, i.e., the cumulative future log-likelihood ratio $\sum_{k=t}^{|y|} \log \frac{\pi_E(y_k|x,y_{<k})}{\pi_\theta(y_k|x,y_{<k})}$.

   This replacement ignores the dependency of future tokens $y_{>t}$ on the current token y_t, and therefore does not in general recover the exact OPD gradient. We return to this issue below.

2. The KL approximation approach presents a different set of subtleties.

   First, it is easy to see the sequence-level KL can be decomposed into the sum of token-level KL:

$$\begin{aligned} KL\Big[P_{\pi_\theta} \| P_{\pi_E}\Big] &= \sum_{t=0}^{|y|} \mathbb{E}_{y_{<t}\sim P_{\pi_\theta}^{<t}} \Big(KL\big[\pi(\cdot|x,y_{<t}) \| \pi_E(\cdot|x,y_{<t})\big] \Big) \\ &= \mathbb{E}_{y\sim P_{\pi_\theta}} \Big(\sum_{t=0}^{|y|} KL\big[\pi(\cdot|x,y_{<t}) \| \pi_E(\cdot|x,y_{<t})\big] \Big). \end{aligned} \tag{3.1}$$

   In practical implementations, each token-level KL term is either estimated from samples or approximated using truncated distribution. Let f_t denote either construction. The practical objective is

$$\sum_{t=0}^{|\tau|} f_t(\pi_\theta, \pi_E), \tag{3.2}$$

   and the gradient is computed as

$$\sum_{t=0}^{|\tau|} \nabla_\theta f_t(\pi_\theta, \pi_E). \tag{3.3}$$

   There common choices for f_t include:

   - sample-based KL estimator, such as k_2, k_3 estimators (Schulman, 2020).
   - KL divergence computed exactly between top-K approximation of $\pi_\theta(\cdot|x,y_{<t})$ and $\pi_E(\cdot|x,y_{<t})$.

   Top-k approximation will introduce truncation bias to both the objective and its gradient. The sample-based approach can provide unbiased estimation of the objective, but the gradient estimator (3.3) might still be biased for two different reasons:

   (i) An unbiased KL estimator f_t does not necessarily imply that $\nabla_\theta f_t$ is an unbiased estimator of the KL gradient. This is because the sampling distribution itself depends on θ, differentiating the expecatation introduces an additional score-fucntion term that is not captured by naively differentiating the sample estimator. For example, the commonly used low-variance estimator k_3 is an unbiased estimator of KL, but differentiating it directly produces a biased gradient. In contrast, the gradient obtained from k_2 is unbiased. See, e.g., Tang and Munos (2025), for detailed discussion.

Method	Token-level KL		Sequence-level KL (OPD)	
	Objective	Gradient	Objective	Gradient
Sample estimator (k_2)	Unbiased	Unbiased	Unbiased	Biased
Sample estimator (k_3)	Unbiased	Biased	Unbiased	Biased
Top-k KL approximation	Biased	Biased	Biased	Biased
RL surrogate (token-level log-ratio)	N/A	N/A	N/A	Biased
RL surrogate (2.2)	N/A	N/A	N/A	Unbiased

Table 2: Bias properties of practical OPD implementations.

(ii) For estimators like k_2, even thought it yields unbiased estimate of token-level KL gradient, the resulting gradient (3.3) remains biased because it ignores the effect of the current token y_t on future tokens $y_{>t}$, and consequencely, on all subsequent token-level KL terms.

References

John Schulman. Approximating kl divergence. *http://joschu.net/blog/kl-approx.html*, 2020.

Yunhao Tang and Rémi Munos. On a few pitfalls in kl divergence gradient estimation for rl. *arXiv preprint arXiv:2506.09477*, 2025.