Gradient Estimation II: 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]

Presenter: Amanjit Singh Kainth

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

Motivation

Estimating Expectations

- Loss functions in various machine learning problems are usually defined as an expectation over a set of random variables
- Estimating gradients of loss functions $\mathcal{L}(\theta) = \mathbb{E}_{x \sim p(x \mid \theta)}[f(x;\theta)]$ using samples essential to gradient-based optimization
- Utilize framework of graphical models to specify models involving stochastic computation, and derive estimators for $\boldsymbol{\nabla}_\theta \mathcal{L}$

Motivation

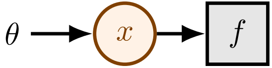
Estimating Expectations

- Loss functions in various machine learning problems are usually defined as an expectation over a set of random variables
- Estimating gradients of loss functions $\mathcal{L}(\theta) = \mathbb{E}_{x \sim p(x \mid \theta)} [f(x; \theta)]$ using samples essential to gradient-based optimization
- Utilize framework of graphical models to specify models involving stochastic computation, and derive estimators for $\boldsymbol{\nabla}_\theta \mathcal{L}$
- This paper: Construct modified objective $\mathcal{L}_{⚁} \rightarrowtail \mathcal{L}$ so that an automatic differentiation package yields unbiased estimators for gradients of any order *i.e.* $\boldsymbol{\nabla}_\theta^n \mathcal{L} = \mathbb{E} [\boldsymbol{\nabla}_\theta^n \mathcal{L}_{⚁}]$

Stochastic computation graphs (SCGs)

Schulman et al. (2015)

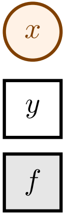
θ	Input node
x (circle)	Stochastic node
y (box)	Deterministic node
f (shaded box)	Cost node



Stochastic computation graphs (SCGs)

Schulman et al. (2015)

θ Input node

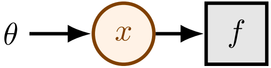


Stochastic node

Deterministic node

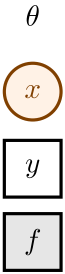
Cost node

Objective: $\mathcal{L} = \mathbb{E}_{x \sim p(x;\theta)}\left[f(x)\right]$



Stochastic computation graphs (SCGs)

Schulman et al. (2015)



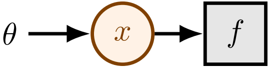
Input node

Stochastic node

Deterministic node

Cost node

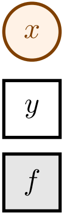
Objective: $\mathcal{L} = \mathbb{E}_{x \sim p(x;\theta)}\left[f(x)\right]$



Goal: $\boldsymbol{\nabla}_\theta \mathcal{L}$

Stochastic computation graphs (SCGs)

Schulman et al. (2015)



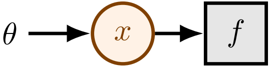
θ Input node

Stochastic node

Deterministic node

Cost node

Objective: $\mathcal{L} = \mathbb{E}_{x \sim p(x;\theta)}\left[f(x)\right]$

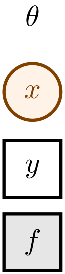


Goal: $\boldsymbol{\nabla}_\theta \mathcal{L}$

$$\boldsymbol{\nabla}_\theta \mathcal{L} = \boldsymbol{\nabla}_\theta \mathbb{E}_x\left[f(x)\right] = \mathbb{E}_x \overbrace{\left[f(x)\frac{\partial}{\partial\theta}\log p(x;\theta)\right]}^{\text{Score function (SF) estimator}}$$

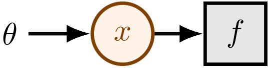
Stochastic computation graphs (SCGs)

Schulman et al. (2015)



- Input node
- Stochastic node
- Deterministic node
- Cost node

Objective: $\mathcal{L} = \mathbb{E}_{x \sim p(x;\theta)}\left[f(x)\right]$

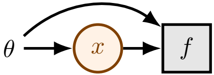
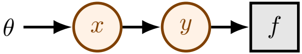
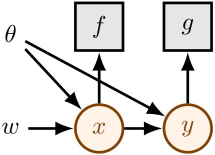
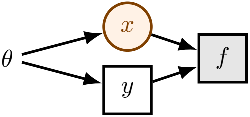


Goal: $\boldsymbol{\nabla}_\theta \mathcal{L}$

$$\boldsymbol{\nabla}_\theta \mathcal{L} = \boldsymbol{\nabla}_\theta \mathbb{E}_x\left[f(x)\right] = \mathbb{E}_x \overbrace{\left[f(x)\frac{\partial}{\partial\theta}\log p(x;\theta)\right]}^{\text{Score function (SF) estimator}}$$

SCGs: compute the most efficient gradient estimator(s)

Gradient Estimators and SCGs

SCG	$\mathcal{L}$	$\hat{g}, \nabla_\theta \mathcal{L} = \mathbb{E}\left[\hat{g}\right]$
	$\mathbb{E}_x\left[f(x;\theta)\right]$	$f(x;\theta)\frac{\partial}{\partial\theta}\log p(x \mid \theta) + \frac{\partial}{\partial\theta}f(x;\theta)$
	$\mathbb{E}_{x,y}\left[f(y)\right]$	$f(y)\frac{\partial}{\partial\theta}p(x \mid \theta)$
	$\mathbb{E}_{x,y}\left[f(x) + g(y)\right]$	$(f(x) + g(y))\,\frac{\partial}{\partial\theta}\log p(x \mid \theta, w) + g(y)\frac{\partial}{\partial\theta}\log p(y \mid x, \theta)$
	$\mathbb{E}_x\left[f(x, y(\theta))\right]$	$f(x, y(\theta))\frac{\partial}{\partial\theta}\log p(x \mid \theta) + \frac{\partial f}{\partial y}\frac{\partial y}{\partial\theta}$

Terminology

- Denote a SCG by $\mathcal{G} = (\overbrace{\mathcal{V}}^{\text{nodes}}, \overbrace{\mathcal{E}}^{\text{edges}})$
- $v \prec w$ (v "influences" w) $\implies \exists\, (a_i)_{i=1}^{K} \subseteq \mathcal{V}$, with $K \geq 0$, s.t.

$$\{(v, a_1), (a_K, w)\} \cup \{(a_i, a_{i+1})_{i=1}^{K-1}\} \subseteq \mathcal{E}$$

Terminology

- Denote a SCG by $\mathcal{G} = (\overbrace{\mathcal{V}}^{\text{nodes}}, \overbrace{\mathcal{E}}^{\text{edges}})$
- $v \prec w$ (v "influences" w) $\implies \exists\, (a_i)_{i=1}^{K} \subseteq \mathcal{V}$, with $K \geq 0$, s.t.

$$\{(v, a_1), (a_K, w)\} \cup \{(a_i, a_{i+1})_{i=1}^{K-1}\} \subseteq \mathcal{E}$$

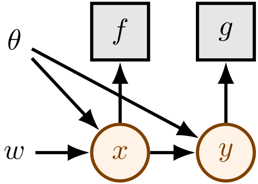
Θ: Input nodes, $\mathcal{D}$: Deterministic nodes
$\mathcal{S}$: Stochastic nodes, $\mathcal{C}$: Cost nodes
$v \prec^D w$: v deterministically influences w
$\text{DEPS}_v =: \{w \in \Theta \cup \mathcal{S} \mid w \prec^D v\}$
$v \in \mathcal{S}$, $p(v \mid \text{DEPS}_v)$: conditional distribution
$v \in \mathcal{D}$, $c(\text{DEPS}_v)$: deterministic function
$\hat{Q}_v = \sum_{\substack{c \in \mathcal{C}, \\ v \prec c}} \hat{c}$: downstream costs of v
$\mathcal{W}_c(\theta) = \{w \mid w \in \mathcal{S}, w \prec c, \theta \prec w\}$
$\hat{v}$: sampled value of v.

Terminology

- Denote a SCG by $\mathcal{G} = (\overbrace{\mathcal{V}}^{\text{nodes}}, \overbrace{\mathcal{E}}^{\text{edges}})$
- $v \prec w$ (v "influences" w) $\implies \exists\, (a_i)_{i=1}^{K} \subseteq \mathcal{V}$, with $K \geq 0$, s.t.

$$\{(v, a_1), (a_K, w)\} \cup \{(a_i, a_{i+1})_{i=1}^{K-1}\} \subseteq \mathcal{E}$$

Θ: Input nodes, $\mathcal{D}$: Deterministic nodes
$\mathcal{S}$: Stochastic nodes, $\mathcal{C}$: Cost nodes
$v \prec^D w$: v deterministically influences w
$\text{DEPS}_v =: \{w \in \Theta \cup \mathcal{S} \mid w \prec^D v\}$
$v \in \mathcal{S}$, $p(v \mid \text{DEPS}_v)$: conditional distribution
$v \in \mathcal{D}$, $c(\text{DEPS}_v)$: deterministic function
$\hat{Q}_v = \sum_{\substack{c \in \mathcal{C},\\ v \prec c}} \hat{c}$: downstream costs of v
$\mathcal{W}_c(\theta) = \{w \mid w \in \mathcal{S}, w \prec c, \theta \prec w\}$
$\hat{v}$: sampled value of v.



$\theta \prec \{x, y, f, g\}$, but only $\theta \prec^D \{x, y\}$

Gradient Estimators for SCGs

(Schulman et al., 2015, Theorem 1)

The set of cost nodes $\mathcal{C}$ are associated with objective

$$\mathcal{L} = \mathbb{E}\left[\sum_{c \in \mathcal{C}} c\right]$$

Gradient Estimators for SCGs

(Schulman et al., 2015, Theorem 1)

The set of cost nodes $\mathcal{C}$ are associated with objective

$$\mathcal{L} = \mathbb{E}\left[\sum_{c \in \mathcal{C}} c\right]$$

Conditions for existence of $\nabla_\theta \mathcal{L}$

Given input node $\theta \in \mathcal{X}$, for all edges (v, w) which satisfy $\theta \prec^D v$ and $\theta \prec^D w$, then the following conditions hold:

- if $w \in \mathcal{D}$, the Jacobian $\frac{\partial w}{\partial v}$ exists
- if $w \in \mathcal{S}$, the derivative $\frac{\partial}{\partial v} p(w \mid \textsc{parents}_w)$ exists

Gradient Estimators for SCGs

(Schulman et al., 2015, Theorem 1)

The set of cost nodes $\mathcal{C}$ are associated with objective

$$\mathcal{L} = \mathbb{E}\left[\sum_{c \in \mathcal{C}} c\right]$$

Conditions for existence of $\nabla_\theta \mathcal{L}$

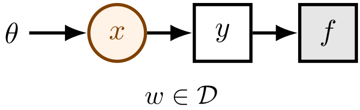
Given input node $\theta \in \mathcal{X}$, for all edges (v, w) which satisfy $\theta \prec^D v$ and $\theta \prec^D w$, then the following conditions hold:

- if $w \in \mathcal{D}$, the Jacobian $\frac{\partial w}{\partial v}$ exists
- if $w \in \mathcal{S}$, the derivative $\frac{\partial}{\partial v} p(w \mid \textsc{parents}_w)$ exists

Note: This does not require all functions in $\mathcal{G}$ to be differentiable.

Gradient Estimators for SCGs

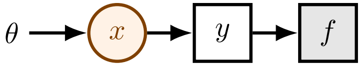
Non-deterministic influence



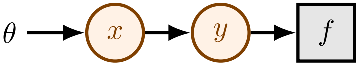
$$\mathcal{L} = \mathbb{E}_x \left[f(y(x)) \right]$$

$$\hat{g} = f(y(x)) \frac{\partial}{\partial \theta} \log p(x \mid \theta)$$

$$\frac{\partial y(x)}{\partial x} \text{ need not exist!}$$



$w \in \mathcal{D}$



$w \in \mathcal{S}$

$$\mathcal{L} = \mathbb{E}_x \left[f(y(x)) \right]$$

$$\hat{g} = f(y(x)) \frac{\partial}{\partial \theta} \log p(x \mid \theta)$$

$\dfrac{\partial y(x)}{\partial x}$ need not exist!

$$\mathcal{L} = \mathbb{E}_{x,y} \left[f(y) \right]$$

$$\hat{g} = f(y) \frac{\partial}{\partial \theta} \log p(x \mid \theta)$$

$p(y \mid x)$ may be unknown!

Gradient Estimators for SCGs

(Schulman et al., 2015, Theorem 1)

Theorem

The gradients of the objective $\mathcal{L} = \mathbb{E}\left[\sum_{c \in \mathcal{C}} c\right]$ *can be computed as*

$$\frac{\partial}{\partial \theta}\mathcal{L} = \mathbb{E}\left[\sum_{c \in \mathcal{C}} c \sum_{\substack{w \prec c, \\ \theta \prec^D w}} \frac{\partial}{\partial \theta} \log p(w \mid \text{DEPS}_w) + \sum_{\substack{c \in \mathcal{C}, \\ \theta \prec^D c}} \frac{\partial}{\partial \theta} c(\text{DEPS}_c)\right] \tag{1}$$

$$= \mathbb{E}\left[\overbrace{\sum_{\substack{w \in \mathcal{S}, \\ \theta \prec^D w}} \left(\frac{\partial}{\partial \theta} \log p(w \mid \text{DEPS}_w)\right) \hat{Q}_w}^{\text{score-function part}} + \overbrace{\sum_{\substack{c \in \mathcal{C} \\ \theta \prec^D c}} \frac{\partial}{\partial \theta} c(\text{DEPS}_c)}^{\text{pathwise derivative term}}\right] \tag{2}$$

given the aforementioned differentiability requirements.

Surrogate loss (SL) functions

(Schulman et al., 2015, Corollary 1)

Corollary

Define a surrogate objective for $\mathcal{L} = \mathbb{E}\left[\sum_{c\in\mathcal{C}} c\right]$ *as*

$$SL\left(\sum_{c\in\mathcal{C}} c\right) := \sum_{\substack{w\in\mathcal{S},\\ \theta\prec^D w}} \log p(w \mid \textit{DEPS}_w)\hat{Q}_w + \sum_{c\in\mathcal{C}} c(\textit{DEPS}_c)$$

$$\left(\nabla_\theta \mathcal{L}\right)_{SL} =: \mathbb{E}\left[\nabla_\theta SL(\cdot)\right], \quad g_{SL} = \nabla_\theta SL(\cdot)$$

Surrogate loss (SL) functions

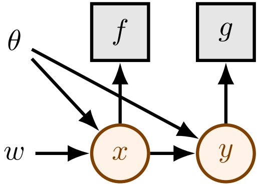
(Schulman et al., 2015, Corollary 1)

Corollary

Define a surrogate objective for $\mathcal{L} = \mathbb{E}\left[\sum_{c \in \mathcal{C}} c\right]$ *as*

$$SL\left(\sum_{c \in \mathcal{C}} c\right) := \sum_{\substack{w \in \mathcal{S}, \\ \theta \prec^D w}} \log p(w \mid \textsc{deps}_w) \hat{Q}_w + \sum_{c \in \mathcal{C}} c(\textsc{deps}_c)$$

$$(\nabla_\theta \mathcal{L})_{SL} =: \mathbb{E}\left[\nabla_\theta SL(\cdot)\right], \quad g_{SL} = \nabla_\theta SL(\cdot)$$



Surrogate loss (SL) functions

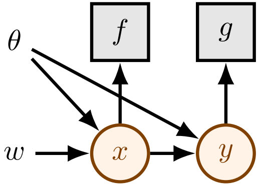
(Schulman et al., 2015, Corollary 1)

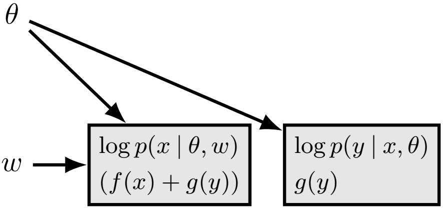
Corollary

Define a surrogate objective for $\mathcal{L} = \mathbb{E}\left[\sum_{c \in \mathcal{C}} c\right]$ *as*

$$SL\left(\sum_{c \in \mathcal{C}} c\right) := \sum_{\substack{w \in \mathcal{S}, \\ \theta \prec^D w}} \log p(w \mid \textsc{deps}_w) \hat{Q}_w + \sum_{c \in \mathcal{C}} c(\textsc{deps}_c)$$

$$\left(\nabla_\theta \mathcal{L}\right)_{SL} =: \mathbb{E}\left[\nabla_\theta SL(\cdot)\right], \quad g_{SL} = \nabla_\theta SL(\cdot)$$





Higher Order Derivatives

Exact gradient

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

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

Higher Order Derivatives

Surrogate loss gradient estimator

$$
\begin{aligned}
\mathrm{SL}\left(f(x;\theta)\right) &= \hat{f}(x;\theta)\log p(x \mid \theta) + f(x;\theta) \\
\left(\boldsymbol{\nabla}_\theta \mathcal{L}\right)_{\mathrm{SL}} &= \mathbb{E}_x\left[\boldsymbol{\nabla}_\theta \mathrm{SL}\left(f(x;\theta)\right)\right] \\
&= \mathbb{E}_x\left[\hat{f}(x;\theta)\boldsymbol{\nabla}_\theta \log p(x \mid \theta) + \boldsymbol{\nabla}_\theta f(x;\theta)\right] \\
&= \mathbb{E}_x[g_{\mathrm{SL}}(x;\theta)] \qquad \text{(unbiased first-order estimate)}
\end{aligned}
$$

Higher Order Derivatives

Surrogate loss gradient estimator

$$\begin{aligned}
\mathrm{SL}\left(f(x;\theta)\right) &= \hat{f}(x;\theta)\log p(x \mid \theta) + f(x;\theta) \\
\left(\nabla_\theta \mathcal{L}\right)_{\mathrm{SL}} &= \mathbb{E}_x\left[\nabla_\theta \mathrm{SL}\left(f(x;\theta)\right)\right] \\
&= \mathbb{E}_x\left[\hat{f}(x;\theta)\nabla_\theta \log p(x \mid \theta) + \nabla_\theta f(x;\theta)\right] \\
&= \mathbb{E}_x[g_{\mathrm{SL}}(x;\theta)] \qquad \text{(unbiased first-order estimate)}
\end{aligned}$$

$$\begin{aligned}
\mathrm{SL}\left(g_{\mathrm{SL}}(x;\theta)\right) &= \hat{g}_{\mathrm{SL}}(x;\theta)\log p(x \mid \theta) + g_{\mathrm{SL}}(x;\theta) \\
\left(\nabla^2_\theta \mathcal{L}\right)_{\mathrm{SL}} &= \mathbb{E}_x\left[\nabla_\theta \mathrm{SL}\left(g_{\mathrm{SL}}(x;\theta)\right)\right] \\
&= \mathbb{E}_x\left[\hat{g}_{\mathrm{SL}}(x;\theta)\nabla_\theta \log p(x \mid \theta) + \nabla_\theta g_{\mathrm{SL}}(x;\theta)\right]
\end{aligned}$$

Higher Order Derivatives

Surrogate loss gradient estimator

$$
\begin{aligned}
\mathrm{SL}\left(f(x;\theta)\right) &= \hat{f}(x;\theta)\log p(x\mid\theta) + f(x;\theta) \\
\left(\boldsymbol{\nabla}_\theta \mathcal{L}\right)_{\mathrm{SL}} &= \mathbb{E}_x\left[\boldsymbol{\nabla}_\theta \mathrm{SL}\left(f(x;\theta)\right)\right] \\
&= \mathbb{E}_x\left[\hat{f}(x;\theta)\boldsymbol{\nabla}_\theta \log p(x\mid\theta) + \boldsymbol{\nabla}_\theta f(x;\theta)\right] \\
&= \mathbb{E}_x[g_{\mathrm{SL}}(x;\theta)] \qquad \text{(unbiased first-order estimate)}
\end{aligned}
$$

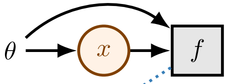
$$
\begin{aligned}
\mathrm{SL}\left(g_{\mathrm{SL}}(x;\theta)\right) &= \hat{g}_{\mathrm{SL}}(x;\theta)\log p(x\mid\theta) + g_{\mathrm{SL}}(x;\theta) \\
\left(\boldsymbol{\nabla}_\theta^2 \mathcal{L}\right)_{\mathrm{SL}} &= \mathbb{E}_x\left[\boldsymbol{\nabla}_\theta \mathrm{SL}\left(g_{\mathrm{SL}}(x;\theta)\right)\right] \\
&= \mathbb{E}_x\left[\hat{g}_{\mathrm{SL}}(x;\theta)\boldsymbol{\nabla}_\theta \log p(x\mid\theta) + \boldsymbol{\nabla}_\theta g_{\mathrm{SL}}(x;\theta)\right]
\end{aligned}
$$

$$
\boldsymbol{\nabla}_\theta g(x;\theta) - \boldsymbol{\nabla}_\theta g_{\mathrm{SL}}(x;\theta) = \boldsymbol{\nabla}_\theta f(x;\theta)\boldsymbol{\nabla}_\theta \log p(x;\theta)
$$

Higher Order Derivatives

Surrogate loss gradient estimator

Surrogate Loss Approach



$\nabla_\theta \mathcal{L}$

$\log(p(x;\theta))\hat{f} + f$ $\xrightarrow{\nabla}$ $g_{\text{SL}} = \hat{f}\nabla_\theta \log p(x;\theta) + \nabla_\theta f$

$\nabla^2_\theta \mathcal{L}$

$\log p(x;\theta)\hat{g}_{\text{SL}} + \hat{f}\nabla_\theta \log p(x;\theta) + \nabla_\theta f$ $\xrightarrow{\nabla}$ $\hat{g}_{\text{SL}}\nabla_\theta \log p(x;\theta) + \hat{f}\nabla^2_\theta \log p(x;\theta) + \nabla_\theta \hat{f}\nabla_\theta \log p(x;\theta) + \nabla^2_\theta f$

$\nabla_\theta \hat{f}\nabla_\theta \log p(x;\theta) = \mathbf{0}$

$\nabla^n_\theta \mathcal{L}$

DiCE Gradient Estimator

(Foerster et al., 2018b, Theorem 1)

MagicBox (⚀) Operator

$\boxdot : \mathcal{S} \to \mathbb{R}$, operates on a set of stochastic nodes $\mathcal{W} \subseteq \mathcal{S}$, satisfying

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

DiCE Gradient Estimator

(Foerster et al., 2018b, Theorem 1)

MagicBox ($\boxdot$) Operator

$\boxdot : \mathcal{S} \to \mathbb{R}$, operates on a set of stochastic nodes $\mathcal{W} \subseteq \mathcal{S}$, satisfying

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

DiCE Objective

Recall $\mathcal{L} = \mathbb{E}\left[\sum_{c \in \mathcal{C}} c\right]$. The associated DiCE objective $\mathcal{L}_{\boxdot}$ as

$$\mathcal{L}_{\boxdot} = \sum_{c \in \mathcal{C}} \boxdot(\mathcal{W}_c) c \tag{3}$$

Claim: $\mathbb{E}\left[\nabla_\theta^n \mathcal{L}_{\boxdot}\right] \rightarrowtail \nabla_\theta^n \mathcal{L} \quad \forall n \in \mathbb{N}$

Comparing DiCE and SL Estimators

$$g_{\text{SL}} = \sum_{c \in \mathcal{C}} \nabla_\theta c + \sum_{w \in \mathcal{S}} \overbrace{\hat{Q}_w \nabla_\theta \log p(w;\theta)}^{\text{total downstream costs } \times \text{ grad logprob}}$$

$$g_{\boxdot} = \sum_{c \in \mathcal{C}} \overbrace{\boxdot(\mathcal{W}_c)}^{\text{retain dependencies}} \left(\nabla_\theta c + \overbrace{c \sum_{w \in \mathcal{W}_c} \log p(w;\theta)}^{\text{sum total of upstream grad logprobs}} \right)$$

Comparing DiCE and SL Estimators

$$g_{\text{SL}} = \sum_{c \in \mathcal{C}} \nabla_\theta c + \sum_{w \in \mathcal{S}} \overbrace{\hat{Q}_w \nabla_\theta \log p(w;\theta)}^{\text{total downstream costs } \times \text{ grad logprob}}$$

$$g_{\boxdot} = \sum_{c \in \mathcal{C}} \overbrace{\boxdot(\mathcal{W}_c)}^{\text{retain dependencies}} \left(\nabla_\theta c + \overbrace{c \sum_{w \in \mathcal{W}_c} \log p(w;\theta)}^{\text{sum total of upstream grad logprobs}} \right)$$

- $\hat{Q}_w$ associated with growing number of downstream costs

Comparing DiCE and SL Estimators

$$g_{\text{SL}} = \sum_{c \in \mathcal{C}} \nabla_\theta c + \sum_{w \in \mathcal{S}} \overbrace{\hat{Q}_w \nabla_\theta \log p(w;\theta)}^{\text{total downstream costs} \times \text{grad logprob}}$$

$$g_{⚁} = \sum_{c \in \mathcal{C}} \overbrace{⚁(\mathcal{W}_c)}^{\text{retain dependencies}} \left(\nabla_\theta c + \overbrace{c \sum_{w \in \mathcal{W}_c} \log p(w;\theta)}^{\text{sum total of upstream grad logprobs}} \right)$$

- $\hat{Q}_w$ associated with growing number of downstream costs
- DiCE retains upstream dependencies *i.e.* $\mathcal{W}_{c^n} = \mathcal{W}_{c^{n+1}}$

Comparing DiCE and SL Estimators

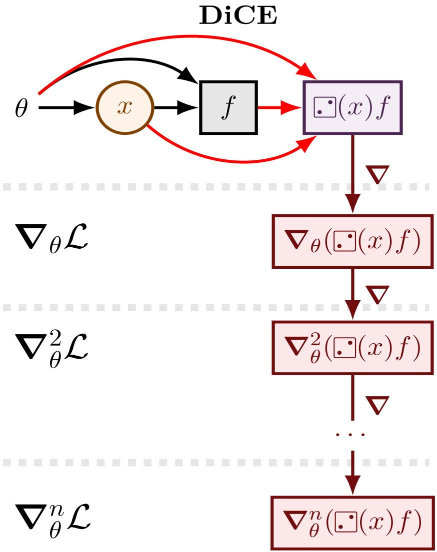
$$g_{\text{SL}} = \sum_{c \in \mathcal{C}} \nabla_\theta c + \sum_{w \in \mathcal{S}} \overbrace{\hat{Q}_w \nabla_\theta \log p(w;\theta)}^{\text{total downstream costs} \times \text{grad logprob}}$$

$$g_{⚁} = \sum_{c \in \mathcal{C}} \overbrace{⚁(\mathcal{W}_c)}^{\text{retain dependencies}} \left(\nabla_\theta c + \overbrace{c \sum_{w \in \mathcal{W}_c} \log p(w;\theta)}^{\text{sum total of upstream grad logprobs}} \right)$$

- $\hat{Q}_w$ associated with growing number of downstream costs
- DiCE retains upstream dependencies *i.e.* $\mathcal{W}_{c^n} = \mathcal{W}_{c^{n+1}}$
- SL requires repeated/recursive application to compute surrogates for higher-order gradients; DiCE - just differentiate!

Higher Order Derivatives

DiCE gradient estimator



DiCE Gradient Estimator

Implementing MagicBox

MagicBox (⚁) Operator

⚁ : $\mathcal{S} \to \mathbb{R}$, operates on a set of stochastic nodes $\mathcal{W} \subseteq \mathcal{S}$, satisfying

- ⚁$(\mathcal{W}) \rightarrowtail 1$ (forward mode)
- ∇_θ⚁$(\mathcal{W}) =$ ⚁$(\mathcal{W}) \sum_{w \in \mathcal{W}} \nabla_\theta \log p(w; \theta)$ (backward mode)

[1] `torch.detach`, `tf.stop_gradient`, `jax.lax.stop_gradient`

DiCE Gradient Estimator

Implementing MagicBox

MagicBox (⚁) Operator

⚁ : $\mathcal{S} \to \mathbb{R}$, operates on a set of stochastic nodes $\mathcal{W} \subseteq \mathcal{S}$, satisfying

- ⚁$(\mathcal{W}) \rightarrowtail 1$ (forward mode)
- ∇_θ⚁$(\mathcal{W}) =$ ⚁$(\mathcal{W}) \sum_{w \in \mathcal{W}} \nabla_\theta \log p(w; \theta)$ (backward mode)

Use the `stop_gradient` [1] trick! ($\texttt{sg}(x) = x$, with $\nabla_x \texttt{sg}(x) = 0$)

[1] `torch.detach`, `tf.stop_gradient`, `jax.lax.stop_gradient`

MagicBox (⚀) Operator

⚀ : $\mathcal{S} \to \mathbb{R}$, operates on a set of stochastic nodes $\mathcal{W} \subseteq \mathcal{S}$, satisfying

- ⚀$(\mathcal{W}) \rightarrowtail 1$ (forward mode)
- ∇_θ⚀$(\mathcal{W})$ = ⚀$(\mathcal{W}) \sum_{w \in \mathcal{W}} \nabla_\theta \log p(w; \theta)$ (backward mode)

Use the `stop_gradient` [1] trick! ($\mathtt{sg}(x) = x$, with $\nabla_x \mathtt{sg}(x) = 0$)

$$\text{⚀}(\mathcal{W}) = \exp\left(\tau - \mathtt{sg}(\tau)\right),$$

$$\tau = \sum_{w \in \mathcal{W}} \log p(w; \theta),$$

[1] `torch.detach`, `tf.stop_gradient`, `jax.lax.stop_gradient`

MagicBox (⚀) Operator

$⚀ : \mathcal{S} \to \mathbb{R}$, operates on a set of stochastic nodes $\mathcal{W} \subseteq \mathcal{S}$, satisfying

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

Use the `stop_gradient` [1] trick! ($\texttt{sg}(x) = x$, with $\nabla_x \texttt{sg}(x) = 0$)

$$⚀(\mathcal{W}) = \exp\left(\tau - \texttt{sg}(\tau)\right), \qquad \tau = \sum_{w \in \mathcal{W}} \log p(w;\theta),$$

$$⚀(\mathcal{W}) = \frac{\tilde{p}}{\texttt{sg}(\tilde{p})}, \qquad \tilde{p} = \prod_{w \in \mathcal{W}} p(w;\theta)$$

[1] `torch.detach`, `tf.stop_gradient`, `jax.lax.stop_gradient`

Application: Multi-Agent Reinforcement learning (RL)

(Foerster et al., 2018a, Learning with opponent-learning awareness (LOLA))

- Learn policy of LOLA agent π_{θ_1} by differentiating through policy gradient update of opponent π_{θ_2}

$$\mathcal{L}^1(\theta_1, \theta_2)_{\text{LOLA}} = \mathbb{E}_{\pi_{\theta_1}, \pi_{\theta_2 + \Delta\theta_2(\theta_1, \theta_2)}}\left[\mathcal{L}^1\right], \text{where}$$
$$\Delta\theta_2(\theta_1, \theta_2) = \alpha_2 \nabla_{\theta_2} \mathbb{E}_{\pi_{\theta_1}, \pi_{\theta_2}}\left[\mathcal{L}^2\right], \quad \mathcal{L}^i = \sum_{t=0}^{T} \gamma^t r_t^i \tag{4}$$

Application: Multi-Agent Reinforcement learning (RL)

(Foerster et al., 2018a, Learning with opponent-learning awareness (LOLA))

- Learn policy of LOLA agent π_{θ_1} by differentiating through policy gradient update of opponent π_{θ_2}

$$\mathcal{L}^1(\theta_1, \theta_2)_{\text{LOLA}} = \mathbb{E}_{\pi_{\theta_1}, \pi_{\theta_2 + \Delta\theta_2(\theta_1, \theta_2)}} \left[\mathcal{L}^1\right], \text{where}$$
$$\Delta\theta_2(\theta_1, \theta_2) = \alpha_2 \nabla_{\theta_2} \mathbb{E}_{\pi_{\theta_1}, \pi_{\theta_2}} \left[\mathcal{L}^2\right], \quad \mathcal{L}^i = \sum_{t=0}^{T} \gamma^t r_t^i \tag{4}$$

- Applying SL trick produces biased estimator

Application: Multi-Agent Reinforcement learning (RL)

(Foerster et al., 2018a, Learning with opponent-learning awareness (LOLA))

- Learn policy of LOLA agent π_{θ_1} by differentiating through policy gradient update of opponent π_{θ_2}

$$\mathcal{L}^1(\theta_1, \theta_2)_{\text{LOLA}} = \mathbb{E}_{\pi_{\theta_1}, \pi_{\theta_2 + \Delta\theta_2(\theta_1, \theta_2)}} \left[\mathcal{L}^1\right], \text{where}$$
$$\Delta\theta_2(\theta_1, \theta_2) = \alpha_2 \nabla_{\theta_2} \mathbb{E}_{\pi_{\theta_1}, \pi_{\theta_2}} \left[\mathcal{L}^2\right], \quad \mathcal{L}^i = \sum_{t=0}^{T} \gamma^t r_t^i \tag{4}$$

- Applying SL trick produces biased estimator
- LOLA-DiCE objective

$$\mathcal{L}^i_{⚀(\theta_1,\theta_2)} = \sum_t ⚀\overbrace{\left(\left\{a_{j\in\{1,2\}}^{t'\leq t}\right\}\right)}^{\text{actions taken by both agents}} \gamma^t r_t^i \quad \forall i \in \{1, 2\} \tag{5}$$

Algorithm 1: LOLA-DiCE: policy gradient update for θ_1

input Policy parameters of the agent, θ_1, and of the opponent, θ_2

Initialize: $\theta_2' \leftarrow \theta_2$

for k in $1 \ldots K$ **do** // inner loop lookahead steps

  Rollout trajectories τ_k under $(\pi_{\theta_1}, \pi_{\theta_2'})$

  Update: $\theta_2' \leftarrow \theta_2' + \alpha_2 \nabla_{\theta_2'} \mathcal{L}^2_{⚀(\theta_1, \theta_2')}$ // lookahead update

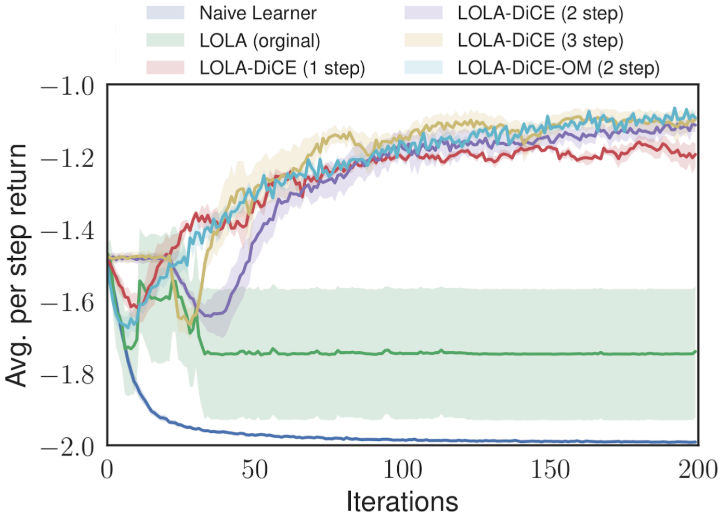
end for

Rollout trajectories τ under $(\pi_{\theta_1}, \pi_{\theta_2'})$.

Update: $\theta_1' \leftarrow \theta_1 + \alpha_1 \nabla_{\theta_1} \mathcal{L}^1_{⚀(\theta_1, \theta_2')}$ // PG update

output θ_1'.

Application: Multi-Agent Reinforcement learning (RL)
LOLA-DiCE



Conclusion

- **Key Idea**: Given a SCG, we can augment it with the DiCE objective in order to generate gradient estimators of any order by differentiating!

Conclusion

- **Key Idea**: Given a SCG, we can augment it with the DiCE objective in order to generate gradient estimators of any order by differentiating!
- **Scope**: Useful for higher order optimization algorithms (Newton's method), mixed derivatives (LOLA), autodiff automates computation of higher order estimators, can be used as a drop-in replacement for manually derived higher-order approximations or biased estimators

Conclusion

- **Key Idea**: Given a SCG, we can augment it with the DiCE objective in order to generate gradient estimators of any order by differentiating!
- **Scope**: Useful for higher order optimization algorithms (Newton's method), mixed derivatives (LOLA), autodiff automates computation of higher order estimators, can be used as a drop-in replacement for manually derived higher-order approximations or biased estimators
- **Limitations**: Variance of $g_{\boxdot}$ might still be high enough to not guarantee convergence, might still need to employ the same variance reduction techniques (eg. control variates). Ideas from REBAR (Tucker et al., 2017), RELAX (Grathwohl et al., 2018) might also be applicable
- Follow-up work (Farquhar et al., 2019) reformulates DiCE for advantage estimation, that discounts the influence of past actions (causal) for estimators of higher order derivatives

Gregory Farquhar, Shimon Whiteson, and Jakob Foerster. Loaded DiCE: Trading off bias and variance in any-order score function gradient estimators for reinforcement learning. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alchë Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32, pages 8151–8162. Curran Associates, Inc., Sep 2019. URL https://proceedings.neurips.cc/paper/2019/file/6fd6b030c6afec018415662d0db43f9d-Paper.pdf.

Jakob Foerster, Richard Y Chen, Maruan Al-Shedivat, Shimon Whiteson, Pieter Abbeel, and Igor Mordatch. Learning with opponent-learning awareness. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pages 122–130. International Foundation for Autonomous Agents and Multiagent Systems, 2018a.

Jakob Foerster, Gregory Farquhar, Maruan Al-Shedivat, Tim Rocktäschel, Eric Xing, and Shimon Whiteson. DiCE: The infinitely differentiable Monte Carlo estimator. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1529–1538, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018b. PMLR. URL http://proceedings.mlr.press/v80/foerster18a.html.

Will Grathwohl, Dami Choi, Yuhuai Wu, Geoff Roeder, and David Duvenaud. Backpropagation through the void: Optimizing control variates for black-box gradient estimation. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=SyzKd1bCW.

References III

Shakir Mohamed, Mihaela Rosca, Michael Figurnov, and Andriy Mnih. Monte carlo gradient estimation in machine learning. *Journal of Machine Learning Research*, 21(132):1–62, 2020. URL http://jmlr.org/papers/v21/19-346.html.

John Schulman, Nicolas Heess, Theophane Weber, and Pieter Abbeel. Gradient estimation using stochastic computation graphs. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28, pages 3528–3536. Curran Associates, Inc., 2015. URL https://proceedings.neurips.cc/paper/2015/file/de03beffeed9da5f3639a621bcab5dd4-Paper.pdf.

George Tucker, Andriy Mnih, Chris J Maddison, John Lawson, and Jascha Sohl-Dickstein. Rebar: Low-variance, unbiased gradient estimates for discrete latent variable models. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30, pages 2627–2636. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/ebd6d2f5d60ff9afaeda1a81fc53e2d0-Paper.pdf.

Thëophane Weber, Nicolas Heess, Lars Buesing, and David Silver. Credit assignment techniques in stochastic computation graphs. In Kamalika Chaudhuri and Masashi Sugiyama, editors, *Proceedings of Machine Learning Research*, volume 89 of *Proceedings of Machine Learning Research*, pages 2650–2660. PMLR, 16–18 Apr 2019. doi: None. URL http://proceedings.mlr.press/v89/weber19a.html.

Appendix: Theorem 1

Proof of(1)(Schulman et al., 2015, Appendix A)

$$\nabla_\theta \mathbb{E}_{\substack{v\in\mathcal{S},\\ v\prec c}}[c] = \nabla_\theta \int \prod_{\substack{v\in\mathcal{S},\\ v\prec c}} p(v \mid \textsc{deps}_v) dv \;\; c(\textsc{deps}_c) \qquad \text{denote } c = c(\textsc{deps}_c)$$

Appendix: Theorem 1

Proof of (1) (Schulman et al., 2015, Appendix A)

$$\nabla_\theta \mathbb{E}_{\substack{v \in \mathcal{S}, \\ v \prec c}}[c] = \nabla_\theta \int \prod_{\substack{v \in \mathcal{S}, \\ v \prec c}} p(v \mid \textsc{deps}_v) dv \quad c(\textsc{deps}_c) \qquad \text{denote } c = c(\textsc{deps}_c)$$

$$= \int \prod_{\substack{v \in \mathcal{S}, \\ v \prec c}} p(v \mid \textsc{deps}_v) dv \left[\nabla_\theta c + c \sum_{w \in \mathcal{W}_c(\theta)} \frac{\nabla_\theta p(w \mid \textsc{deps}_w)}{p(w \mid \textsc{deps}_w)} \right]$$

Appendix: Theorem 1

Proof of (1) (Schulman et al., 2015, Appendix A)

$$\nabla_\theta \mathbb{E}_{\substack{v \in \mathcal{S}, \\ v \prec c}}[c] = \nabla_\theta \int \prod_{\substack{v \in \mathcal{S}, \\ v \prec c}} p(v \mid \text{DEPS}_v) dv \quad c(\text{DEPS}_c) \qquad \text{denote } c = c(\text{DEPS}_c)$$

$$= \int \prod_{\substack{v \in \mathcal{S}, \\ v \prec c}} p(v \mid \text{DEPS}_v) dv \left[\nabla_\theta c + c \sum_{w \in \mathcal{W}_c(\theta)} \frac{\nabla_\theta p(w \mid \text{DEPS}_w)}{p(w \mid \text{DEPS}_w)} \right]$$

$$= \int \prod_{\substack{v \in \mathcal{S}, \\ v \prec c}} p(v \mid \text{DEPS}_v) dv \left[\nabla_\theta c + c \sum_{w \in \mathcal{W}_c(\theta)} \nabla_\theta \log p(w \mid \text{DEPS}_w) \right]$$

Appendix: Theorem 1

Proof of(1)(Schulman et al., 2015, Appendix A)

$$
\begin{aligned}
\nabla_\theta \mathbb{E}_{\substack{v\in\mathcal{S},\\ v\prec c}}[c] &= \nabla_\theta \int \prod_{\substack{v\in\mathcal{S},\\ v\prec c}} p(v \mid \text{DEPS}_v)dv \quad c(\text{DEPS}_c) \qquad \text{denote } c = c(\text{DEPS}_c)\\
&= \int \prod_{\substack{v\in\mathcal{S},\\ v\prec c}} p(v \mid \text{DEPS}_v)dv \left[\nabla_\theta c + c \sum_{w\in\mathcal{W}_c(\theta)} \frac{\nabla_\theta p(w \mid \text{DEPS}_w)}{p(w \mid \text{DEPS}_w)}\right]\\
&= \int \prod_{\substack{v\in\mathcal{S},\\ v\prec c}} p(v \mid \text{DEPS}_v)dv \left[\nabla_\theta c + c \sum_{w\in\mathcal{W}_c(\theta)} \nabla_\theta \log p(w \mid \text{DEPS}_w)\right]\\
&= \mathbb{E}_{\substack{v\in\mathcal{S},\\ v\prec c}}\left[\nabla_\theta c + \boldsymbol{c} \sum_{w\in\mathcal{W}_c(\theta)} \nabla_\theta \log p(w \mid \text{DEPS}_w)\right] = \mathbb{E}_{\substack{v\in\mathcal{S},\\ v\prec c}}[g_{\text{EXACT}}]
\end{aligned}
$$

Appendix: Theorem 1

Proof of (1) (Schulman et al., 2015, Appendix A)

$$\nabla_\theta \mathbb{E}_{\substack{v\in\mathcal{S},\\ v\prec c}}[c] = \nabla_\theta \int \prod_{\substack{v\in\mathcal{S},\\ v\prec c}} p(v \mid \text{DEPS}_v)dv \quad c(\text{DEPS}_c) \qquad \text{denote } c = c(\text{DEPS}_c)$$

$$= \int \prod_{\substack{v\in\mathcal{S},\\ v\prec c}} p(v \mid \text{DEPS}_v)dv \left[\nabla_\theta c + c \sum_{w\in\mathcal{W}_c(\theta)} \frac{\nabla_\theta p(w \mid \text{DEPS}_w)}{p(w \mid \text{DEPS}_w)}\right]$$

$$= \int \prod_{\substack{v\in\mathcal{S},\\ v\prec c}} p(v \mid \text{DEPS}_v)dv \left[\nabla_\theta c + c \sum_{w\in\mathcal{W}_c(\theta)} \nabla_\theta \log p(w \mid \text{DEPS}_w)\right]$$

$$= \mathbb{E}_{\substack{v\in\mathcal{S},\\ v\prec c}} \left[\nabla_\theta c + \boldsymbol{c} \sum_{w\in\mathcal{W}_c(\theta)} \nabla_\theta \log p(w \mid \text{DEPS}_w)\right] = \mathbb{E}_{\substack{v\in\mathcal{S},\\ v\prec c}}[g_{\text{EXACT}}]$$

$$= \mathbb{E}_{\substack{v\in\mathcal{S},\\ v\prec c}} \left[\nabla_\theta c + \hat{\boldsymbol{c}} \sum_{w\in\mathcal{W}_c(\theta)} \nabla_\theta \log p(w \mid \text{DEPS}_w)\right] = \mathbb{E}_{\substack{v\in\mathcal{S},\\ v\prec c}}[g_{\text{SL}}]$$

Appendix: DiCE Gradient Estimator

Proof by Induction

- Consider $\mathcal{L} = \mathbb{E}[c]$, and its (exact) gradient estimator

$$\nabla_\theta \mathcal{L} = \mathbb{E}\left[\nabla_\theta c + c \sum_{w \in \mathcal{W}_c} \nabla_\theta \log p(w;\theta)\right] \tag{6}$$

- Consider $\mathcal{L} = \mathbb{E}\left[c\right]$, and its (exact) gradient estimator

$$\nabla_\theta \mathcal{L} = \mathbb{E}\left[\nabla_\theta c + c \sum_{w \in \mathcal{W}_c} \nabla_\theta \log p(w;\theta)\right] \tag{6}$$

- For each $c \in \mathcal{C}$, define the sequence $\{\mathbb{E}[c^n]\}_{n \in \mathbb{N}}$ inductively as

$$c^0 = c, \quad \mathbb{E}[c^{n+1}] = \nabla_\theta \mathbb{E}[c^n] \implies \mathbb{E}\left[c^n\right] = \nabla_\theta^n \mathbb{E}[c]$$

Taking $\mathcal{L} = \mathbb{E}[c^n]$ in (6), we recover the first order recurrence

$$c^{n+1} = \nabla_\theta c^n + c^n \sum_{w \in \mathcal{W}_{c^n}} \nabla_\theta \log p(w;\theta) \tag{7}$$

Appendix: DiCE Gradient Estimator

Proof by Induction

- Define $c^n_{\boxdot} = c^n \boxdot(\mathcal{W}_{c^n})$. Then $\mathbb{E}[c^n_{\boxdot}] \rightarrowtail \mathbb{E}[c^n] = \nabla^n_\theta \mathbb{E}[c]$
- Noting that (7) implies $\mathcal{W}_{c^n} = \mathcal{W}_{c^{n+1}}$

$$
\begin{aligned}
\nabla_\theta c^n_{\boxdot} &= \nabla_\theta (c^n \boxdot(\mathcal{W}_{c^n})) \\
&= c^n \nabla_\theta \boxdot(\mathcal{W}_{c^n}) + \boxdot(\mathcal{W}_{c^n}) \nabla_\theta c^n \\
&= c^n \boxdot(\mathcal{W}_{c^n}) \left(\sum_{w \in \mathcal{W}_{c^n}} \nabla_\theta \log p(w; \theta) \right) + \boxdot(\mathcal{W}_{c^n}) \nabla_\theta c^n \\
&= \boxdot(\mathcal{W}_{c^n}) \left(\nabla_\theta c^n + c^n \sum_{w \in \mathcal{W}_{c^n}} \nabla_\theta \log p(w; \theta) \right) \\
&= \boxdot(\mathcal{W}_{c^{n+1}}) c^{n+1} = c^{n+1}_{\boxdot} \\
\implies c^n_{\boxdot} &= \nabla^n_\theta c^0_{\boxdot} = \nabla^n_\theta \mathcal{L}_{\boxdot} \qquad \blacksquare
\end{aligned}
$$