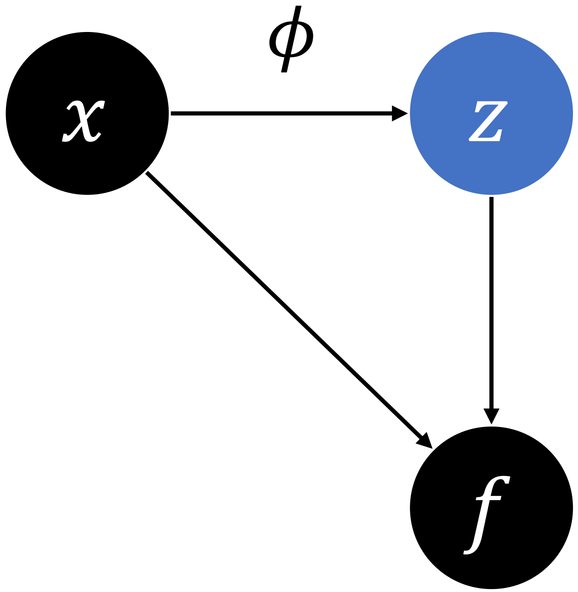
Evaluating the Variance of Likelihood-Ratio Gradient Estimators

Seiya Tokui [1,2] Issei Sato [2,3]

[1] Preferred Networks [2] The University of Tokyo [3] RIKEN

ICML 2017 @ Sydney

Task: Gradient estimation for stochastic computational graph



Computational Graph

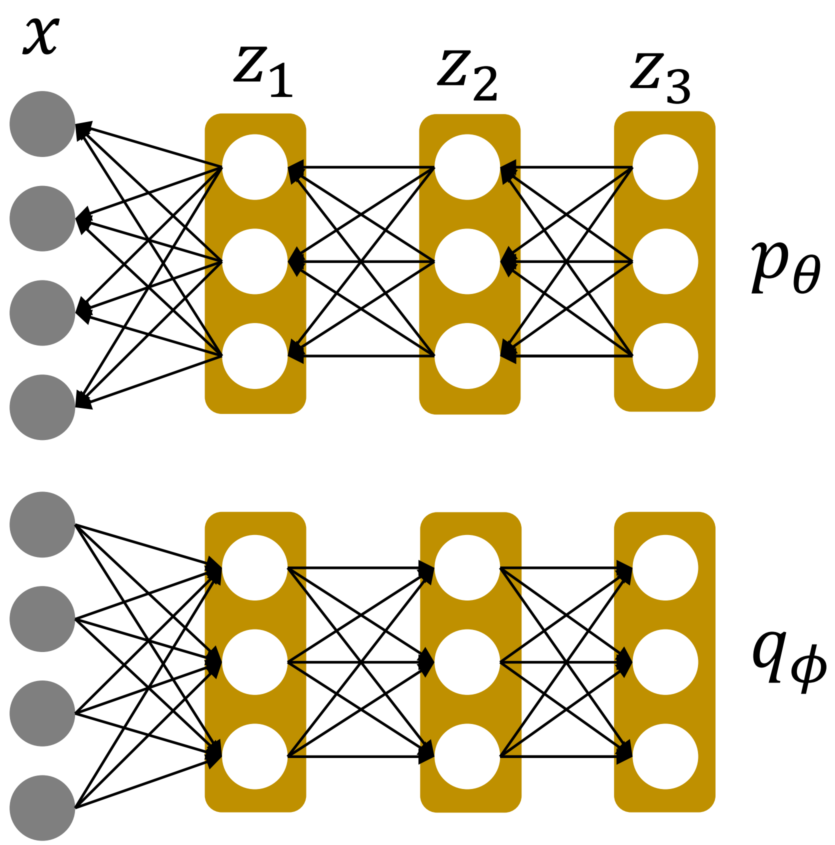
Want to compute the following gradient:

$$\nabla_\phi \mathbb{E}_{q_\phi(z|x)} f(x, z)$$

If

- No stochasiticity in z
  (q is a delta distribution)
  → use *backprop*
- z is stochastic
  (stochastic computational graph)
  → **need more techniques**

Example: Variational inference in deep directed graphical models



Graphical Models

Generative model

$$p_\theta(x,z) = \underline{p_\theta(x|z_1)}\ \underline{p_\theta(z_1|z_2)}\ \underline{p_\theta(z_2|z_3)}\ p_\theta(z_3)$$

Each factor is written by a NN

Approximate posterior

$$q_\phi(z|x) = \underline{q_\phi(z_1|x)}\underline{q_\phi(z_2|z_1)}\underline{q_\phi(z_3|z_2)}$$

Each factor is written by a NN

Objective function (variational bound)

$$\mathcal{L}(\phi,\theta) \coloneqq \mathbb{E}_{q_\phi(z|x)} \log \frac{p_\theta(x,z)}{q_\phi(z|x)} = f(x,z)$$

We want to compute $\nabla_\phi \mathcal{L}$ to optimize $\mathcal{L}$ with a gradient method.

Overview of unbiased estimators

Likelihood-ratio estimators

- ✓ z can be continuous or discrete
- ✓ f can be non-continuous
- ✓ Tend to have high variance
- ✓ Many (heuristic) techniques to reduce the variance exist

Reparameterization trick

- ✓ z must be continuous
- ✓ f must be differentiable
- ✓ Tend to have low variance in practice (but not guaranteed)

Our finding: when there are M random variables, also likelihood-ratio estimators can be formulated with reparameterization for $M-1$ variables
→ ***unified framework of gradient estimators***

A unified framework of gradient estimators

Let $z = (z_1, \dots, z_M)$ and $q_\phi(z|x) = \prod_{i=1}^{M} q_{\phi_i}(z_i|\underline{\mathrm{pa}_i})$.

The set of parents of z_i

Suppose we have a *reparameterization formula:*

$$z_i \sim q_{\phi_i}(z_i|\mathrm{pa}_i) \iff z_i = g_{\phi_i}(\mathrm{pa}_i, \epsilon_i), \quad \underline{\epsilon_i} \sim p(\epsilon_i)$$

Noise variable that generates z_i

Exchange ∇ and $\mathbb{E}$ *partially* for each i :

Differentiable even if g is non-continuous ($\Leftarrow z_i$ is discrete)

$$\nabla_{\phi_i} \mathbb{E}_{q_\phi(z|x)} f(x, z) = \nabla_{\phi_i} \mathbb{E}_\epsilon f\left(x, g_\phi(x, \epsilon)\right) = \mathbb{E}_{\epsilon_{\setminus i}} \nabla_{\phi_i} \underline{\mathbb{E}_{\epsilon_i} f(x, g_\phi(x, \epsilon))}$$

Reparameterization

[Williams, 1992][Kingma & Welling, 2014]
[Rezende+, 2014][Titsias & Lázaro-Gredilla, 2014]

Local marginalization

[Titsias & Lázaro-Gredilla, 2015]

A unified framework of gradient estimators

$$\nabla_{\phi_i}\mathbb{E}_{q_\phi(z|x)}f(x,z) = \mathbb{E}_{\epsilon_{\setminus i}}\underbrace{\nabla_{\phi_i}\mathbb{E}_{\epsilon_i}f(x,g_\phi(x,\epsilon))}_{\text{Local gradient}}$$

Each method differs in how to estimate the local gradient.

- Likelihood-ratio estimator: use log derivative trick
- Reparameterization estimator: use reparameterization trick
- *Optimal estimator:* exactly (or numerically) compute the inner expectation

Likelihood-ratio estimator under the framework

$$\nabla_{\phi_i}\mathbb{E}_{\epsilon_i} f(x,z) = \mathbb{E}_{\epsilon_i}\big(f(x,z) - \underbrace{b_i(x,\epsilon)}_{\text{Baseline}}\big)\nabla_{\phi_i}\log q_{\phi_i}(z_i|\mathrm{pa}_i) + \underbrace{C_i(x,\epsilon_{\setminus i})}_{\text{Residual}}$$

Baseline	Definition	Example
Constant	b_i is a constant of x and ϵ. $C_i = 0$.	Running average of sampled f
Independent	$b_i(x,\epsilon_{\setminus i})$ is a constant of ϵ_i. $C_i = 0$.	Input-dependent baseline Local signal [Mnih & Gregor, 2014]
Linear	$b_i(x,\epsilon)$ is linear against z_i.	MuProp [Gu+, 2016]
Fully-informed	$b_i(x,\epsilon)$ may be nonlinear against z_i.	The optimal estimator (general)

Reparameterization estimator under the framework

Apply the reparameterization trick to the local gradient:

$$\nabla_{\phi_i} \mathbb{E}_{\epsilon_i} f\left(x, g_\phi(x, \epsilon)\right) = \mathbb{E}_{\epsilon_i} \nabla_{\phi_i} f(x, g_\phi(x, \epsilon))$$

- If g_ϕ is not continuous, the above equation does not hold (in other words, Monte Carlo estimation of the right hand side has *infinite variance*).
- Otherwise, the reparameterization trick can be used.

Optimal estimator under the framework

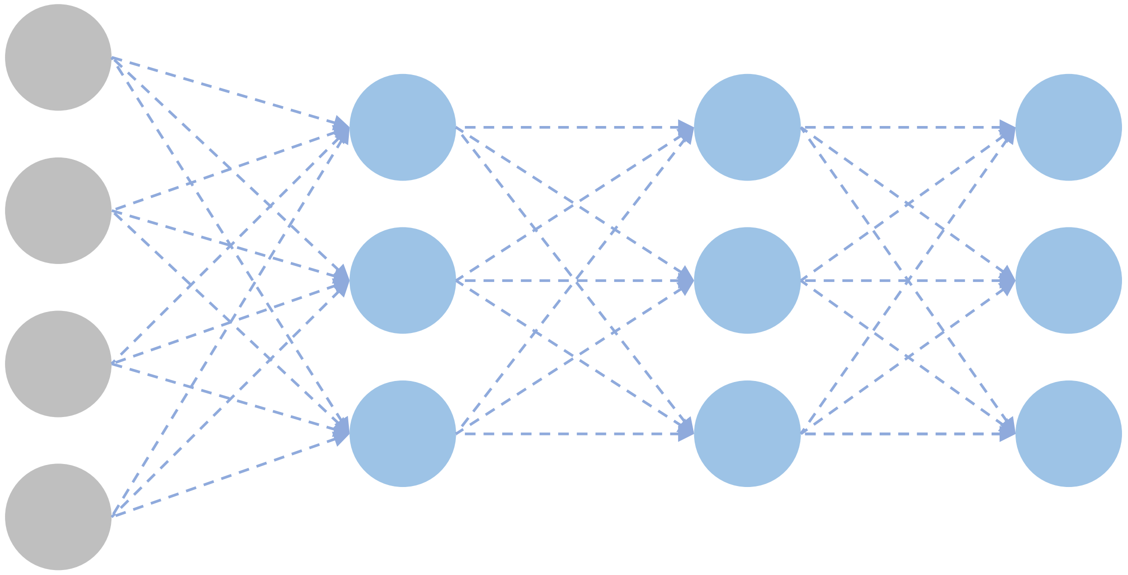
(slow for use in every iteration, yet useful for evaluation purpose)

Exactly (or numerically) compute the local gradient:

$$\nabla_{\phi_i} \mathbb{E}_{\epsilon_i} f\left(x, g_\phi(x, \epsilon)\right) = \sum_{z_i} f(x, z) \nabla_{\phi_i} q_{\phi_i}(z_i | \mathrm{pa}_i)$$

Implementation (Bernoulli case):

- Sample ϵ and compute $z = g_\phi(x, \epsilon)$ and $f(x, z)$
- For each i:
  - Flip z_i and resample descendants of z_i with fixed $\epsilon_{\backslash i}$
  - Recompute $f(x, z)$
  - Compute the local gradient (the above equation)



Optimal estimator under the framework
(slow for use in every iteration, yet useful for evaluation purpose)

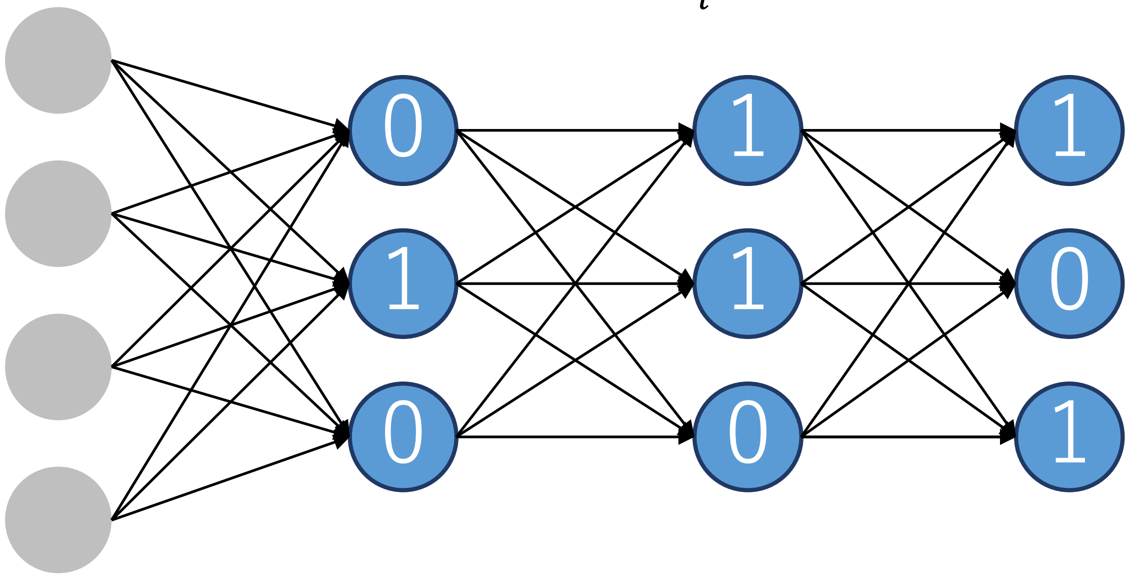
Exactly (or numerically) compute the local gradient:

$$\nabla_{\phi_i} \mathbb{E}_{\epsilon_i} f\left(x, g_\phi(x, \epsilon)\right) = \sum_{z_i} f(x, z) \nabla_{\phi_i} q_{\phi_i}(z_i | \mathrm{pa}_i)$$

Implementation (Bernoulli case):

- Sample ϵ and compute $z = g_\phi(x, \epsilon)$ and $f(x, z)$
- For each i:
  - Flip z_i and resample descendants of z_i with fixed $\epsilon_{\setminus i}$
  - Recompute $f(x, z)$
  - Compute the local gradient (the above equation)

All the ϵ_i's are fixed.



Optimal estimator under the framework
(slow for use in every iteration, yet useful for evaluation purpose)

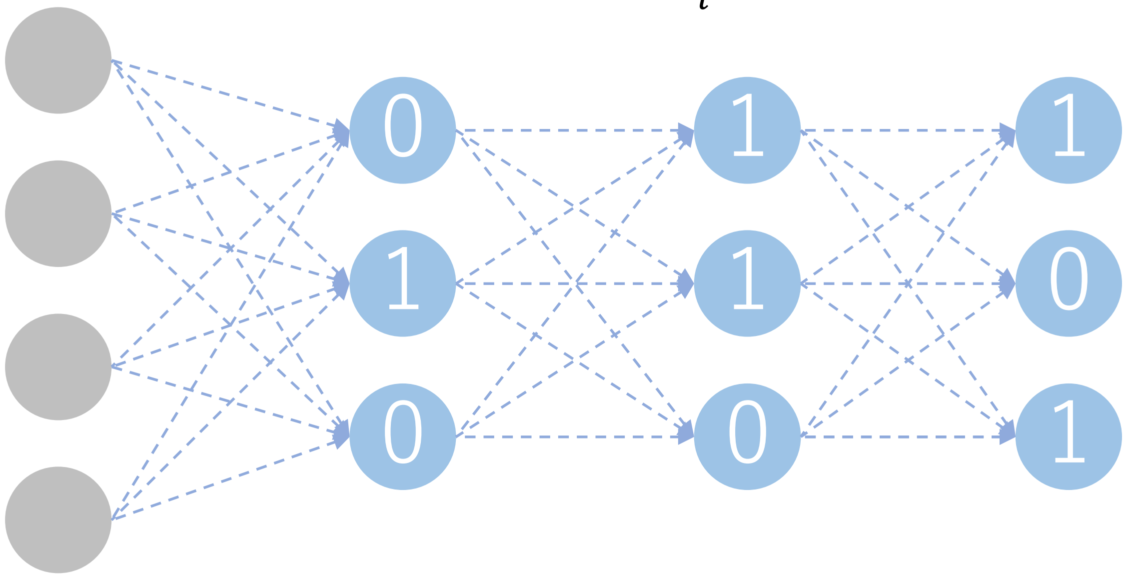
Exactly (or numerically) compute the local gradient:

$$\nabla_{\phi_i} \mathbb{E}_{\epsilon_i} f\left(x, g_\phi(x, \epsilon)\right) = \sum_{z_i} f(x, z) \nabla_{\phi_i} q_{\phi_i}(z_i | \mathrm{pa}_i)$$

Implementation (Bernoulli case):

- Sample ϵ and compute $z = g_\phi(x, \epsilon)$ and $f(x, z)$
- For each i:
  - Flip z_i and resample descendants of z_i with fixed $\epsilon_{\setminus i}$
  - Recompute $f(x, z)$
  - Compute the local gradient (the above equation)

All the ϵ_i's are fixed.



Optimal estimator under the framework
(slow for use in every iteration, yet useful for evaluation purpose)

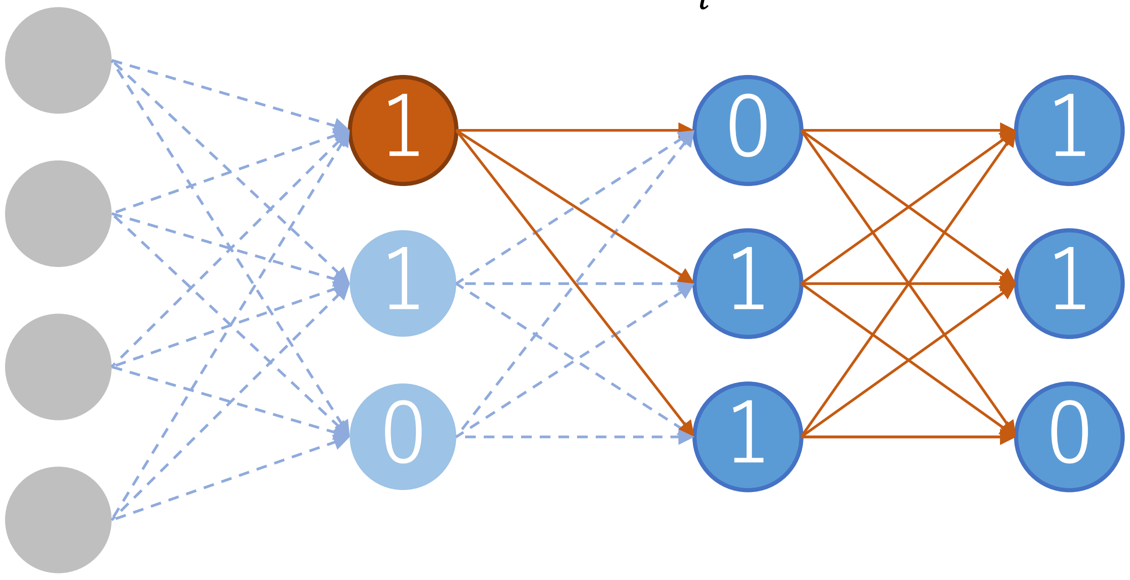
Exactly (or numerically) compute the local gradient:

$$\nabla_{\phi_i}\mathbb{E}_{\epsilon_i} f\left(x, g_\phi(x,\epsilon)\right) = \sum_{z_i} f(x,z)\nabla_{\phi_i} q_{\phi_i}(z_i|\mathrm{pa}_i)$$

Implementation (Bernoulli case):

- Sample ϵ and compute $z = g_\phi(x,\epsilon)$ and $f(x,z)$
- For each i:
  - Flip z_i and resample descendants of z_i with fixed $\epsilon_{\setminus i}$
  - Recompute $f(x,z)$
  - Compute the local gradient (the above equation)

All the ϵ_i's are fixed.



Optimal estimator under the framework

(slow for use in every iteration, yet useful for evaluation purpose)

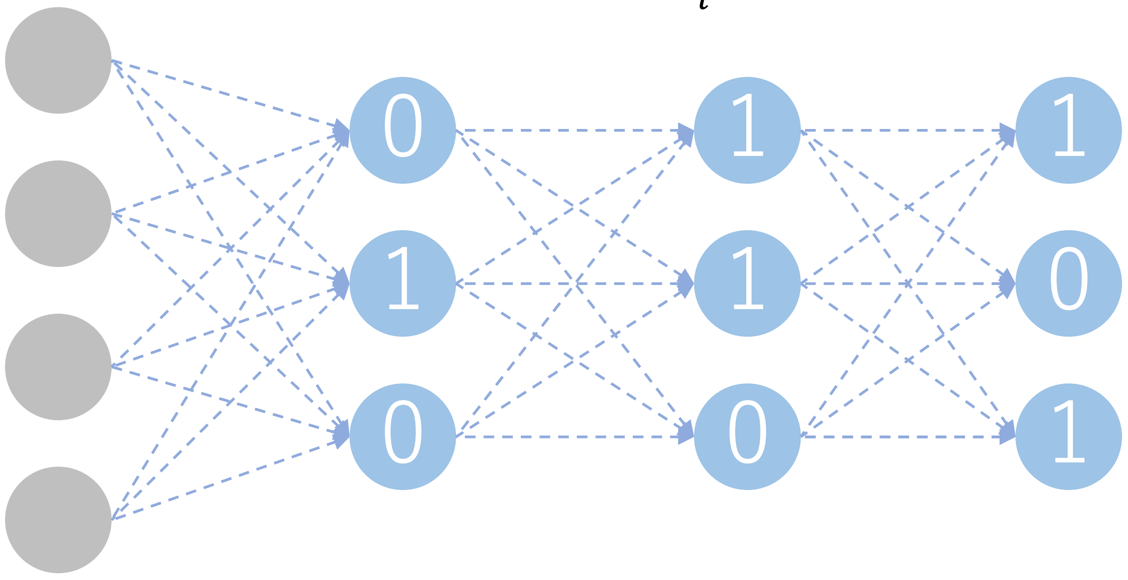
Exactly (or numerically) compute the local gradient:

$$\nabla_{\phi_i}\mathbb{E}_{\epsilon_i} f\left(x, g_\phi(x,\epsilon)\right) = \sum_{z_i} f(x,z)\nabla_{\phi_i} q_{\phi_i}(z_i|\mathrm{pa}_i)$$

Implementation (Bernoulli case):

- Sample ϵ and compute $z = g_\phi(x,\epsilon)$ and $f(x,z)$
- For each i:
  - Flip z_i and resample descendants of z_i with fixed $\epsilon_{\setminus i}$
  - Recompute $f(x,z)$
  - Compute the local gradient (the above equation)

All the ϵ_i's are fixed.



Optimal estimator under the framework
(slow for use in every iteration, yet useful for evaluation purpose)

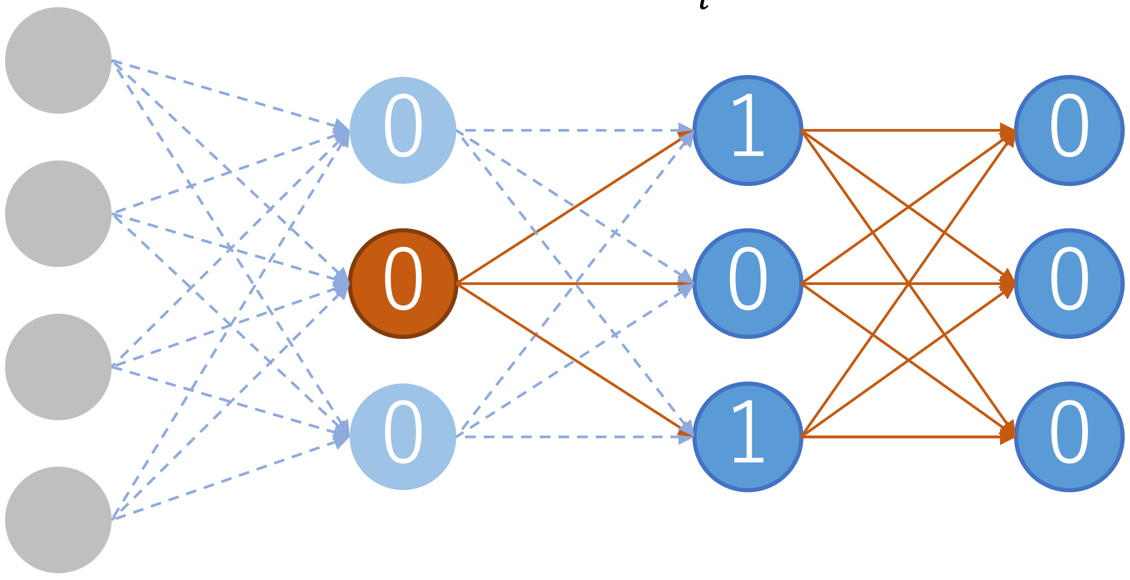
Exactly (or numerically) compute the local gradient:

$$\nabla_{\phi_i} \mathbb{E}_{\epsilon_i} f\left(x, g_\phi(x, \epsilon)\right) = \sum_{z_i} f(x, z) \nabla_{\phi_i} q_{\phi_i}(z_i | \mathrm{pa}_i)$$

Implementation (Bernoulli case):

- Sample ϵ and compute $z = g_\phi(x, \epsilon)$ and $f(x, z)$
- For each i:
  - Flip z_i and resample descendants of z_i with fixed $\epsilon_{\setminus i}$
  - Recompute $f(x, z)$
  - Compute the local gradient (the above equation)

All the ϵ_i's are fixed.



Optimal estimator under the framework

(slow for use in every iteration, yet useful for evaluation purpose)

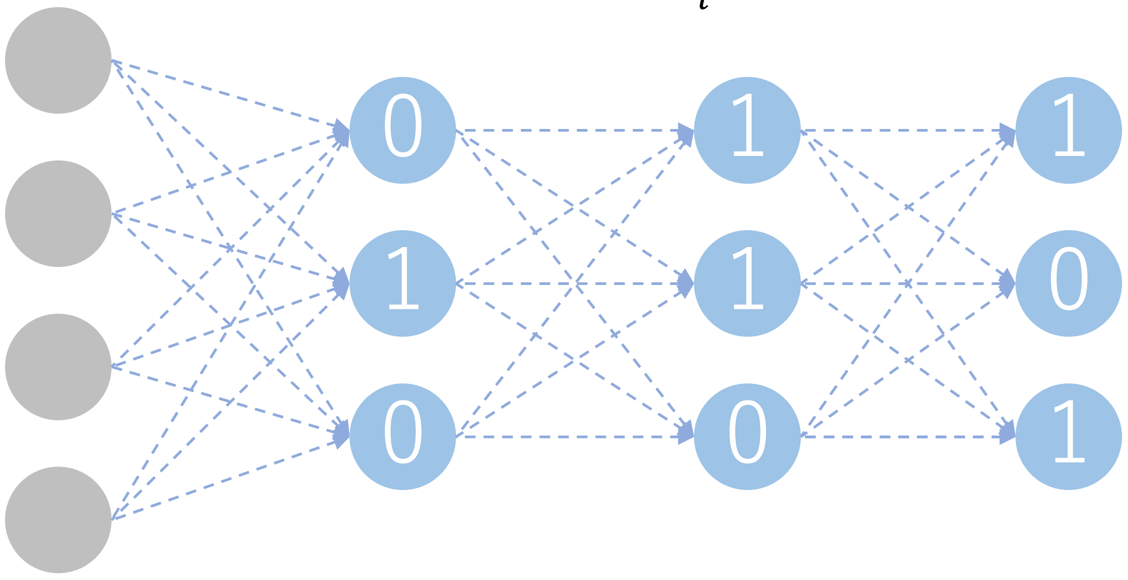
Exactly (or numerically) compute the local gradient:

$$\nabla_{\phi_i} \mathbb{E}_{\epsilon_i} f\left(x, g_\phi(x,\epsilon)\right) = \sum_{z_i} f(x,z) \nabla_{\phi_i} q_{\phi_i}(z_i|\mathrm{pa}_i)$$

Implementation (Bernoulli case):

- Sample ϵ and compute $z = g_\phi(x,\epsilon)$ and $f(x,z)$
- For each i:
  - Flip z_i and resample descendants of z_i with fixed $\epsilon_{\setminus i}$
  - Recompute $f(x,z)$
  - Compute the local gradient (the above equation)

All the ϵ_i's are fixed.



Optimal estimator under the framework

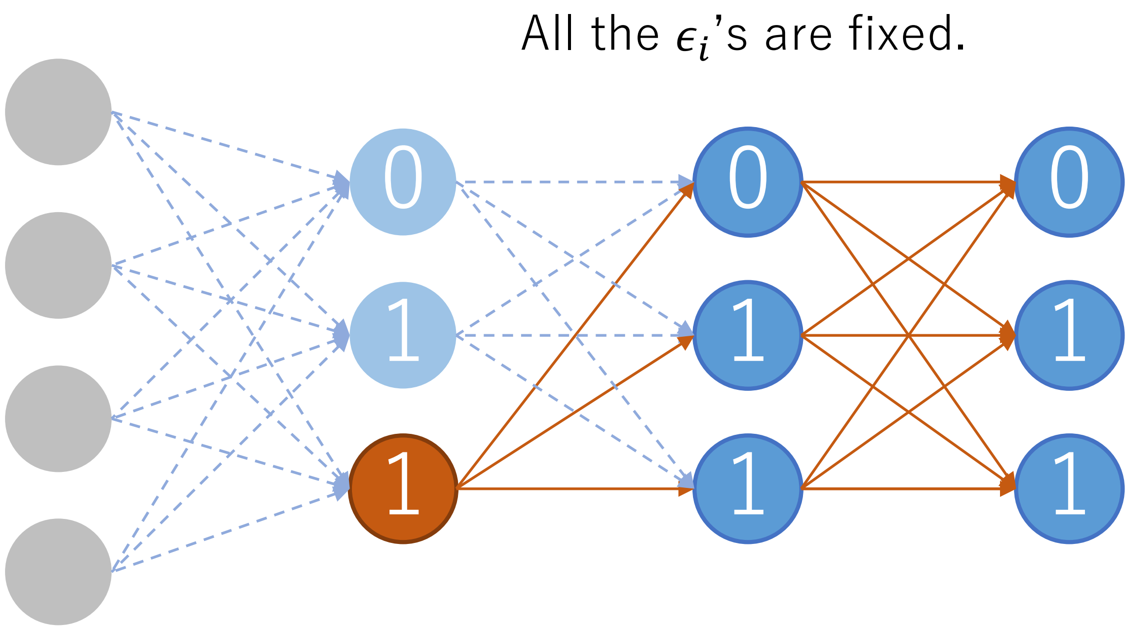
(slow for use in every iteration, yet useful for evaluation purpose)

Exactly (or numerically) compute the local gradient:

$$\nabla_{\phi_i} \mathbb{E}_{\epsilon_i} f\left(x, g_\phi(x, \epsilon)\right) = \sum_{z_i} f(x, z) \nabla_{\phi_i} q_{\phi_i}(z_i | \mathrm{pa}_i)$$

Implementation (Bernoulli case):

- Sample ϵ and compute $z = g_\phi(x, \epsilon)$ and $f(x, z)$
- For each i:
  - Flip z_i and resample descendants of z_i with fixed $\epsilon_{\backslash i}$
  - Recompute $f(x, z)$
  - Compute the local gradient (the above equation)



Optimal estimator under the framework

(slow for use in every iteration, yet useful for evaluation purpose)

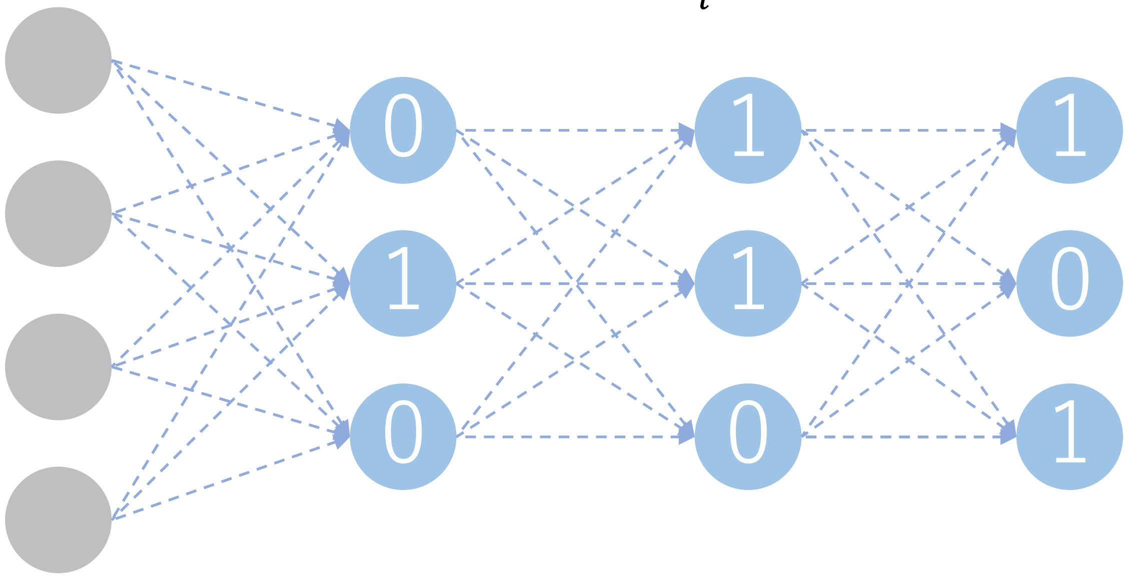
Exactly (or numerically) compute the local gradient:

$$\nabla_{\phi_i} \mathbb{E}_{\epsilon_i} f\left(x, g_\phi(x, \epsilon)\right) = \sum_{z_i} f(x, z) \nabla_{\phi_i} q_{\phi_i}(z_i | \mathrm{pa}_i)$$

Implementation (Bernoulli case):

- Sample ϵ and compute $z = g_\phi(x, \epsilon)$ and $f(x, z)$
- For each i:
  - Flip z_i and resample descendants of z_i with fixed $\epsilon_{\setminus i}$
  - Recompute $f(x, z)$
  - Compute the local gradient (the above equation)

All the ϵ_i's are fixed.



Optimal estimator under the framework
(slow for use in every iteration, yet useful for evaluation purpose)

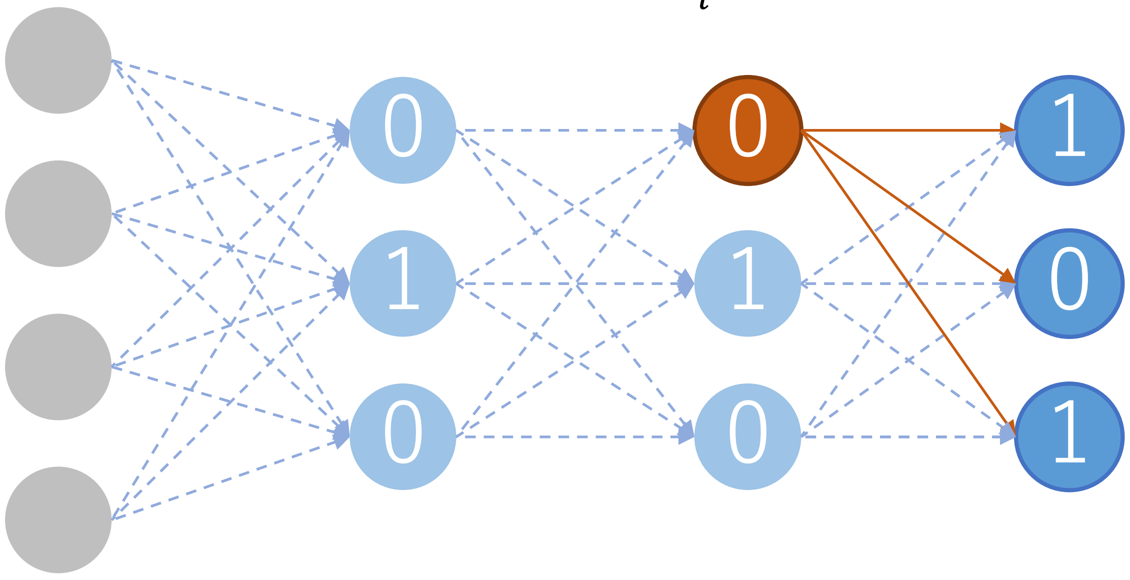
Exactly (or numerically) compute the local gradient:

$$\nabla_{\phi_i} \mathbb{E}_{\epsilon_i} f\left(x, g_\phi(x, \epsilon)\right) = \sum_{z_i} f(x, z) \nabla_{\phi_i} q_{\phi_i}(z_i | \mathrm{pa}_i)$$

Implementation (Bernoulli case):

- Sample ϵ and compute $z = g_\phi(x, \epsilon)$ and $f(x, z)$
- For each i:
    - Flip z_i and resample descendants of z_i with fixed $\epsilon_{\setminus i}$
    - Recompute $f(x, z)$
    - Compute the local gradient (the above equation)

All the ϵ_i's are fixed.



Optimal estimator under the framework

(slow for use in every iteration, yet useful for evaluation purpose)

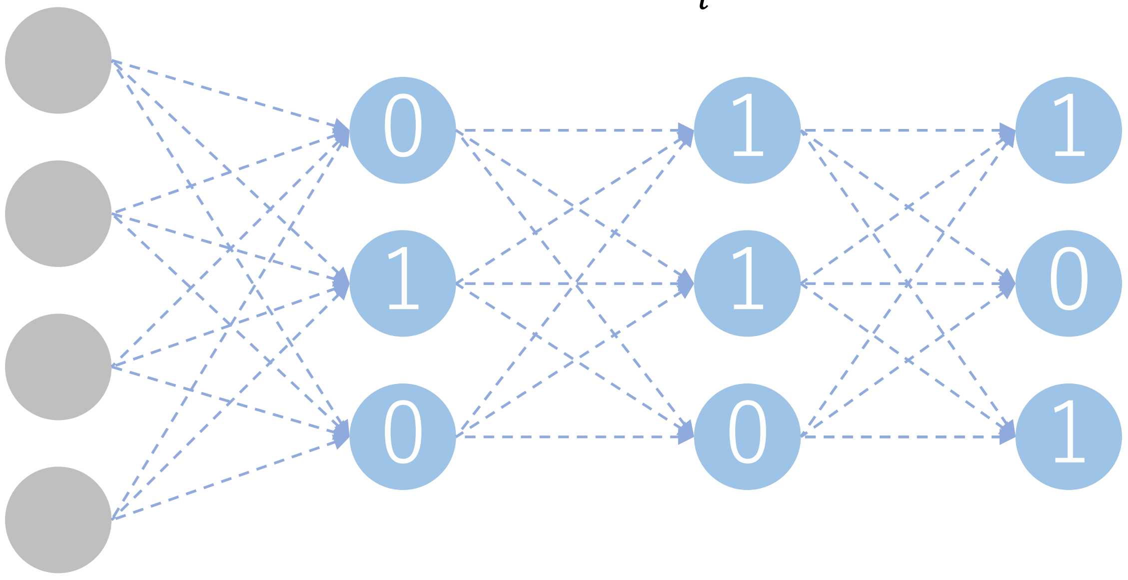
Exactly (or numerically) compute the local gradient:

$$\nabla_{\phi_i}\mathbb{E}_{\epsilon_i} f\left(x, g_\phi(x,\epsilon)\right) = \sum_{z_i} f(x,z)\nabla_{\phi_i} q_{\phi_i}(z_i|\mathrm{pa}_i)$$

Implementation (Bernoulli case):

- Sample ϵ and compute $z = g_\phi(x,\epsilon)$ and $f(x,z)$
- For each i:
  - Flip z_i and resample descendants of z_i with fixed $\epsilon_{\setminus i}$
  - Recompute $f(x,z)$
  - Compute the local gradient (the above equation)

All the ϵ_i's are fixed.



Optimal estimator under the framework

(slow for use in every iteration, yet useful for evaluation purpose)

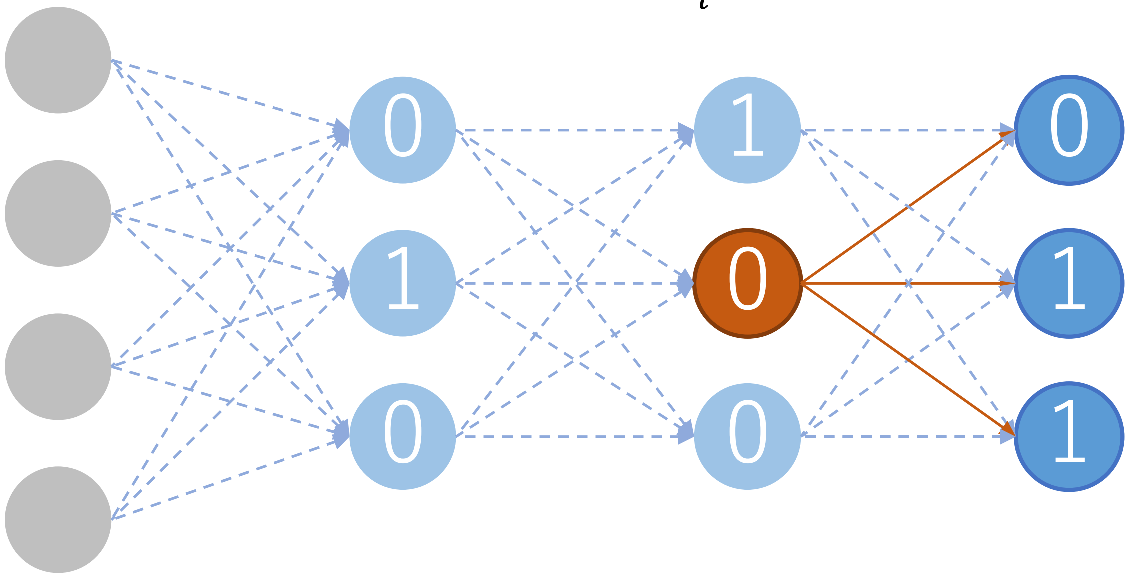
Exactly (or numerically) compute the local gradient:

$$\nabla_{\phi_i} \mathbb{E}_{\epsilon_i} f\left(x, g_\phi(x,\epsilon)\right) = \sum_{z_i} f(x,z) \nabla_{\phi_i} q_{\phi_i}(z_i|\mathrm{pa}_i)$$

Implementation (Bernoulli case):

- Sample ϵ and compute $z = g_\phi(x,\epsilon)$ and $f(x,z)$
- For each i:
  - Flip z_i and resample descendants of z_i with fixed $\epsilon_{\setminus i}$
  - Recompute $f(x,z)$
  - Compute the local gradient (the above equation)

All the ϵ_i's are fixed.



Optimal estimator under the framework
(slow for use in every iteration, yet useful for evaluation purpose)

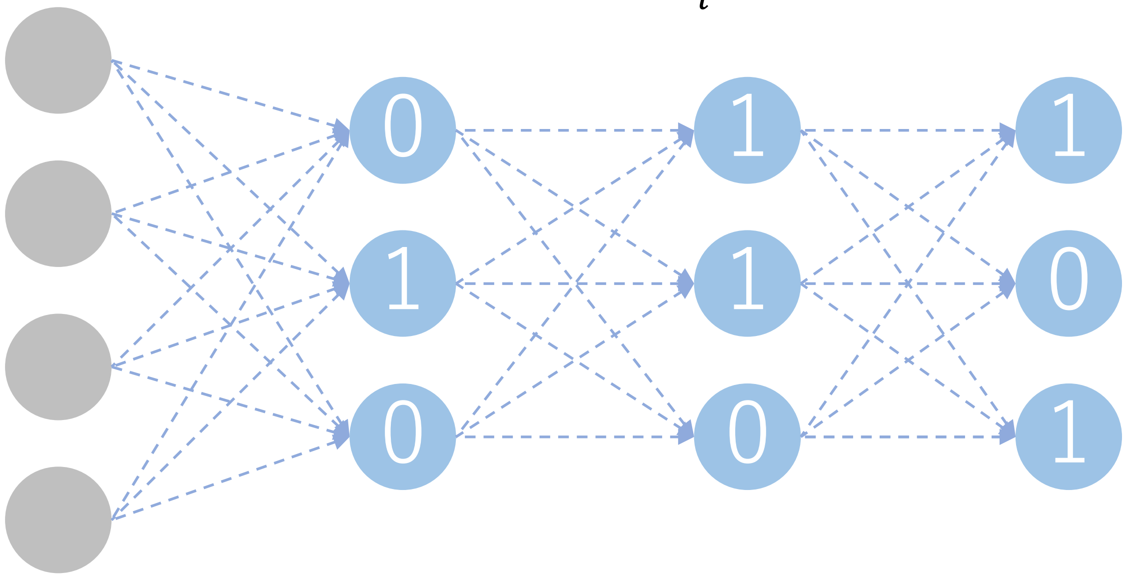
Exactly (or numerically) compute the local gradient:

$$\nabla_{\phi_i}\mathbb{E}_{\epsilon_i} f\left(x, g_\phi(x,\epsilon)\right) = \sum_{z_i} f(x,z)\nabla_{\phi_i} q_{\phi_i}(z_i|\mathrm{pa}_i)$$

Implementation (Bernoulli case):

- Sample ϵ and compute $z = g_\phi(x,\epsilon)$ and $f(x,z)$
- For each i:
  - Flip z_i and resample descendants of z_i with fixed $\epsilon_{\setminus i}$
  - Recompute $f(x,z)$
  - Compute the local gradient (the above equation)

All the ϵ_i's are fixed.



Optimal estimator under the framework

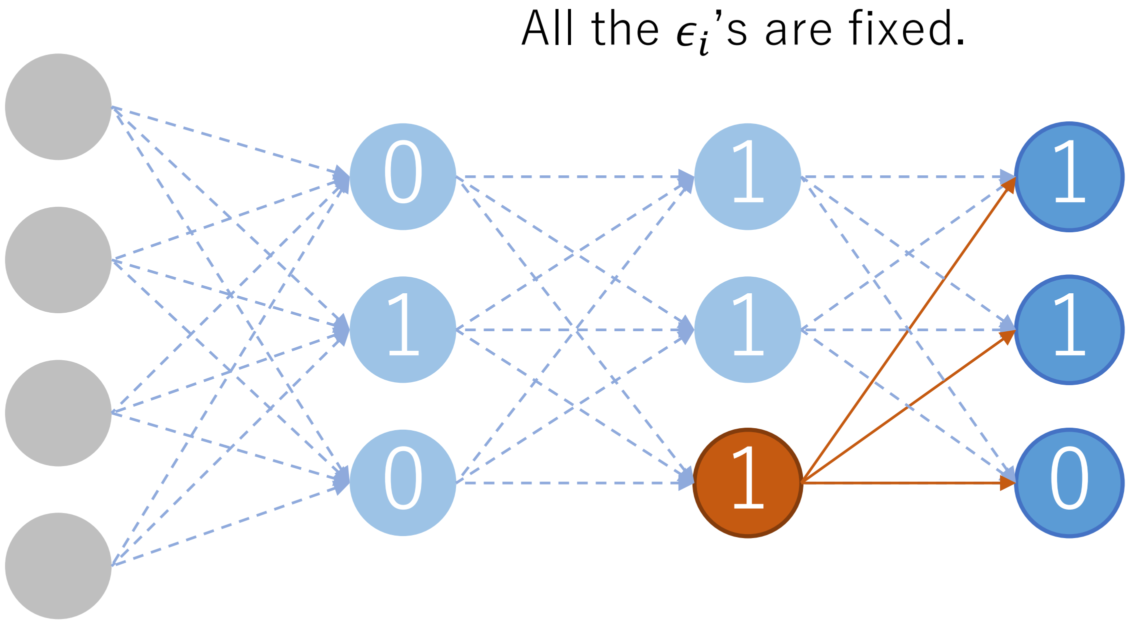
(slow for use in every iteration, yet useful for evaluation purpose)

Exactly (or numerically) compute the local gradient:

$$\nabla_{\phi_i}\mathbb{E}_{\epsilon_i} f\left(x, g_\phi(x,\epsilon)\right) = \sum_{z_i} f(x,z)\nabla_{\phi_i} q_{\phi_i}(z_i|\mathrm{pa}_i)$$

Implementation (Bernoulli case):

- Sample ϵ and compute $z = g_\phi(x,\epsilon)$ and $f(x,z)$
- For each i:
  - Flip z_i and resample descendants of z_i with fixed $\epsilon_{\setminus i}$
  - Recompute $f(x,z)$
  - Compute the local gradient (the above equation)



Optimal estimator under the framework

(slow for use in every iteration, yet useful for evaluation purpose)

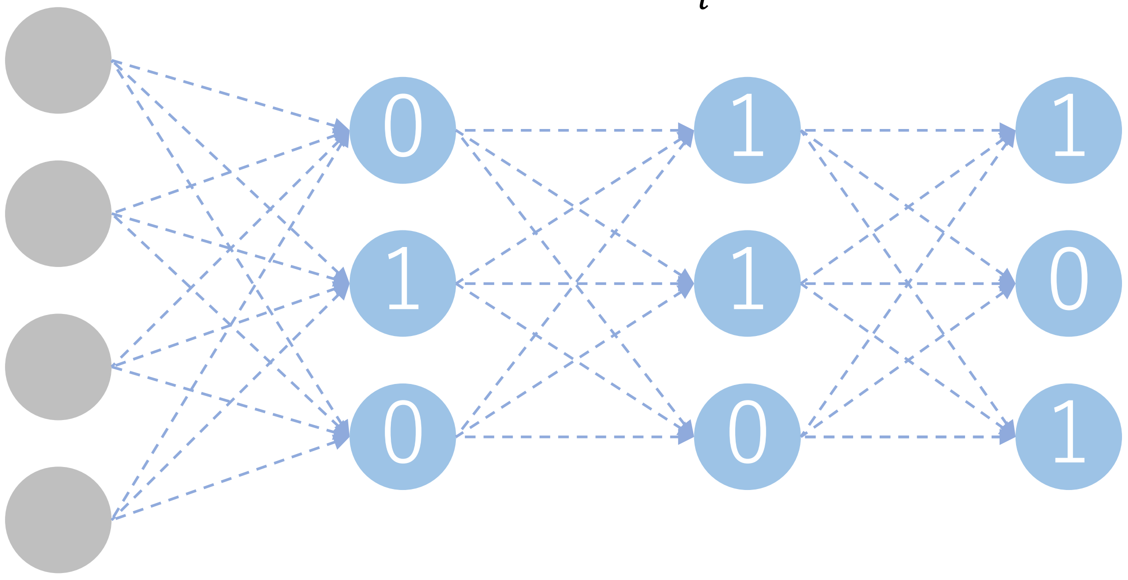
Exactly (or numerically) compute the local gradient:

$$\nabla_{\phi_i}\mathbb{E}_{\epsilon_i} f\left(x, g_\phi(x,\epsilon)\right) = \sum_{z_i} f(x,z)\nabla_{\phi_i} q_{\phi_i}(z_i|\mathrm{pa}_i)$$

Implementation (Bernoulli case):

- Sample ϵ and compute $z = g_\phi(x,\epsilon)$ and $f(x,z)$
- For each i:
  - Flip z_i and resample descendants of z_i with fixed $\epsilon_{\setminus i}$
  - Recompute $f(x,z)$
  - Compute the local gradient (the above equation)

All the ϵ_i's are fixed.



Optimal estimator under the framework

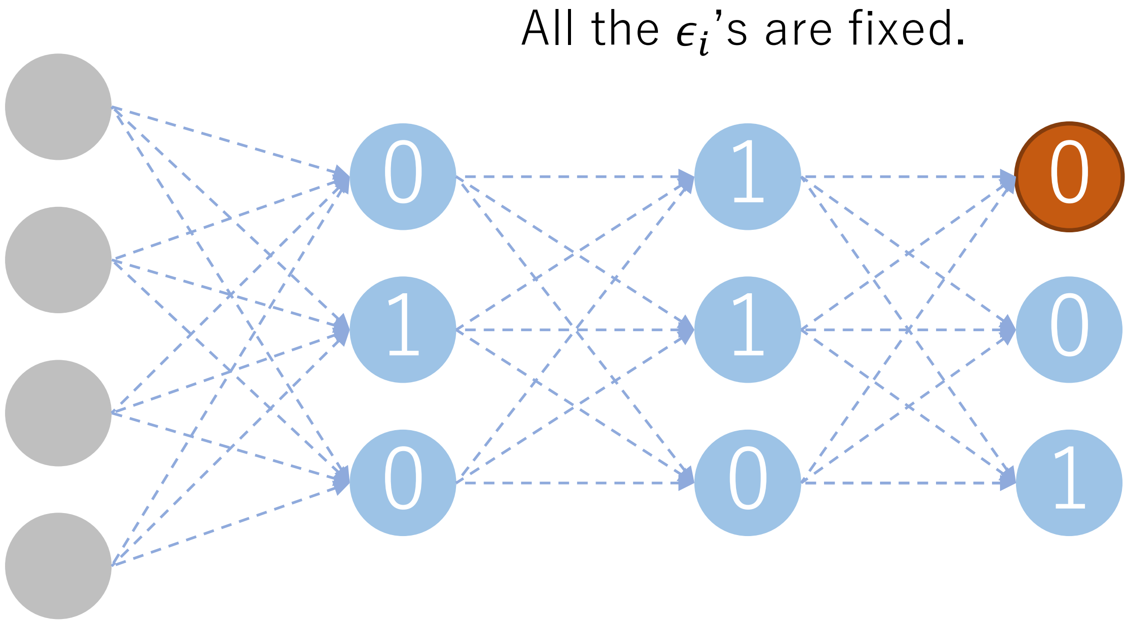
(slow for use in every iteration, yet useful for evaluation purpose)

Exactly (or numerically) compute the local gradient:

$$\nabla_{\phi_i}\mathbb{E}_{\epsilon_i} f\left(x, g_\phi(x,\epsilon)\right) = \sum_{z_i} f(x,z)\nabla_{\phi_i} q_{\phi_i}(z_i|\mathrm{pa}_i)$$

Implementation (Bernoulli case):

- Sample ϵ and compute $z = g_\phi(x,\epsilon)$ and $f(x,z)$
- For each i:
  - Flip z_i and resample descendants of z_i with fixed $\epsilon_{\setminus i}$
  - Recompute $f(x,z)$
  - Compute the local gradient (the above equation)



Optimal estimator under the framework

(slow for use in every iteration, yet useful for evaluation purpose)

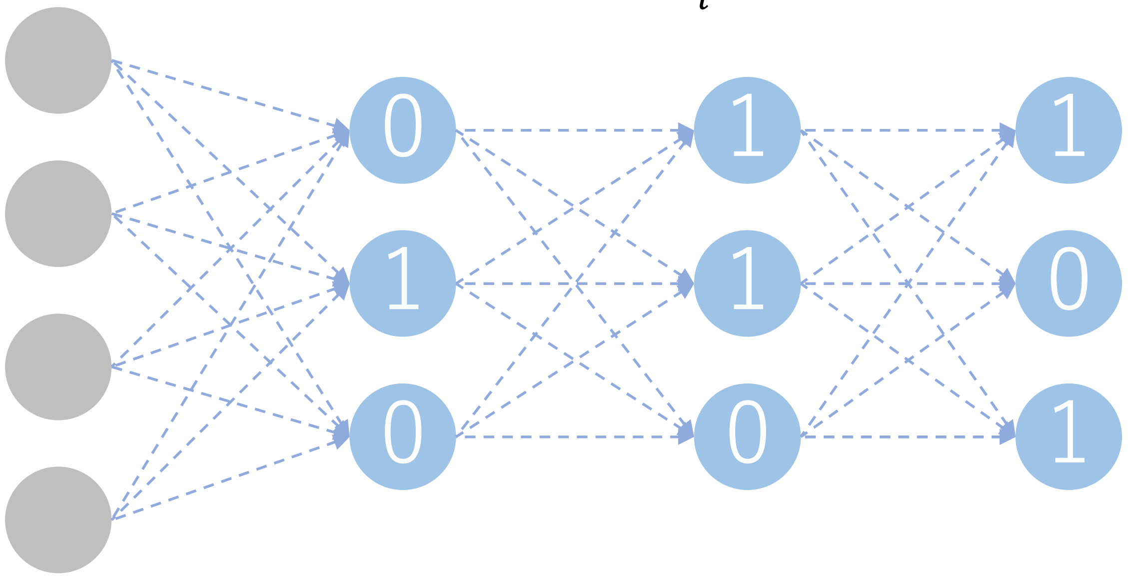
Exactly (or numerically) compute the local gradient:

$$\nabla_{\phi_i} \mathbb{E}_{\epsilon_i} f\left(x, g_\phi(x, \epsilon)\right) = \sum_{z_i} f(x, z) \nabla_{\phi_i} q_{\phi_i}(z_i | \mathrm{pa}_i)$$

Implementation (Bernoulli case):

- Sample ϵ and compute $z = g_\phi(x, \epsilon)$ and $f(x, z)$
- For each i:
  - Flip z_i and resample descendants of z_i with fixed $\epsilon_{\setminus i}$
  - Recompute $f(x, z)$
  - Compute the local gradient (the above equation)

All the ϵ_i's are fixed.



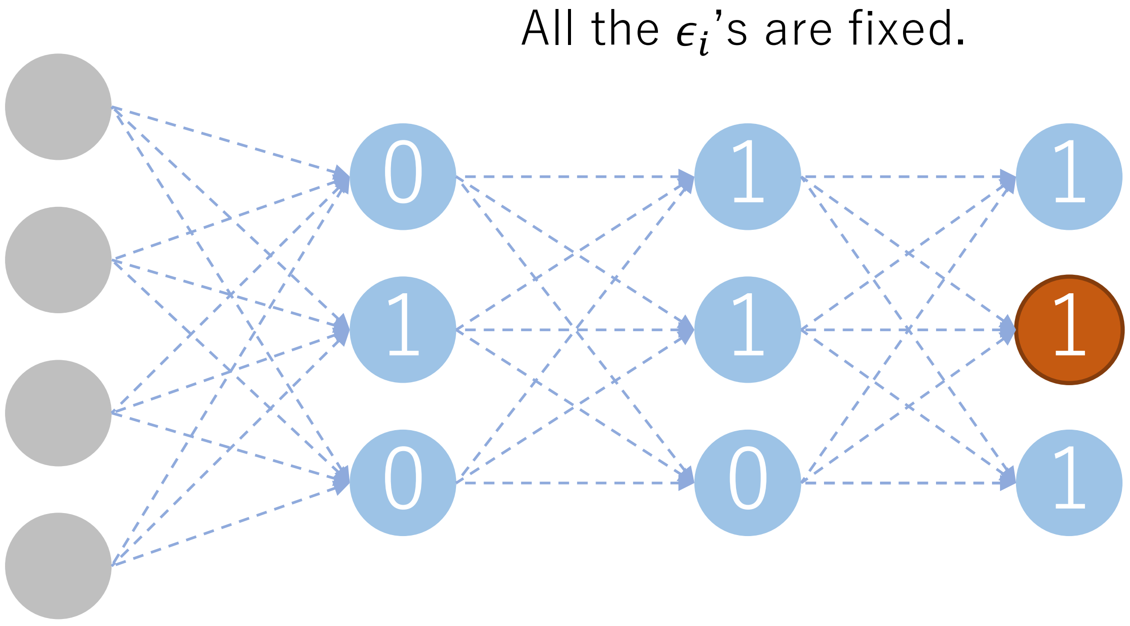
Optimal estimator under the framework
(slow for use in every iteration, yet useful for evaluation purpose)

Exactly (or numerically) compute the local gradient:

$$\nabla_{\phi_i} \mathbb{E}_{\epsilon_i} f\left(x, g_\phi(x, \epsilon)\right) = \sum_{z_i} f(x, z) \nabla_{\phi_i} q_{\phi_i}(z_i | \mathrm{pa}_i)$$

Implementation (Bernoulli case):

- Sample ϵ and compute $z = g_\phi(x, \epsilon)$ and $f(x, z)$
- For each i:
  - Flip z_i and resample descendants of z_i with fixed $\epsilon_{\setminus i}$
  - Recompute $f(x, z)$
  - Compute the local gradient (the above equation)



Optimal estimator under the framework

(slow for use in every iteration, yet useful for evaluation purpose)

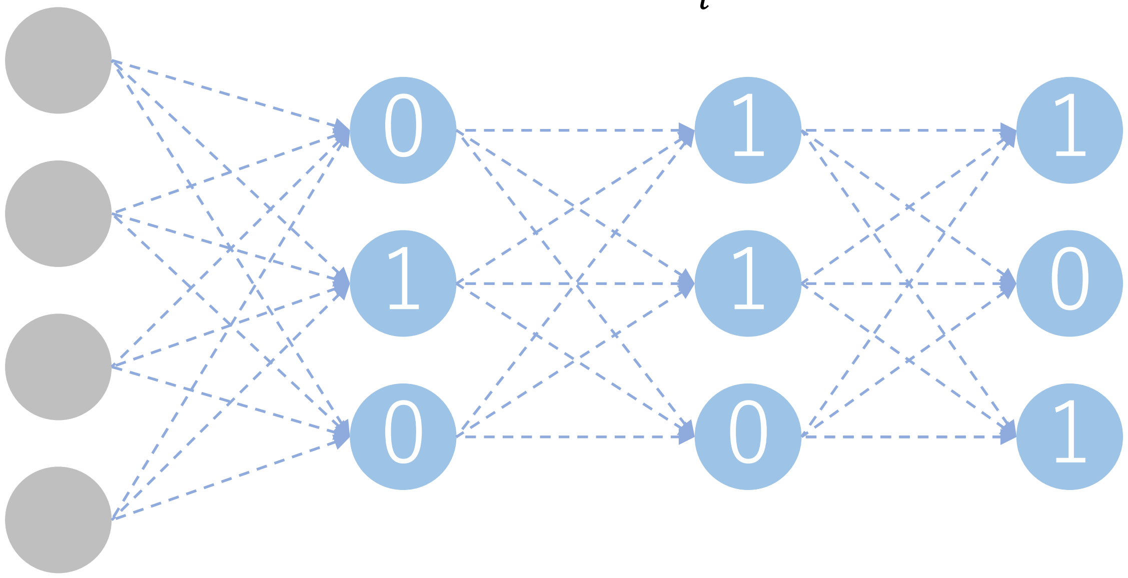
Exactly (or numerically) compute the local gradient:

$$\nabla_{\phi_i} \mathbb{E}_{\epsilon_i} f\left(x, g_\phi(x, \epsilon)\right) = \sum_{z_i} f(x, z) \nabla_{\phi_i} q_{\phi_i}(z_i | \mathrm{pa}_i)$$

Implementation (Bernoulli case):

- Sample ϵ and compute $z = g_\phi(x, \epsilon)$ and $f(x, z)$
- For each i:
  - Flip z_i and resample descendants of z_i with fixed $\epsilon_{\setminus i}$
  - Recompute $f(x, z)$
  - Compute the local gradient (the above equation)

All the ϵ_i's are fixed.



Optimal estimator under the framework

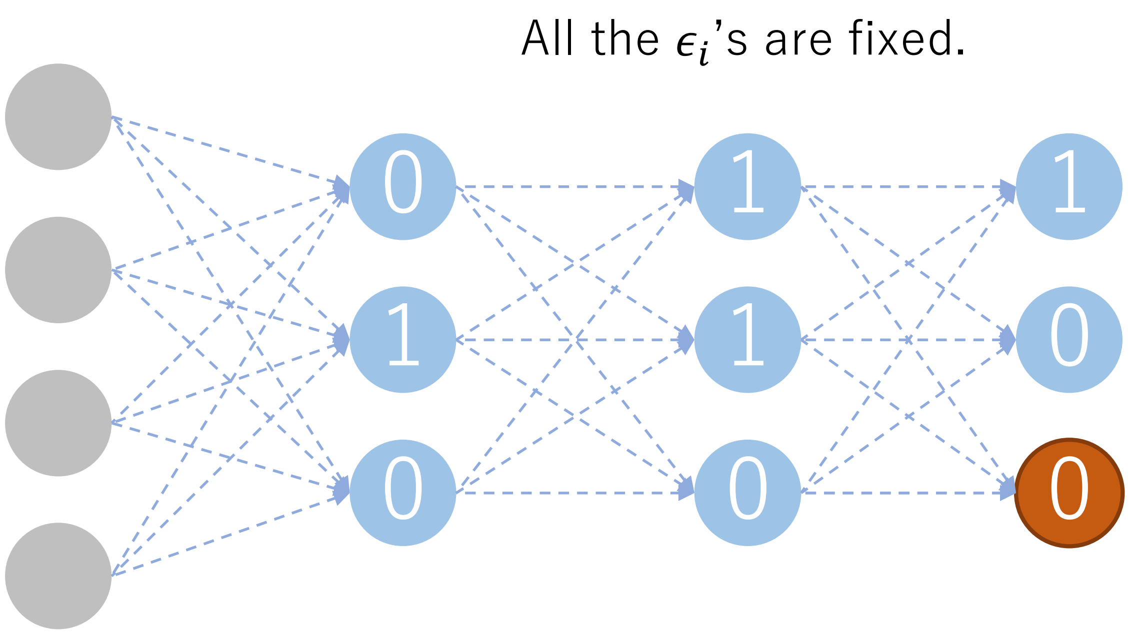
(slow for use in every iteration, yet useful for evaluation purpose)

Exactly (or numerically) compute the local gradient:

$$\nabla_{\phi_i} \mathbb{E}_{\epsilon_i} f\left(x, g_\phi(x, \epsilon)\right) = \sum_{z_i} f(x, z) \nabla_{\phi_i} q_{\phi_i}(z_i | \mathrm{pa}_i)$$

Implementation (Bernoulli case):

- Sample ϵ and compute $z = g_\phi(x, \epsilon)$ and $f(x, z)$
- For each i:
  - Flip z_i and resample descendants of z_i with fixed $\epsilon_{\setminus i}$
  - Recompute $f(x, z)$
  - Compute the local gradient (the above equation)



Optimal estimator under the framework

(slow for use in every iteration, yet useful for evaluation purpose)

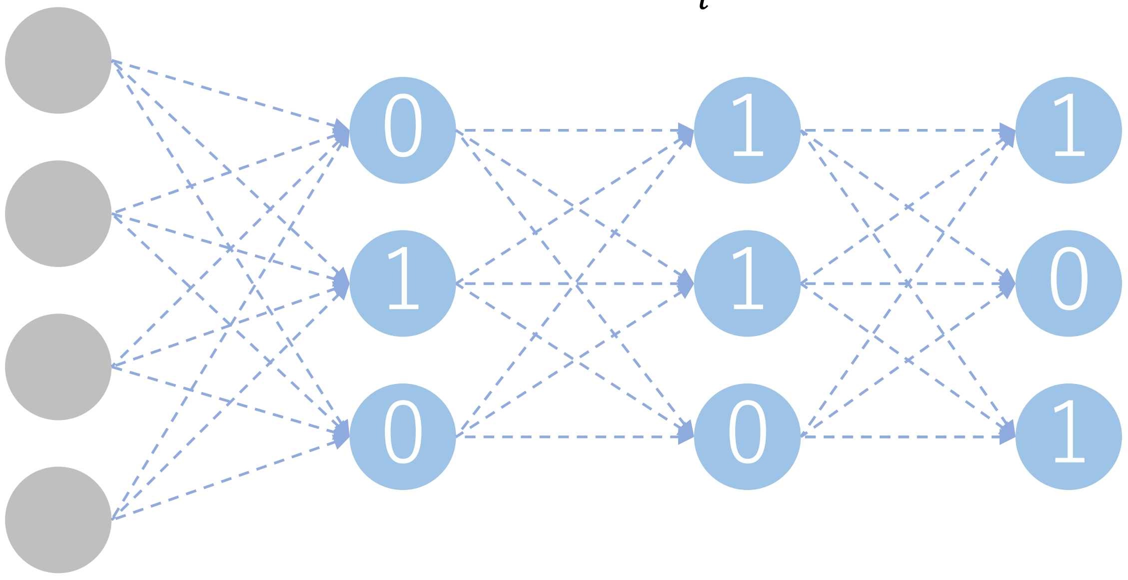
Exactly (or numerically) compute the local gradient:

$$\nabla_{\phi_i}\mathbb{E}_{\epsilon_i}f\left(x, g_\phi(x,\epsilon)\right) = \sum_{z_i} f(x,z)\nabla_{\phi_i}q_{\phi_i}(z_i|\mathrm{pa}_i)$$

Implementation (Bernoulli case):

- Sample ϵ and compute $z = g_\phi(x,\epsilon)$ and $f(x,z)$
- For each i:
  - Flip z_i and resample descendants of z_i with fixed $\epsilon_{\setminus i}$
  - Recompute $f(x,z)$
  - Compute the local gradient (the above equation)

All the ϵ_i's are fixed.



Evaluating the variance of estimators within the framework

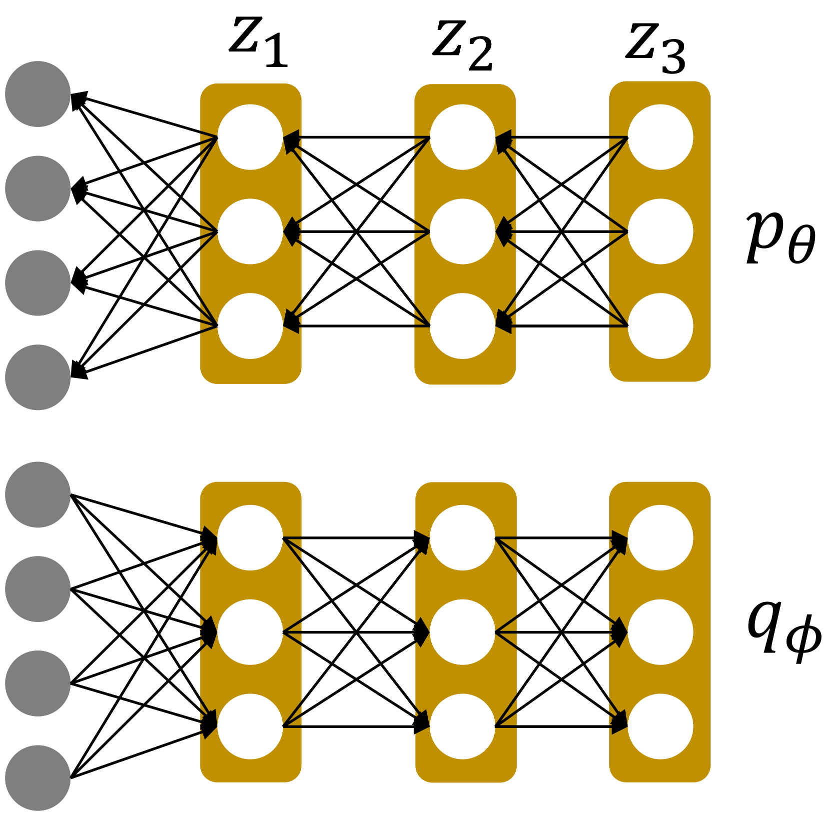
Theorem 1 The optimal estimator achieves the minimum variance among all estimators within the framework.

(∵ the property of Rao-Blackwellization)

Theorem 2 When z_i is a Bernoulli variable, there is an **independent baseline** $b_i^\star$ with which the likelihood-ratio estimator achieves the optimal variance.

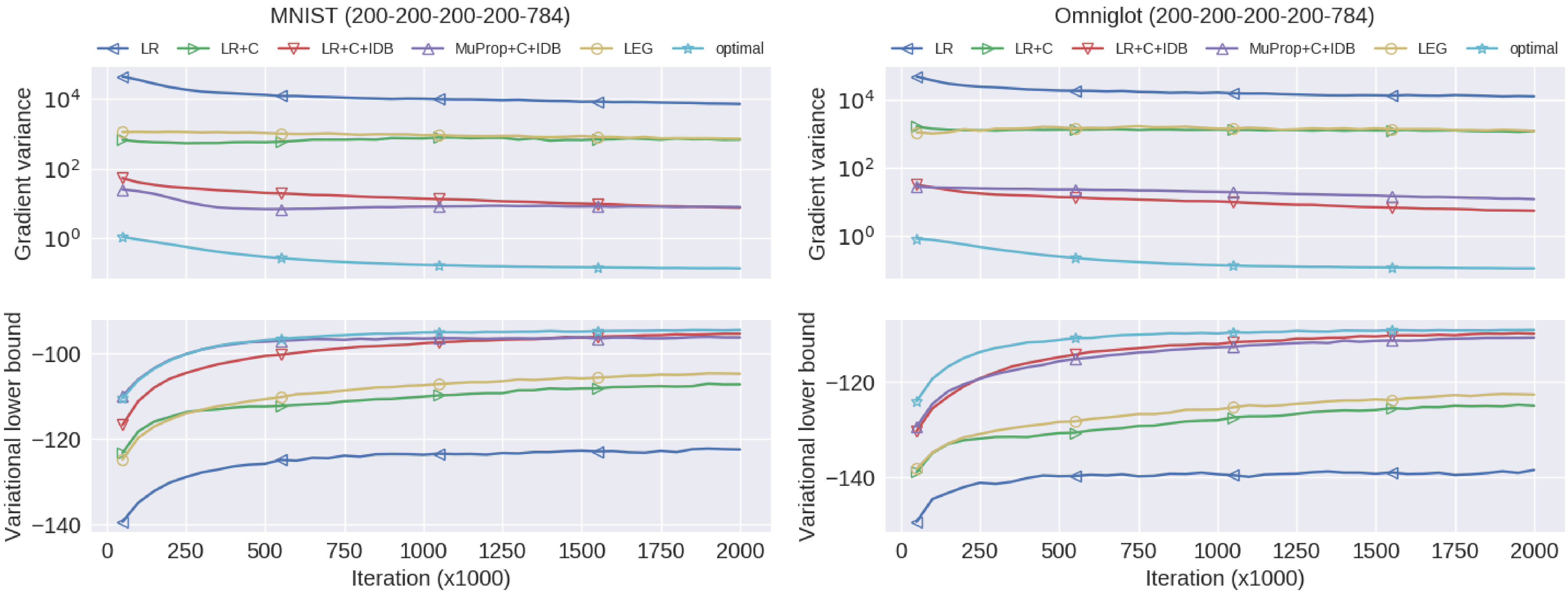
(I.e., LR with independent baseline can be the optimal estimator.)

Experiment: variational learning of sigmoid belief networks



- Datasets: MNIST and Omniglot
- 784-dimensional binary (0/1) inputs
- Each latent variable follows a Bernoulli distribution with the logit given by the net input (i.e., sigmoid-Bernoulli unit)
- In the optimal estimator, the Bernoulli unit is reparameterized by 0/1 thresholding at $\epsilon \sim U(0,1)$.

Experimental results: variational learning of sigmoid belief nets

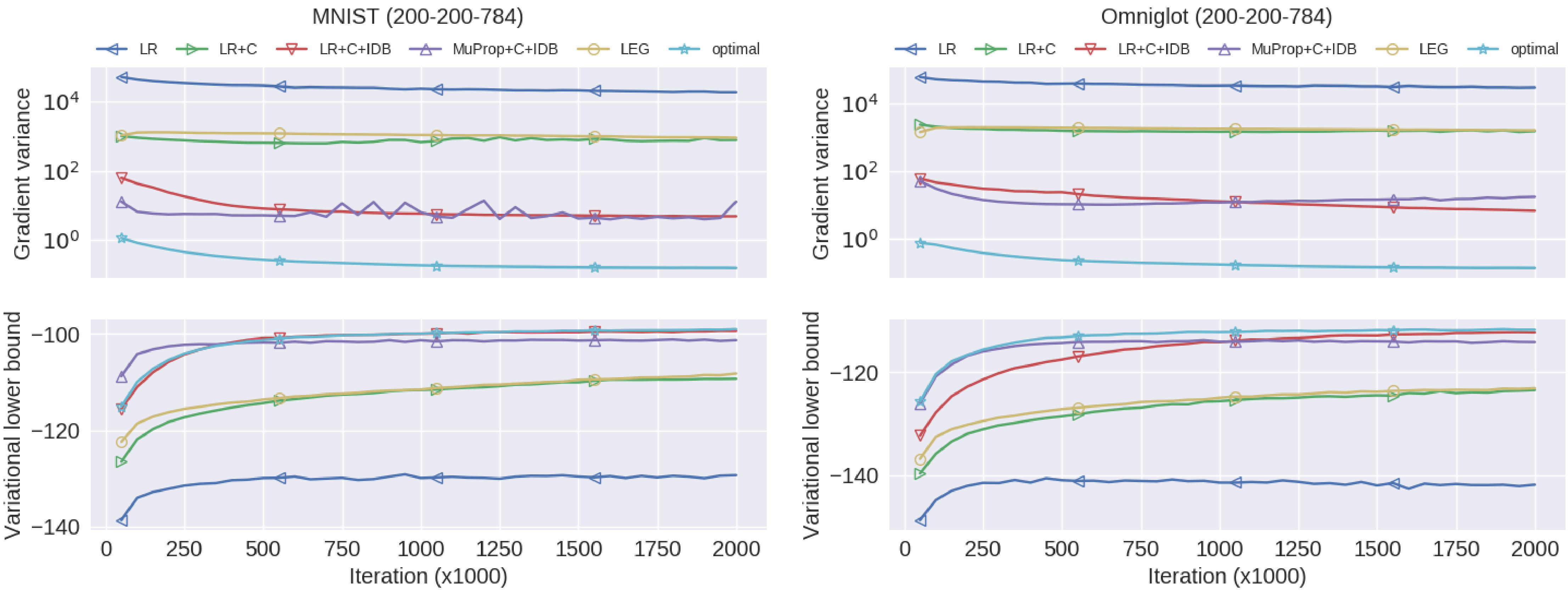


Conclusion

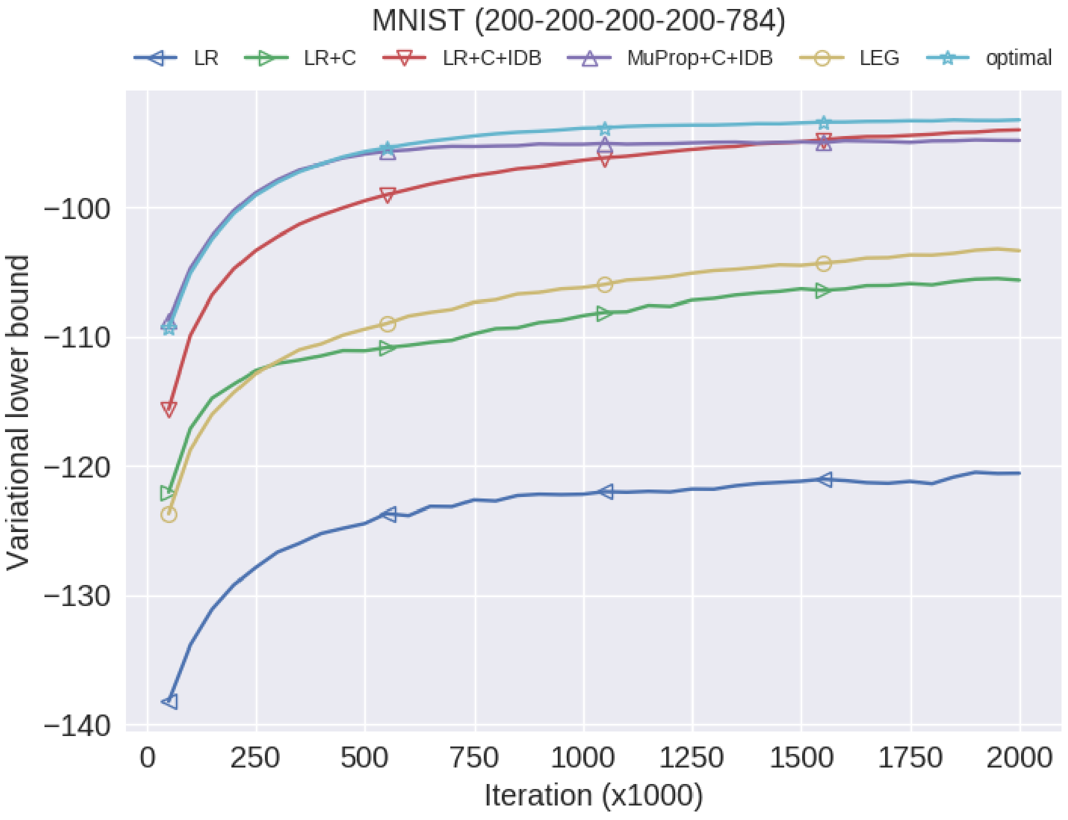
- We proposed a framework of gradient estimators for stochastic computational graph by reparameterization and local marginalization.
- We formulated a hierarchy of baseline techniques for likelihood-ratio estimators and showed the relationship between this hierarchy and the optimal estimator.
- The experimental results show that the variance of gradient estimation for binary discrete variables is approaching to the optimum with recent advancements, yet a non-negligible gap still exists, indicating the possibility of further improvements.

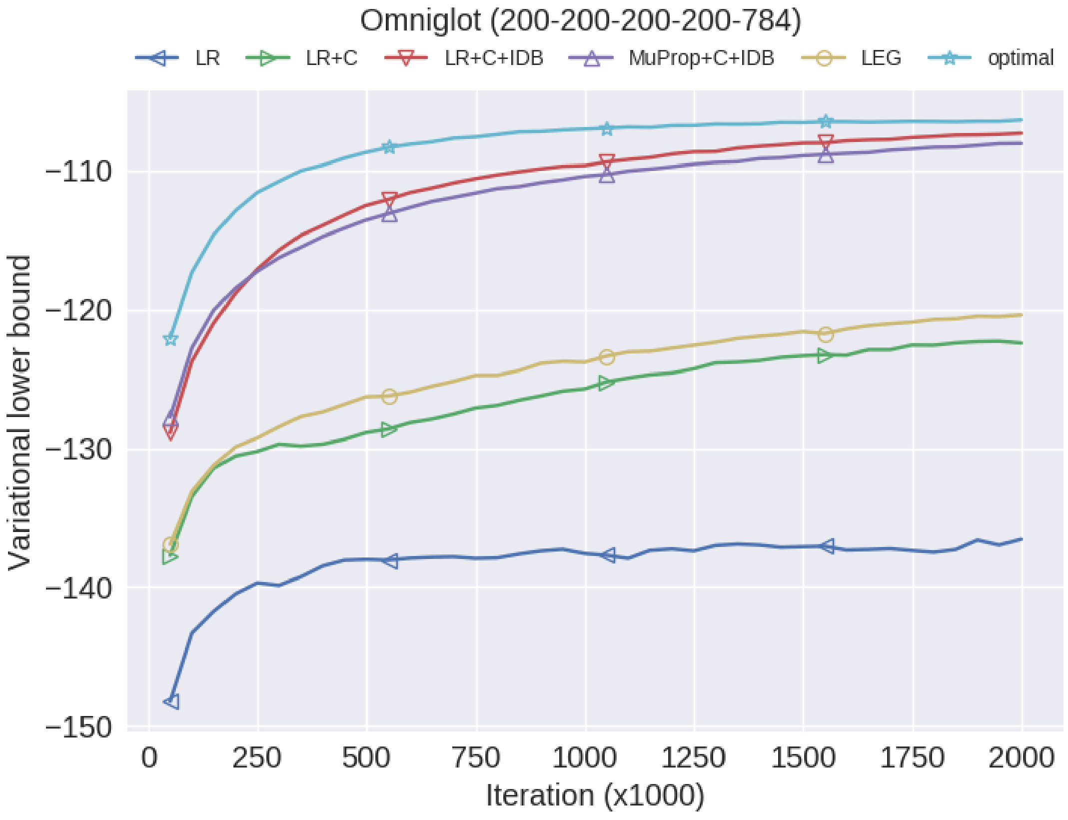
(end)

Appendix: results with shallow networks

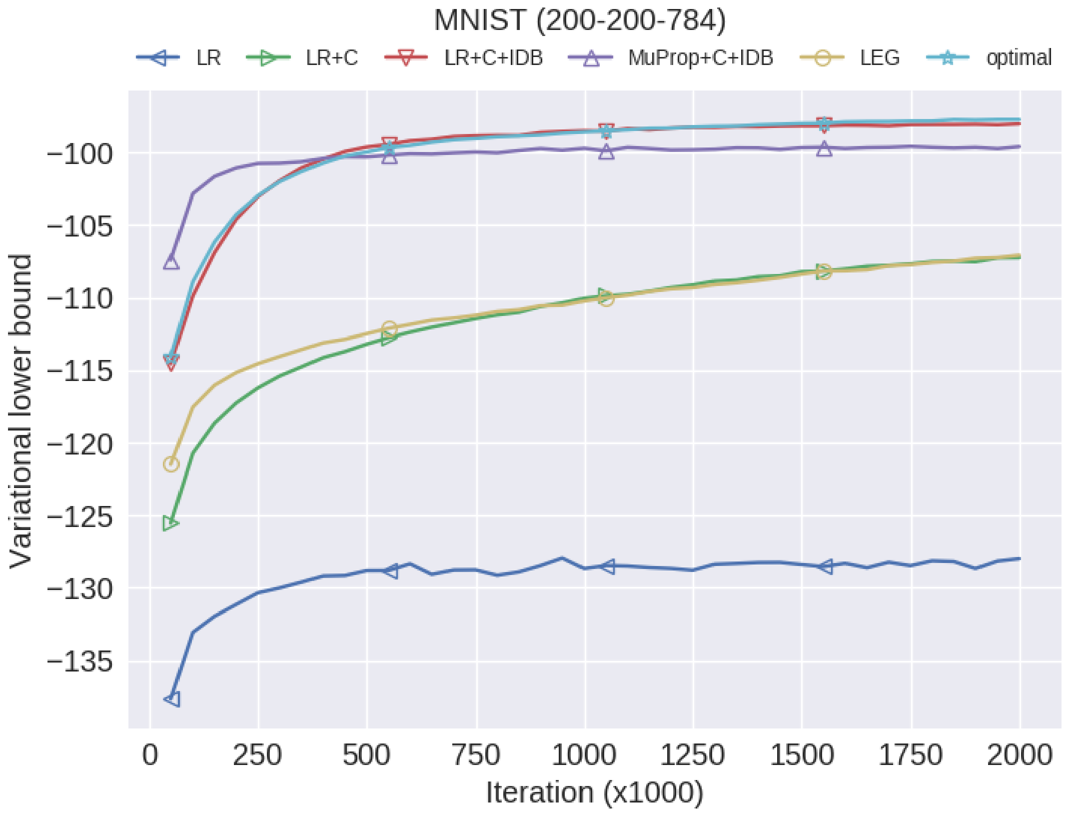


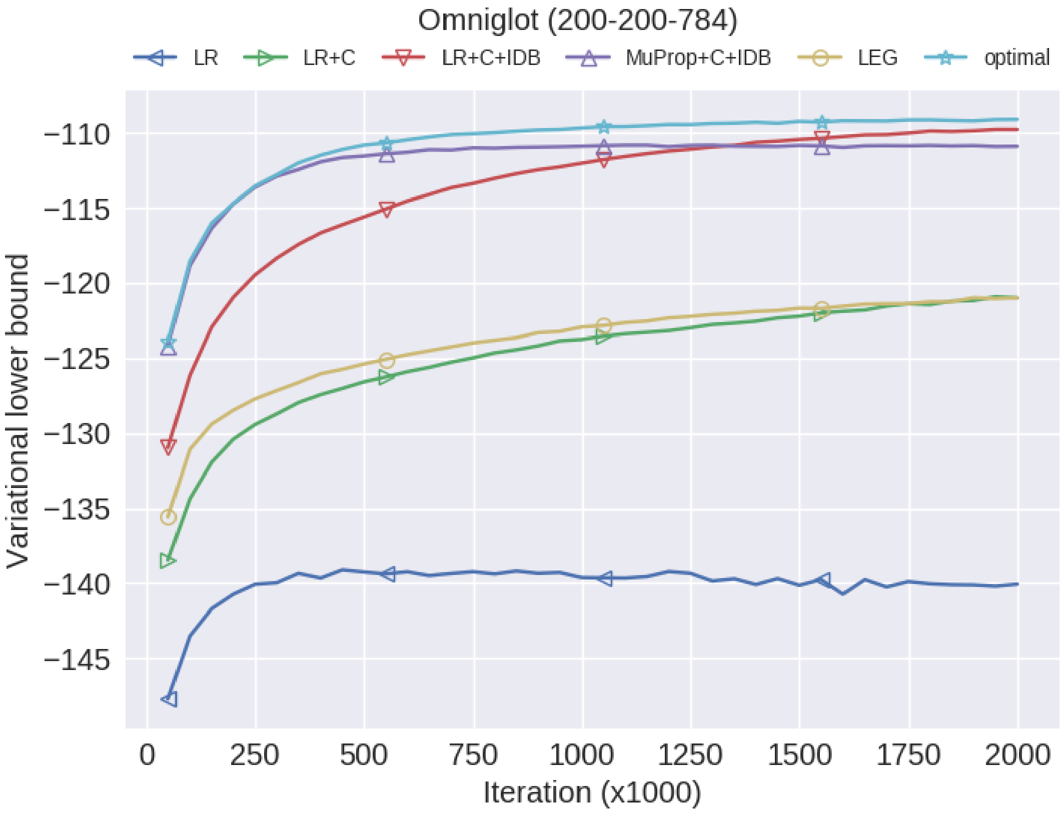
Appendix: training curve of deep networks





Appendix: training curve of shallow networks





Appendix: final performance on test sets

	MNIST (shallow)		MNIST (deep)		Omniglot (shallow)		Omniglot (deep)	
	VB	LL	VB	LL	VB	LL	VB	LL
LR	-127.33	-108.53	-119.93	-103.53	-139.17	-124.08	-137.54	-122.85
LR+C	-107.21	-97.90	-105.38	-95.30	-122.10	-113.87	-123.27	-114.27
LR+C+IDB	-98.04	-92.68	-94.10	-89.02	-111.10	-107.14	-108.72	-105.00
MuProp+C+IDB	-99.96	-94.23	-95.03	-89.83	-112.97	-108.28	-109.55	-105.52
LEG	-106.75	-98.22	-103.26	-93.26	-121.68	-113.56	-121.27	-112.80
optimal	-97.64	-92.55	-93.31	-88.97	-110.60	-106.90	-108.17	-104.85

- VB stands for variational bound
- LL stands for log likelihood, which is approximated by "Monte Carlo objective" using sample of size 50,000