Gradient Estimation Using Stochastic Computation Graphs

Yoonho Lee

Department of Computer Science and Engineering
Pohang University of Science and Technology

May 9, 2017

POSTECH

Outline

Outline

POSTECH

Stochastic Gradient Estimation

$$R(\theta) = E_{w \sim P_\theta(dw)}[f(\theta, w)]$$

Approximate $\nabla_\theta R(\theta)$ can be used for:

- Computational Finance
- Reinforcement Learning
- Variational Inference
- Explaining the brain with neural networks

Stochastic Gradient Estimation

$$R(\theta) = E_{w \sim P_\theta(dw)}[f(\theta, w)] = \int_\Omega f(\theta, w) P_\theta(dw)$$

Stochastic Gradient Estimation

$$R(\theta) = E_{w \sim P_\theta(dw)}[f(\theta, w)] = \int_\Omega f(\theta, w) P_\theta(dw)$$

$$= \int_\Omega f(\theta, w) \frac{P_\theta(dw)}{G(dw)} G(dw)$$

Stochastic Gradient Estimation

$$R(\theta) = E_{w \sim P_\theta(dw)}[f(\theta, w)] = \int_\Omega f(\theta, w) P_\theta(dw)$$

$$= \int_\Omega f(\theta, w) \frac{P_\theta(dw)}{G(dw)} G(dw)$$

$$\begin{aligned}
\nabla_\theta R(\theta) &= \nabla_\theta \int_\Omega f(\theta, w) \frac{P_\theta(dw)}{G(dw)} G(dw) \\
&= \int_\Omega \nabla_\theta (f(\theta, w) \frac{P_\theta(dw)}{G(dw)}) G(dw) \\
&= E_{w \sim G(dw)}[\nabla_\theta f(\theta, w) \frac{P_\theta(dw)}{G(dw)} + f(\theta, w) \frac{\nabla_\theta P_\theta(dw)}{G(dw)}]
\end{aligned}$$

Stochastic Gradient Estimation

So far we have

$$\nabla_\theta R(\theta) = E_{w\sim G(dw)}[\nabla_\theta f(\theta, w)\frac{P_\theta(dw)}{G(dw)} + f(\theta, w)\frac{\nabla_\theta P_\theta(dw)}{G(dw)}]$$

Stochastic Gradient Estimation

So far we have

$$\nabla_\theta R(\theta) = E_{w\sim G(dw)}[\nabla_\theta f(\theta, w)\frac{P_\theta(dw)}{G(dw)} + f(\theta, w)\frac{\nabla_\theta P_\theta(dw)}{G(dw)}]$$

Letting $G = P_{\theta_0}$ and evaluating at $\theta = \theta_0$, we get

$$\begin{aligned}\nabla_\theta R(\theta)\Big|_{\theta=\theta_0} &= E_{w\sim P_{\theta_0}(dw)}[\nabla_\theta f(\theta, w)\frac{P_\theta(dw)}{P_{\theta_0}(dw)} + f(\theta, w)\frac{\nabla_\theta P_\theta(dw)}{P_{\theta_0}(dw)}]\Big|_{\theta=\theta_0} \\ &= E_{w\sim P_{\theta_0}(dw)}[\nabla_\theta f(\theta, w) + f(\theta, w)\nabla_\theta \ln P_\theta(dw)]\Big|_{\theta=\theta_0}\end{aligned}$$

Stochastic Gradient Estimation

So far we have

$$R(\theta) = E_{w \sim P_\theta(dw)}[f(\theta, w)]$$

$$\nabla_\theta R(\theta)\bigg|_{\theta=\theta_0} = E_{w \sim P_{\theta_0}(dw)}[\nabla_\theta f(\theta, w) + f(\theta, w) \nabla_\theta \ln P_\theta(dw)]\bigg|_{\theta=\theta_0}$$

POSTECH

Stochastic Gradient Estimation

So far we have

$$R(\theta) = E_{w \sim P_\theta(dw)}[f(\theta, w)]$$

$$\nabla_\theta R(\theta)\Big|_{\theta=\theta_0} = E_{w \sim P_{\theta_0}(dw)}[\nabla_\theta f(\theta, w) + f(\theta, w)\nabla_\theta \ln P_\theta(dw)]\Big|_{\theta=\theta_0}$$

1. Assuming w fully determines f:

$$\nabla_\theta R(\theta)\Big|_{\theta=\theta_0} = E_{w \sim P_{\theta_0}(dw)}[f(w)\nabla_\theta \ln P_\theta(dw)]\Big|_{\theta=\theta_0}$$

2. Assuming $P_\theta(dw)$ is independent of θ:

$$\nabla_\theta R(\theta) = E_{w \sim P(dw)}[\nabla_\theta f(\theta, w)]$$

POSTECH

Stochastic Gradient Estimation

SF vs PD

$$R(\theta) = E_{w \sim P_\theta(dw)}[f(\theta, w)]$$

Score Function (SF) Estimator:

$$\nabla_\theta R(\theta)\Big|_{\theta=\theta_0} = E_{w \sim P_{\theta_0}(dw)}[f(w) \nabla_\theta \ln P_\theta(dw)]\Big|_{\theta=\theta_0}$$

Pathwise Derivative (PD) Estimator:

$$\nabla_\theta R(\theta) = E_{w \sim P(dw)}[\nabla_\theta f(\theta, w)]$$

Stochastic Gradient Estimation

SF vs PD

$$R(\theta) = E_{w \sim P_\theta(dw)}[f(\theta, w)]$$

Score Function (SF) Estimator:

$$\nabla_\theta R(\theta)\Big|_{\theta=\theta_0} = E_{w \sim P_{\theta_0}(dw)}[f(w)\nabla_\theta \ln P_\theta(dw)]\Big|_{\theta=\theta_0}$$

Pathwise Derivative (PD) Estimator:

$$\nabla_\theta R(\theta) = E_{w \sim P(dw)}[\nabla_\theta f(\theta, w)]$$

- SF allows discontinuous f.
- SF requires sample values f; PD requires derivatives f'.
- SF takes derivatives over the probability distribution of w; PD takes derivatives over the function f.
- SF has high variance on high-dimensional w.
- PD has high variance on rough f.

Stochastic Gradient Estimation

Choices for PD estimator

Target	$p(z;\theta)$	Base $p(\epsilon)$	One-liner $g(\epsilon;\theta)$
Exponential	$\exp(-x); x > 0$	$\epsilon \sim [0;1]$	$\ln(1/\epsilon)$
Cauchy	$\frac{1}{\pi(1+x^2)}$	$\epsilon \sim [0;1]$	$\tan(\pi\epsilon)$
Laplace	$\mathcal{L}(0;1) = \exp(-\lvert x\rvert)$	$\epsilon \sim [0;1]$	$\ln(\frac{\epsilon_1}{\epsilon_2})$
Laplace	$\mathcal{L}(\mu;b)$	$\epsilon \sim [0;1]$	$\mu - bsgn(\epsilon)\ln(1-2\lvert\epsilon\rvert)$
Std Gaussian	$\mathcal{N}(0;1)$	$\epsilon \sim [0;1]$	$\sqrt{\ln(\frac{1}{\epsilon_1})}\cos(2\pi\epsilon_2)$
Gaussian	$\mathcal{N}(\mu;RR^\top)$	$\epsilon \sim \mathcal{N}(0;1)$	$\mu + R\epsilon$
Rademacher	$Rad(\frac{1}{2})$	$\epsilon \sim Bern(\frac{1}{2})$	$2\epsilon - 1$
Log-Normal	$\ln\mathcal{N}(\mu;\sigma)$	$\epsilon \sim \mathcal{N}(\mu;\sigma^2)$	$\exp(\epsilon)$
Inv Gamma	$i\mathcal{G}(k;\theta)$	$\epsilon \sim \mathcal{G}(k;\theta^{-1})$	$\frac{1}{\epsilon}$

blog.shakirm.com/2015/10/machine-learning-trick-of-the-day-4-reparameterisation-tricks/

Outline

Gradient Estimation Using Stochastic Computation Graphs (NIPS 2015)

John Schulman, Nicolas Heess, Theophane Weber, Pieter Abbeel

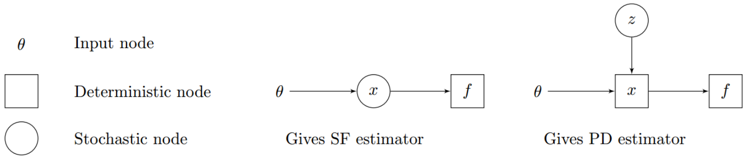
Stochastic Computation Graphs

SF estimator:

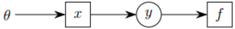
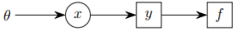
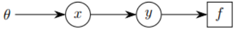
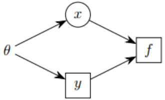
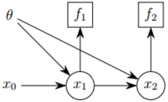
$$\nabla_\theta R(\theta)\Big|_{\theta=\theta_0} = E_{x\sim P_{\theta_0}(dx)}[f(x)\nabla_\theta \ln P_\theta(dx)]\Big|_{\theta=\theta_0}$$

PD estimator:

$$\nabla_\theta R(\theta) = E_{z\sim P(dz)}[\nabla_\theta f(x(\theta,z))]$$



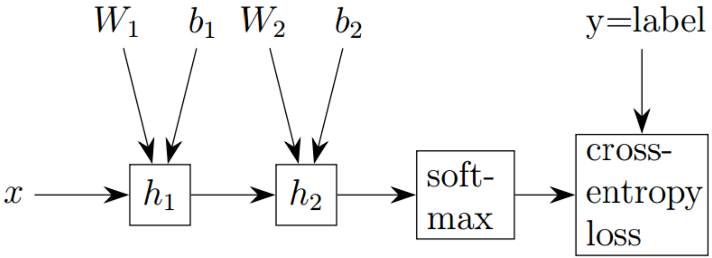
Stochastic Computation Graphs

	Stochastic Computation Graph	Objective	Gradient Estimator
(1)		$\mathbb{E}_y[f(y)]$	$\frac{\partial x}{\partial \theta}\frac{\partial}{\partial x}\log p(y \mid x) f(y)$
(2)		$\mathbb{E}_x[f(y(x))]$	$\frac{\partial}{\partial \theta}\log p(x \mid \theta) f(y(x))$
(3)		$\mathbb{E}_{x,y}[f(y)]$	$\frac{\partial}{\partial \theta}\log p(x \mid \theta) f(y)$
(4)		$\mathbb{E}_x[f(x, y(\theta))]$	$\frac{\partial}{\partial \theta}\log p(x \mid \theta) f(x, y(\theta)) + \frac{\partial y}{\partial \theta}\frac{\partial f}{\partial y}$
(5)		$\mathbb{E}_{x_1,x_2}[f_1(x_1) + f_2(x_2)]$	$\frac{\partial}{\partial \theta}\log p(x_1 \mid \theta, x_0)(f_1(x_1) + f_2(x_2)) + \frac{\partial}{\partial \theta}\log p(x_2 \mid \theta, x_1) f_2(x_2)$

- Stochastic computation graphs are a design choice rather than an intrinsic property of a problem
- Stochastic nodes 'block' gradient flow

Stochastic Computation Graphs

Neural Networks



- Neural Networks are a special case of stochastic computation graphs

Stochastic Computation Graphs

Formal Definition

Definition 1 (Stochastic Computation Graph)**.** *A directed, acyclic graph, with three types of nodes:*

1. *Input nodes, which are set externally, including the parameters we differentiate with respect to.*
2. *Deterministic nodes, which are functions of their parents.*
3. *Stochastic nodes, which are distributed conditionally on their parents.*

Each parent v of a non-input node w is connected to it by a directed edge (v, w).

Stochastic Computation Graphs

Deriving Unbiased Estimators

Condition 1. (Differentiability Requirements) Given input node $\theta \in \Theta$, for all edges (v, w) which satisfy $\theta \prec^D v$ and $\theta \prec^D w$, the following condition holds: if w is deterministic, $\frac{\partial w}{\partial v}$ exists, and if w is stochastic, $\frac{\partial}{\partial v} p(w|\text{PARENTS}_w)$ exists.

Theorem 1. Suppose $\theta \in \Theta$ satisfies Condition 1. The following equations hold:

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

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

Stochastic Computation Graphs

Surrogate loss functions

We have this equation from Theorem 1:

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

Note that by defining

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

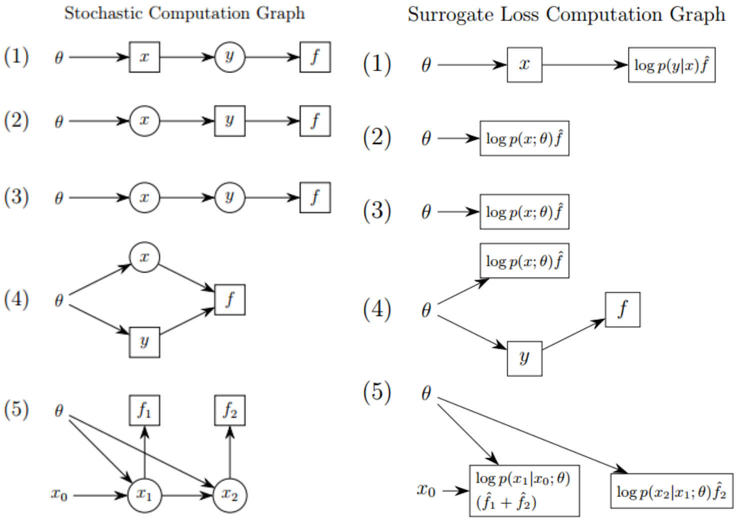
we have

$$\frac{\partial}{\partial\theta}\mathbb{E}\left[\sum_{c\in C} c\right] = \mathbb{E}\left[\frac{\partial}{\partial\theta} L(\Theta, S)\right]$$

POSTECH

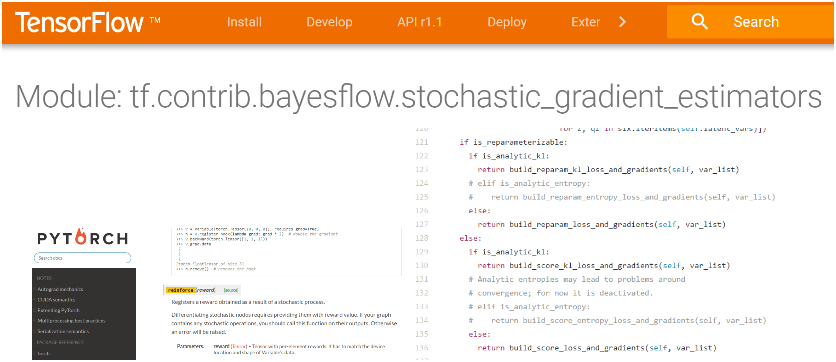
Stochastic Computation Graphs

Surrogate loss functions



Stochastic Computation Graphs

Implementations



- Making stochastic computation graphs with neural network packages is easy

---

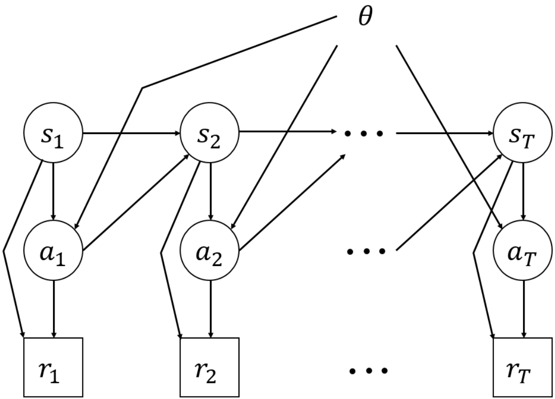
tensorflow.org/api_docs/python/tf/contrib/bayesflow/stochastic_gradient_estimators

github.com/pytorch/pytorch/blob/6a69f7007b37bb36d8a712cdf5cebe3ee0cc1cc8/torch/autograd/variable.py

github.com/blei-lab/edward/blob/master/edward/inferences/klqp.py

Outline

MDP



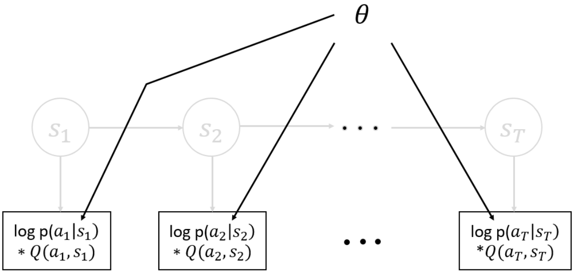
- The MDP formulation itself is a stochastic computation graph

Policy Gradient

Surrogate Loss

Theorem 1 (Policy Gradient). For any MDP, in either the average-reward or start-state formulations,

$$\frac{\partial \rho}{\partial \theta} = \sum_s d^\pi(s) \sum_a \frac{\partial \pi(s,a)}{\partial \theta} Q^\pi(s,a). \tag{2}$$ [1]



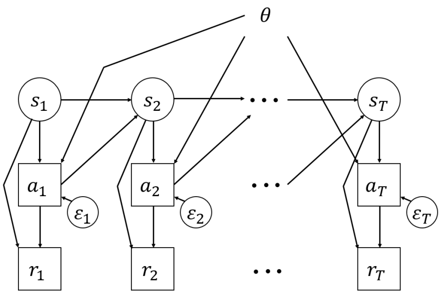
- The policy gradient algorithm is Theorem 1 applied to the MDP graph

[1] Richard S Sutton et al. "Policy Gradient Methods for Reinforcement Learning with Function Approximation". In: *NIPS* (1999).

Stochastic Value Gradient
SVG(0)

Learn Q_ϕ, and update[2]

$$\Delta\theta \propto \nabla_\theta \sum_{t=1}^{T} Q_\phi(s_t, \pi_\theta(s_t, \epsilon_t))$$



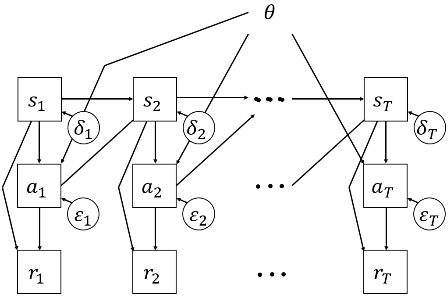
[2] Nicolas Heess et al. "Learning Continuous Control Policies by Stochastic Value Gradients". In: *NIPS* (2015).

Stochastic Value Gradient

SVG(1)

Learn V_ϕ and dynamics model f such that $s_{t+1} = f(s_t, a_t) + \delta_t$.
Given transition (s_t, a_t, s_{t+1}), infer noise δ_t.[3]

$$\Delta\theta \propto \nabla_\theta \sum_{t=1}^{T} r_t + \gamma V_\phi(f(s_t, a_t) + \delta_t)$$

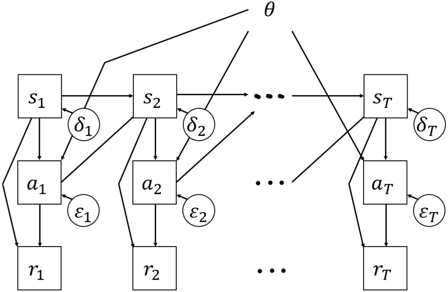


[3] Nicolas Heess et al. "Learning Continuous Control Policies by Stochastic Value Gradients". In: *NIPS* (2015).

Stochastic Value Gradient

SVG(∞)

Learn f, infer all noise variables δ_t. Differentiate through whole graph.[4]



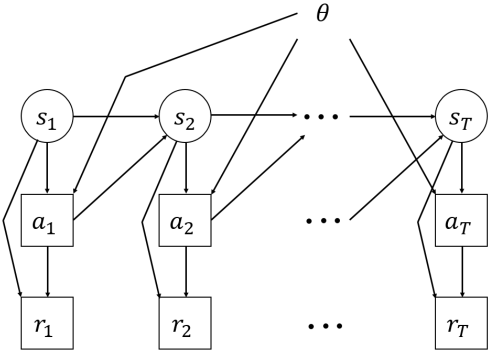
[4] Nicolas Heess et al. "Learning Continuous Control Policies by Stochastic Value Gradients". In: *NIPS* (2015).

Deterministic Policy Gradient

Theorem 1 (Deterministic Policy Gradient Theorem). *Suppose that the MDP satisfies conditions A.1 (see Appendix; these imply that $\nabla_\theta \mu_\theta(s)$ and $\nabla_a Q^\mu(s,a)$ exist and that the deterministic policy gradient exists. Then,*

$$\begin{aligned} \nabla_\theta J(\mu_\theta) &= \int_{\mathcal{S}} \rho^\mu(s) \nabla_\theta \mu_\theta(s) \, \nabla_a Q^\mu(s,a)|_{a=\mu_\theta(s)} \, \mathrm{d}s \\ &= \mathbb{E}_{s\sim\rho^\mu}\left[\nabla_\theta \mu_\theta(s) \, \nabla_a Q^\mu(s,a)|_{a=\mu_\theta(s)}\right] \quad (9) \end{aligned}$$

[5]



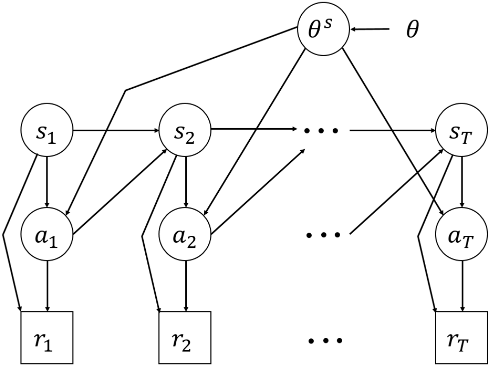
[5] David Silver et al. "Deterministic Policy Gradient Algorithms". In: *ICML* (2014).

Evolution Strategies

Algorithm 1 Evolution Strategies

Input: Learning rate α, noise standard deviation σ, initial policy parameters θ_0
for $t = 0, 1, 2, \ldots$ **do**
  Sample $\epsilon_1, \ldots \epsilon_n \sim \mathcal{N}(0, I)$
  Compute returns $F_i = F(\theta_t + \sigma\epsilon_i)$ for $i = 1, \ldots, n$
  Set $\theta_{t+1} \leftarrow \theta_t + \alpha \frac{1}{n\sigma} \sum_{i=1}^{n} F_i \epsilon_i$
end for

[6]



[6] Tim Salimans et al. “Evolution Strategies as a Scalable Alternative to Reinforcement Learning”. In: arXiv (2017).

Outline

POSTECH

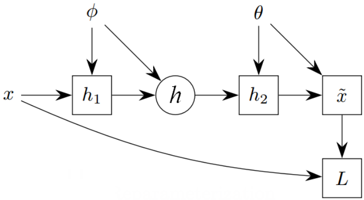
Neural Variational Inference

$$\nabla_\theta \mathcal{L}(x) \approx \frac{1}{n}\sum_{i=1}^{n} \nabla_\theta \log P_\theta(x, h^{(i)})$$

and

$$\nabla_\phi \mathcal{L}(x) \approx \frac{1}{n}\sum_{i=1}^{n} (\log P_\theta(x, h^{(i)}) - \log Q_\phi(h^{(i)}|x)) \times \nabla_\phi \log Q_\phi(h^{(i)}|x).$$

[7]



Where

$$L(x, \theta, \phi) = \mathbb{E}_{h \sim Q(h|x)}[\log P_\theta(x, h) - \log Q_\phi(h|x)]$$

- Score function estimator

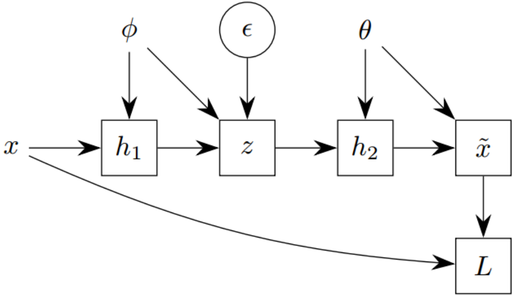
[7] Andriy Mnih and Karol Gregor. "Neural Variational Inference and Learning in Belief Networks". In: ICML (2014).

Variational Autoencoder

We apply this technique to the variational lower bound (eq. (2)), yielding our generic Stochastic Gradient Variational Bayes (SGVB) estimator $\widetilde{\mathcal{L}}^A(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)}) \simeq \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)})$:

$$\widetilde{\mathcal{L}}^A(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)}) = \frac{1}{L}\sum_{l=1}^{L} \log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}, \mathbf{z}^{(i,l)}) - \log q_{\boldsymbol{\phi}}(\mathbf{z}^{(i,l)}|\mathbf{x}^{(i)})$$

$$\text{where} \quad \mathbf{z}^{(i,l)} = g_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}^{(i,l)}, \mathbf{x}^{(i)}) \quad \text{and} \quad \boldsymbol{\epsilon}^{(l)} \sim p(\boldsymbol{\epsilon}) \qquad (6)$$ [8]

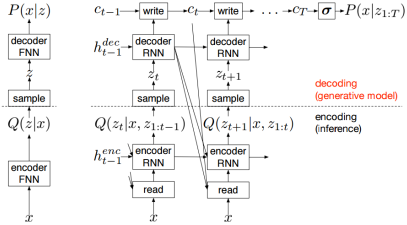


- Pathwise derivative estimator applied to the exact same graph as NVIL

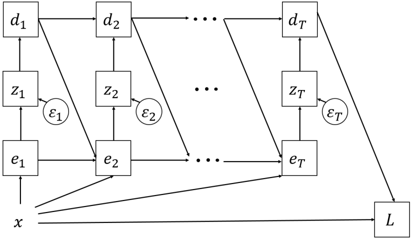
[8] Diederik P Kingma and Max Welling. "Auto-Encoding Variational Bayes". In: *ICLR* (2014).

POSTECH

DRAW



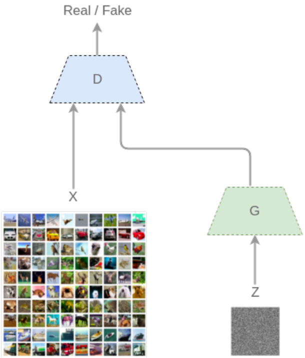
9



- Sequential version of VAE
- Same graph as SVG(∞) with (known) deterministic dynamics(e=state, z=action, L=reward)
- Similarly, VAE can be seen as SVG(∞) with 1 timestep and deterministic dynamics

[9] Karol Gregor et al. “DRAW: A Recurrent Neural Network For Image Generation”. In: *ICML* (2015).

GAN



10

- Connection to SVG(0) has been established in [10]
- Actor has no access to state. Observation is a random choice between real or fake image. Reward is 1 if real, 0 if fake. D learns value function of observation. G's learning signal is D.

[10] David Pfau and Oriol Vinyals. "Connecting Generative Adversarial Networks and Actor-Critic Methods". In: (2016).

Bayes by backprop

$$\frac{\partial}{\partial\theta}\mathbb{E}_{q(\mathbf{w}|\theta)}[f(\mathbf{w},\theta)] = \mathbb{E}_{q(\epsilon)}\left[\frac{\partial f(\mathbf{w},\theta)}{\partial \mathbf{w}}\frac{\partial \mathbf{w}}{\partial\theta} + \frac{\partial f(\mathbf{w},\theta)}{\partial\theta}\right]$$[11]

and our cost function f is:

$$\begin{aligned} f(D,\theta) &= \mathbb{E}_{w\sim q(w|\theta)}[f(w,\theta)] \\ &= \mathbb{E}_{w\sim q(w|\theta)}[-\log p(D|w) + \log q(w|\theta) - \log p(w)] \end{aligned}$$

- From the point of view of stochastic computation graphs, weights and activations are not different
- Estimate ELBO gradient of activation with PD estimator: VAE
- Estimate ELBO gradient of weight with PD estimator: Bayes by Backprop

[11] Charles Blundell et al. "Weight Uncertainty in Neural Networks". In: *ICML* (2015).

Summary

- Stochastic computation graphs explain many different algorithms with one gradient estimator(Theorem 1)
- Stochastic computation graphs give us insight into the behavior of estimators
- Stochastic computation graphs reveal equivalencies:
  - NVIL[7]=Policy gradient[12]
  - VAE[6]/DRAW[4]=SVG(∞)[5]
  - GAN[3]=SVG(0)[5]

References I

[1] Charles Blundell et al. "Weight Uncertainty in Neural Networks". In: *ICML* (2015).

[2] Michael Fu. "Stochastic Gradient Estimation". In: (2005).

[3] Ian J. Goodfellow et al. "Generative Adversarial Networks". In: *NIPS* (2014).

[4] Karol Gregor et al. "DRAW: A Recurrent Neural Network For Image Generation". In: *ICML* (2015).

[5] Nicolas Heess et al. "Learning Continuous Control Policies by Stochastic Value Gradients". In: *NIPS* (2015).

[6] Diederik P Kingma and Max Welling. "Auto-Encoding Variational Bayes". In: *ICLR* (2014).

[7] Andriy Mnih and Karol Gregor. "Neural Variational Inference and Learning in Belief Networks". In: *ICML* (2014).

References II

[8] David Pfau and Oriol Vinyals. "Connecting Generative Adversarial Networks and Actor-Critic Methods". In: (2016).

[9] Tim Salimans et al. "Evolution Strategies as a Scalable Alternative to Reinforcement Learning". In: *arXiv* (2017).

[10] John Schulman et al. "Gradient Estimation Using Stochastic Computation Graphs". In: *NIPS* (2015).

[11] David Silver et al. "Deterministic Policy Gradient Algorithms". In: *ICML* (2014).

[12] Richard S Sutton et al. "Policy Gradient Methods for Reinforcement Learning with Function Approximation". In: *NIPS* (1999).

Thank You