DiCE: The Infinitely Differentiable Monte Carlo Estimator*

Foerster et al.

2018

1 What

A magic objective which can be used repeatedly to obtain correct higher order gradients under Stochastic Computation Graph (SCG) formalism. My previous note on SCG [1] has more information on SCG and why they are cool.

The authors prove and empirically show that DiCE works and provide TensorFlow implementation.

2 Why

The score function trick (or logtrick or REINFORCE etc.) is widely used to estimate the gradient when you cannot just differentiate the objective. We can easily do this with first-order gradients. Moreover, we can apply our ULTRADEEPLEARNING toolboxes to do auto-diff on the differentiable objective whose gradient is the same as the gradient of the original objective. [Schulman et al., 2015] describes the SCG formalism and does this in the *surrogate loss* (SL) approach.

Unfortunately, our life gets much harder when we need higher order gradients, e.g. in meta-learning or multi-agent RL, when we need to differentiate through other agents. Up until now, we could either analytically derive the estimators or repeatedly apply SL approach constructing a new objective every new iteration. Both of the methods are far from being simple. Moreover, as shown in the paper, treating part of the cost as a fixed sample leads to incorrect gradients.

DiCE to the rescue! DiCE allows to use the single objective and apply it repeatedly to obtain higher order grads for your problem letting auto-diff work for you instead of you manipulating the graph.

*Notes by Vitaly Kurin `https://yobibyte.github.io/`

[1] `https://yobibyte.github.io/files/paper_notes/scg.pdf`

3 How

The background is mostly the same as in [Schulman et al., 2015]

Let's start with understanding what's wrong with just taking a gradient of the SL and treat this expression as a new SL and take the gradient of it?

$$\nabla_\theta \mathcal{L} = \nabla_\theta \, \mathbb{E}_x[f(x;\theta)] = \mathbb{E}_x \left[f(x;\theta) \nabla_\theta \log p(x;\theta) + \nabla_\theta f(x;\theta) \right] = \mathbb{E}_x \left[g(x;\theta) \right] \tag{1}$$

The first component inside of the expectation is just the SF trick, the second comes because of the dependence of f on θ.

Now, SL and its grad look like that:

$$SL(\mathcal{L}) = \log p(x;\theta) \hat{f}(x) + f(x;\theta) \tag{2}$$

$$(\nabla_\theta \mathcal{L})_{SL} = \mathbb{E}_x \left[\nabla_\theta \, SL(\mathcal{L}) \right] \tag{3}$$

$$= \mathbb{E}_x [\hat{f}(x) \nabla_\theta \log p(x;\theta) + \nabla_\theta f(x;\theta)] \tag{4}$$

$$= \mathbb{E}_x \left[g_{SL}(x;\theta) \right]. \tag{5}$$

Let's repeat:

$$SL(g_{SL}(x;\theta)) = \log p(x;\theta) \hat{g}_{SL}(x) + g_{SL}(x;\theta) \tag{6}$$

$$(\nabla^2_\theta \mathcal{L})_{SL} = \mathbb{E}_x \left[\nabla_\theta \, SL(g_{SL}) \right] \tag{7}$$

$$= \mathbb{E}_x [\hat{g}_{SL}(x) \nabla_\theta \log p(x;\theta) + \nabla_\theta g_{SL}(x;\theta)] \tag{8}$$

$$\tag{9}$$

But what if we get the grad for the real loss, not the SL:

$$\nabla^2_\theta \mathcal{L} = \nabla_\theta \, \mathbb{E}_x \left[g(x;\theta) \right] = \mathbb{E}_x \left[g(x;\theta) \nabla_\theta \log p(x;\theta) + \nabla_\theta g(x;\theta) \right] \tag{10}$$

Aha! $\nabla_\theta g(x;\theta)$ is not the same as $\nabla_\theta g_{SL}(x;\theta)$:

$$\nabla_\theta g(x;\theta) = \nabla_\theta f(x;\theta) \nabla_\theta \log p(x;\theta) + f(x;\theta) \nabla^2_\theta \log p(x;\theta) + \nabla_\theta f(x;\theta) \tag{11}$$

$$\nabla_\theta g_{SL}(x;\theta) = \hat{f}(x) \nabla^2_\theta \log p(x;\theta) + \nabla^2_\theta f(x;\theta) \tag{12}$$

$$\tag{13}$$

Product rule rules!

A toy example with $x \sim Ber(\theta)$ and $f(x,\theta) = x(1-\theta) + (1-x)(1+\theta)$ shows that the approach with computing the grad of an SL twice is indeed not correct.

The authors introduce $\boxdot$ operator which has the two following properties:

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

The weird arrow means 'model.forward()'.

The DiCE objective is:

$$\mathcal{L}_{\boxdot} = \sum_{c \in \mathcal{C}} \boxdot(\mathcal{W}_c)c \tag{14}$$

The authors prove that $\mathbb{E}[\nabla_\theta^n \mathcal{L}_{\boxdot}] \rightarrowtail \nabla_\theta^n \mathcal{L}, \forall n \in \{0, 1, 2, ...\}$ and this allows us to compute as many orders of the gradient as we want.

This is how you implement this in your favourite library:

$$\boxdot(\mathcal{W}) = \exp(\tau - \perp(\tau)), \tau = \sum_{w \in \mathcal{W}} \log(p(w; \theta)) \tag{15}$$

Taking the gradient of this we can get the second property from above.

The paper has an interesting and insightful notice about formulas (5) and (6) in [Schulman et al., 2015]. The former sums over the stochastic nodes and multiplies the grad of the logprob with a sum of the downstream costs. SL approach relies on this formulation.

In contrast, DiCE relies on formula (6) and sums over costs and multiplies each of the cost with a sum over the gradients of logprobs from upstream stochastic nodes. The authors argue that this helps in further order gradient estimation since we maintain the same backward-looking dependencies each step. Moreover, it's more intuitive as it is centred around the original objective.

What about the variance reduction? Here you are:

$$\mathcal{L}_{\boxdot} = \sum_{c \in \mathcal{C}} \boxdot(\mathcal{W}_c)c + \sum_{win\S} (1 - \boxdot(w))b_w, \tag{16}$$

where b_w is any function of nodes not influenced by w.

4 Evaluation

Apart from the toy example where the authors show the failure case, they apply DiCE for iterated prisoner's dillemma [2].

First, they estimate gradients of the expected return with DiCE for fixed policies and show that DiCE estimators match the grads and the Hessians. Moreover, they also show how the sample size influences the estimation and how the baseline improves the estimation.

Next, they apply DiCE to LOLA [Foerster et al., 2017] where you need to differentiate through your opponents' learning process. This results in a much more stable and requires smaller batch sizes.

[2] `https://en.wikipedia.org/wiki/Prisoner's_dilemma#The_iterated_prisoner's_dilemma`

5 Comments

- Don't quite get the second property of the MAGICBOX as the authors put it 'indicates the new graph nodes that will be constructed to hold the gradients of that object'.
- The paper is another great example of the fact, that you should always check what people consider to be correct =).

References

[Foerster et al., 2017] Foerster, J. N., Chen, R. Y., Al-Shedivat, M., Whiteson, S., Abbeel, P., and Mordatch, I. (2017). Learning with opponent-learning awareness. *arXiv preprint arXiv:1709.04326*.

[Schulman et al., 2015] Schulman, J., Heess, N., Weber, T., and Abbeel, P. (2015). Gradient estimation using stochastic computation graphs. In *Advances in Neural Information Processing Systems*, pages 3528–3536.