DiCE: The Infinitely Differentiable Monte-Carlo Estimator

Jakob Foerster[1], Gregory Farquhar[1]*, Maruan Al-Shedivat[2]*, Tim Rocktäschel[1], Eric Xing[2], Shimon Whiteson[1]

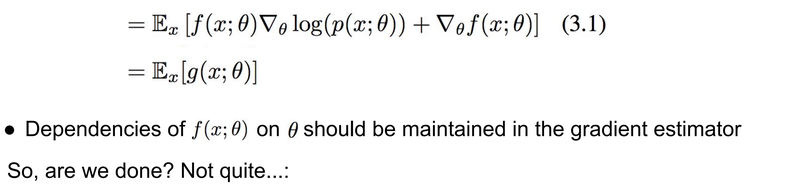
[1]University of Oxford, [2]Carnegie Mellon Universtiy

Motivation

- The score function is commonly used in order to estimate gradients in Reinforcement Learning problems.
- This can be calculated via the Surrogate Loss approach.
- For order derivatives there is currently no satisfactory solution.
- Examples include Meta-Learning for RL and 2nd order optimisation.

Setting and background

- Stochastic Computation Graphs provide a framework for estimating gradients using *Surrogate Losses* (John Schulman et al 2015):



- Surrogate Loss Approach:

$$\mathrm{SL}(\Theta, \mathcal{S}) := \sum_{w \in \mathcal{S}} \log p(w \mid \mathrm{DEPS}_w) \hat{Q}_w + \sum_{c \in \mathcal{C}} c(\mathrm{DEPS}_c)$$

Differentiation

In the Surrogate Loss, Q is treated as a fixed sample independent of parameters (denoted by hat).

- Gradient Estimator:

$$\frac{\partial}{\partial \theta} \mathbb{E}\left[\sum_{c \in \mathcal{C}} c\right] = \mathbb{E}\left[\sum_{\substack{w \in \mathcal{S}, \\ \theta \prec^D w}} \left(\frac{\partial}{\partial \theta} \log p(w \mid \mathrm{DEPS}_w)\right) \hat{Q}_w + \sum_{\substack{c \in \mathcal{C} \\ \theta \prec^D c}} \frac{\partial}{\partial \theta} c(\mathrm{DEPS}_c)\right]$$

- Surrogate Loss Approach cuts dependency of objective on parameters (^)
- This will cause errors in higher-order derivatives
- Compare with exact differentiation:

$$\begin{aligned}
\nabla_\theta \mathcal{L} &= \nabla_\theta \mathbb{E}_x\left[f(x;\theta)\right] \\
&= \nabla_\theta \sum_x p(x;\theta) f(x;\theta) \\
&= \sum_x \nabla_\theta \big(p(x;\theta) f(x;\theta)\big) \\
&= \sum_x \big(f(x;\theta) \nabla_\theta p(x;\theta) + p(x;\theta) \nabla_\theta f(x;\theta)\big) \\
&= \sum_x \big(f(x;\theta) p(x;\theta) \nabla_\theta \log(p(x;\theta)) + p(x;\theta) \nabla_\theta f(x;\theta)\big) \\
&= \mathbb{E}_x\left[f(x;\theta) \nabla_\theta \log(p(x;\theta)) + \nabla_\theta f(x;\theta)\right] \quad (3.1) \\
&= \mathbb{E}_x[g(x;\theta)]
\end{aligned}$$

- Dependencies of $f(x;\theta)$ on θ should be maintained in the gradient estimator

So, are we done? Not quite...:

- How do we compute these gradients efficiently?
- Ideally would like to have an objective that can be differentiated multiple times.
- The gradient of the estimator should produce estimate of the gradient

Method: DiCE

- Define new objective:

$$\mathcal{L}_{\boxdot} = \sum_{c \in \mathcal{C}} \boxdot(\mathcal{W}_c) c$$

Where:

1. $\boxdot(\mathcal{W}) \rightarrowtail 1$,
2. $\nabla_\theta \boxdot(\mathcal{W}) = \boxdot(\mathcal{W}) \sum_{w \in \mathcal{W}} \nabla_\theta \log(p(w;\theta))$

- Meets our requirements!

$$\Rightarrow \mathbb{E}[\nabla_\theta^n \mathcal{L}_{\boxdot}] \rightarrowtail \nabla_\theta^n \mathcal{L}, \forall n \in \{0, 1, 2, \dots\}$$

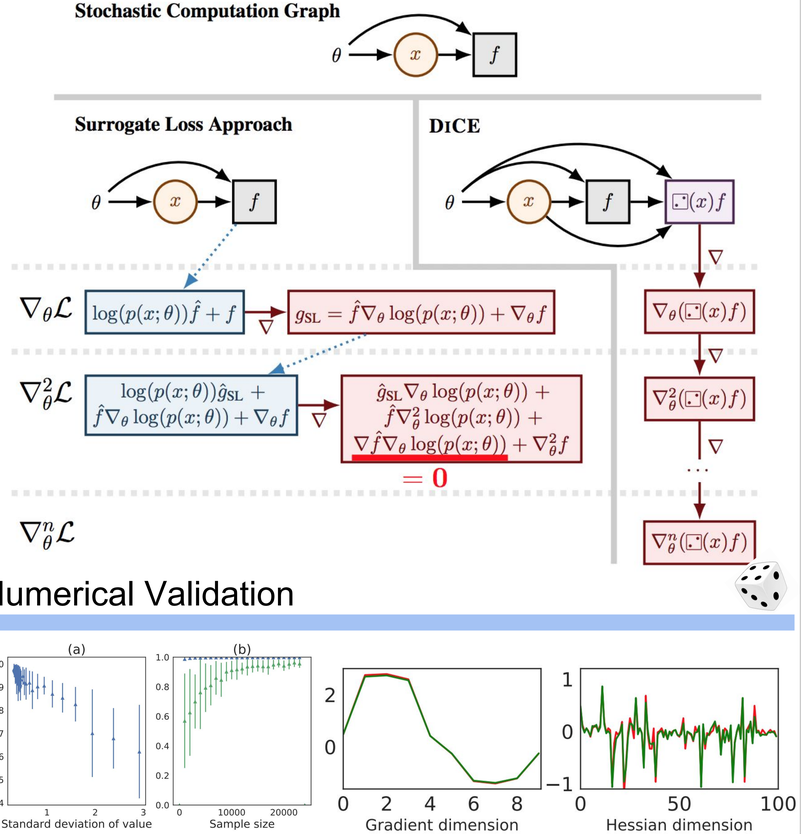
Implementation

- DiCE can be implemented easily in standard deep learning libraries:

$$\boxdot(\mathcal{W}) = \exp\left(\tau - \bot(\tau)\right), \quad \tau = \sum_{w \in \mathcal{W}} \log(p(w;\theta)), \quad \bot(\tau) : \text{stop\_gradient}$$

- Can include baseline naturally

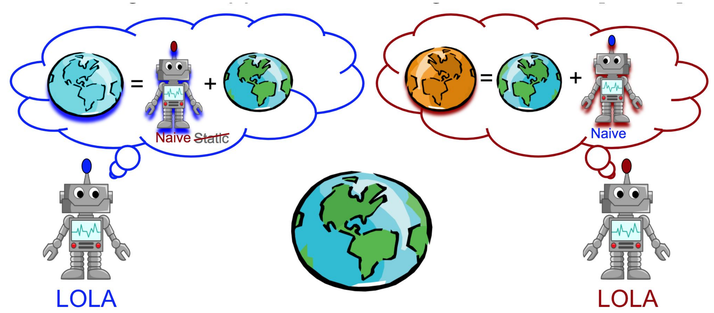
$$\mathcal{L}_{\boxdot} = \sum_{c \in \mathcal{C}} \boxdot(\mathcal{W}_c) c + \sum_{w \in \mathcal{S}} (1 - \boxdot(\{w\})) b_w$$



Numerical Validation

Application Example: LOLA-DiCE

- When multiple agents are learning the learning step of each of the agents depends on the policy of all other agents.
- "Learning with Opponent-Learning Awareness" (Foerster et al 2017) explicitly accounts for this dependency and differentiates through the learning step:

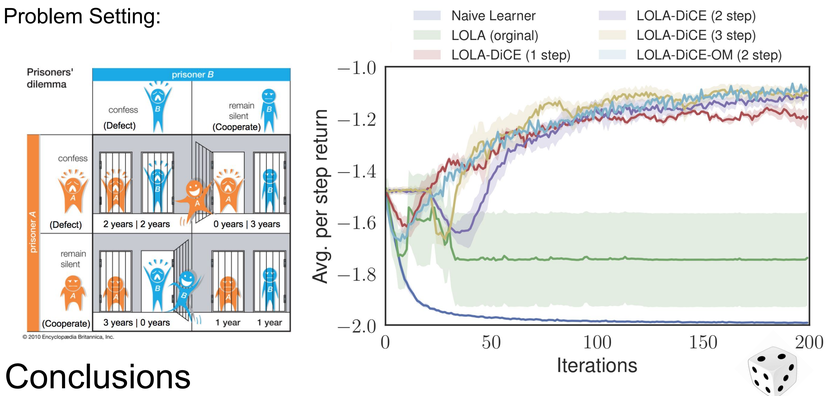


- The original paper relied on a Taylor expansion, second order gradients and a large batch size (4000). LOLA can be implemented exactly using DiCE:

$$\mathcal{L}^1_{\boxdot}(\theta_1, \theta_2)_{\mathrm{LOLA}} = \mathbb{E}_{\pi_{\theta_1}, \pi_{\theta_2 + \Delta\theta_2(\theta_1, \theta_2)}}\left[\mathcal{L}^1_{\boxdot}\right]$$

$$\Delta\theta_2(\theta_1, \theta_2) = \alpha \nabla_{\theta_2} \mathbb{E}_{\pi_{\theta_1}, \pi_{\theta_2}}\left[\mathcal{L}^2_{\boxdot}\right],$$

Problem Setting:



Conclusions

- DiCE offers a new approach to estimating gradients in stochastic computation graphs
- Arbitrary order gradients of the DiCE objective correspond to gradient estimators
- DiCE can be implemented easily in standard learning frameworks!

Acknowledgements and References

Thanks to: Oxford-Google DeepMind Graduate Scholarship, the UK EPSRC CDT in Autonomous Intelligent Machines and Systems

[1] "Gradient Estimation using Stochastic Computation Graphs" (John Schulman, Nicolas Heess, Theophane Weber, Pieter Abbeel)
[2] "Learning with Opponent-Learning Awareness" (Jakob N. Foerster, Richard Y. Chen, Maruan Al-Shedivat, Shimon Whiteson, Pieter Abbeel, Igor Mordatch)