# Smoothness Analysis for Probabilistic Programs

Wonyeol Lee
CMU (→ POSTECH)

Joint work with: Xavier Rival, Hongseok Yang, Hangyeol Yu

Slides based on: [POPL'23], [NeurIPS'18]

PROBPROG 2024

Part 1

# Smoothness Analysis for Probabilistic Programs

Wonyeol Lee
CMU (→ POSTECH)

Joint work with: Xavier Rival, Hongseok Yang, Hangyeol Yu

Slides based on: [POPL'23], [NeurIPS'18]

PROBPROG 2024

Part 1

# Smoothness Analysis for Probabilistic Programs

Part 2


Wonyeol Lee

CMU (→ POSTECH)


Joint work with: Xavier Rival, Hongseok Yang, Hangyeol Yu

Slides based on: [POPL'23], [NeurIPS'18]

PROBPROG 2024

# $f$ is "smooth"

- In mathematics, $f$ is infinitely differentiable.
- In optimization, $\nabla f$ is Lipschitz continuous.

# $f$ is "smooth"

- In mathematics, $f$ is infinitely differentiable.
- In optimization, $\nabla f$ is Lipschitz continuous.

- In this talk, $f$ is "well-behaved."
- Examples:
  - differentiable, infinitely differentiable, ...
  - continuous, Lipschitz continuous, ...
  - locally bounded, measurable, ...

→ Part 1: Smoothness in Probabilistic Programming

[NeurIPS’18]

Part 2: Smoothness Analysis of Programs

[POPL’23]

# Probabilistic Programming

- Example:

```
def p(): # model
  z = pyro.sample(“z”, Normal(0., 5.))
  if (z > 0):  pyro.sample(“x”, Normal( 1., 1.), obs=0.)
  else:        pyro.sample(“x”, Normal(-2., 1.), obs=0.)
```

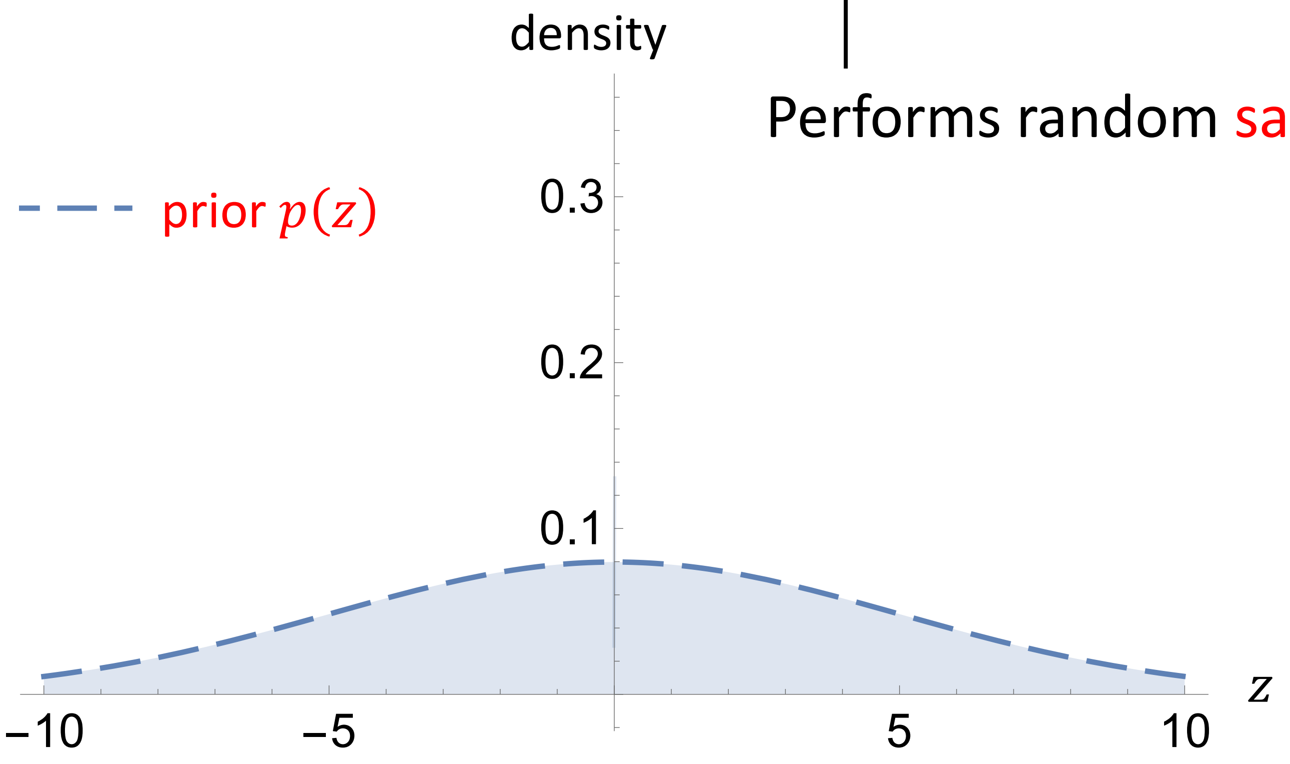
Written in Pyro language.
Describes a probabilistic model to be inferred.

# Probabilistic Programming

- Example:

```
def p(): # model
  z = pyro.sample(“z”, Normal(0., 5.))
  if (z > 0): pyro.sample(“x”, Normal( 1., 1.), obs=0.)
  else:       pyro.sample(“x”, Normal(-2., 1.), obs=0.)
```

Performs random sampling.

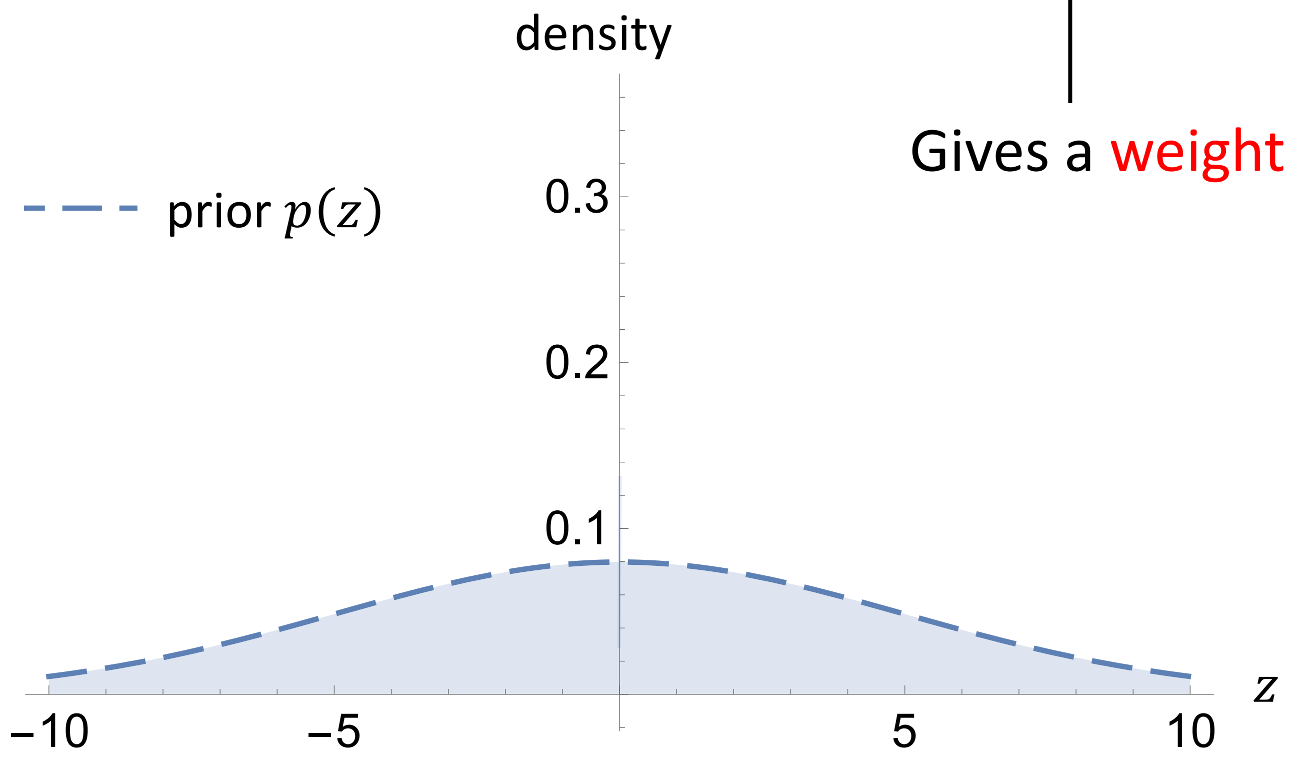


# Probabilistic Programming

- Example:

```
def p(): # model
  z = pyro.sample("z", Normal(0., 5.))
  if (z > 0):  pyro.sample("x", Normal( 1., 1.), obs=0.)
  else:        pyro.sample("x", Normal(-2., 1.), obs=0.)
```

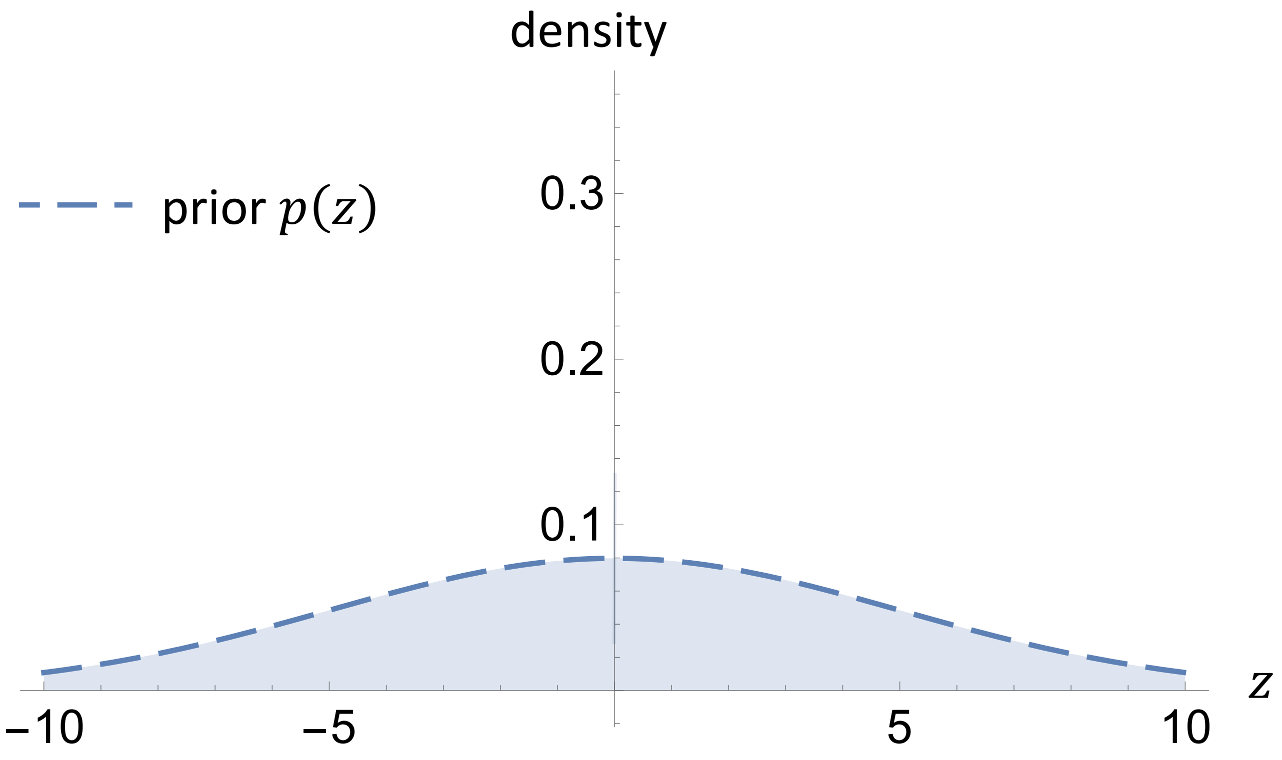
Gives a weight to a trace of program.



# Probabilistic Programming

- Example:

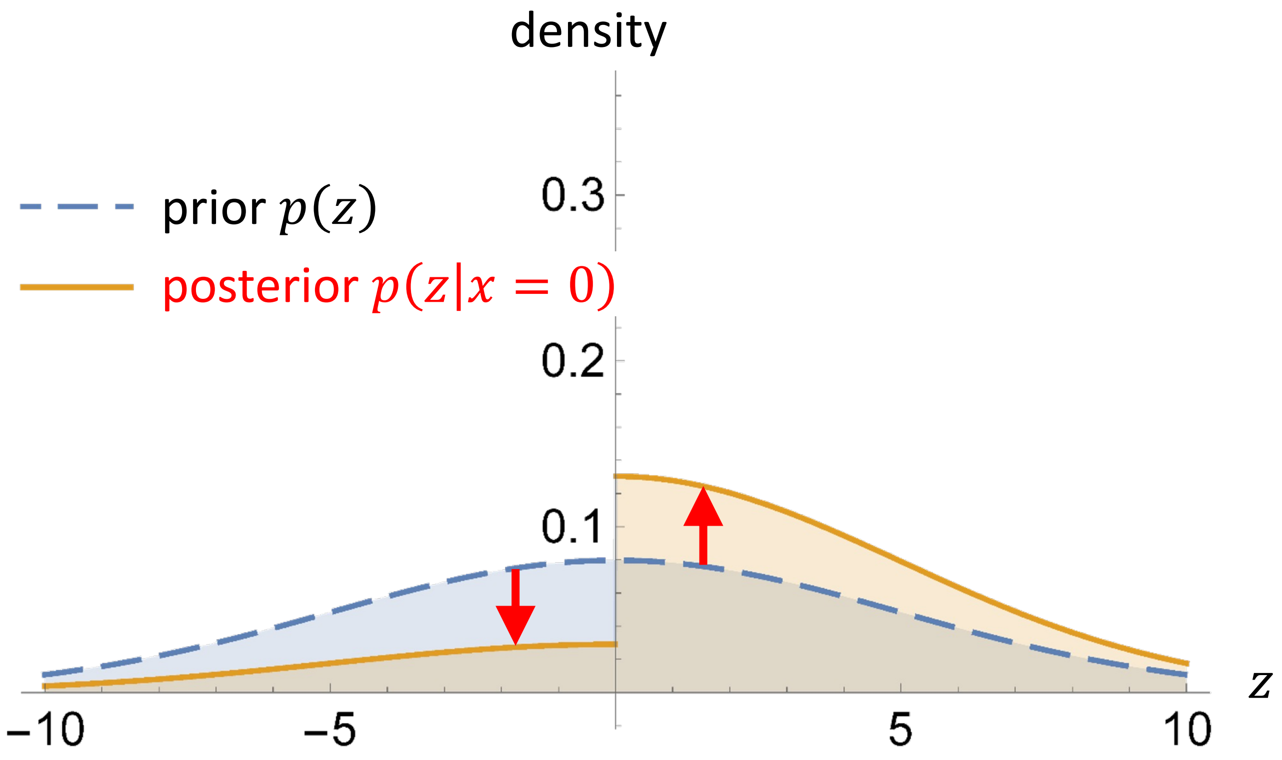
```
def p(): # model
  z = pyro.sample("z", Normal(0., 5.))
  if (z > 0):  pyro.sample("x", Normal( 1., 1.), obs=0.)
  else:        pyro.sample("x", Normal(-2., 1.), obs=0.)
```



# Probabilistic Programming

- Example:

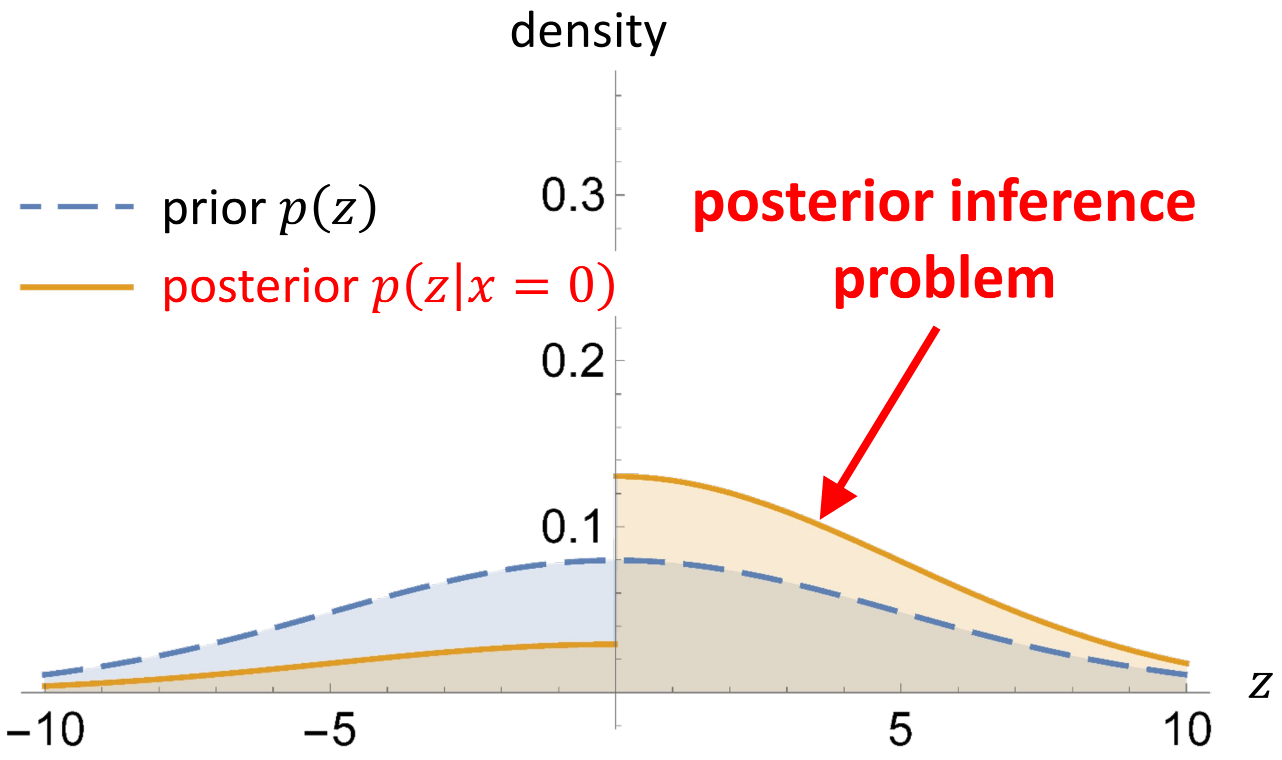
```
def p(): # model
  z = pyro.sample("z", Normal(0., 5.))
  if (z > 0):  pyro.sample("x", Normal( 1., 1.), obs=0.)
  else:        pyro.sample("x", Normal(-2., 1.), obs=0.)
```



# Probabilistic Programming

- Example:

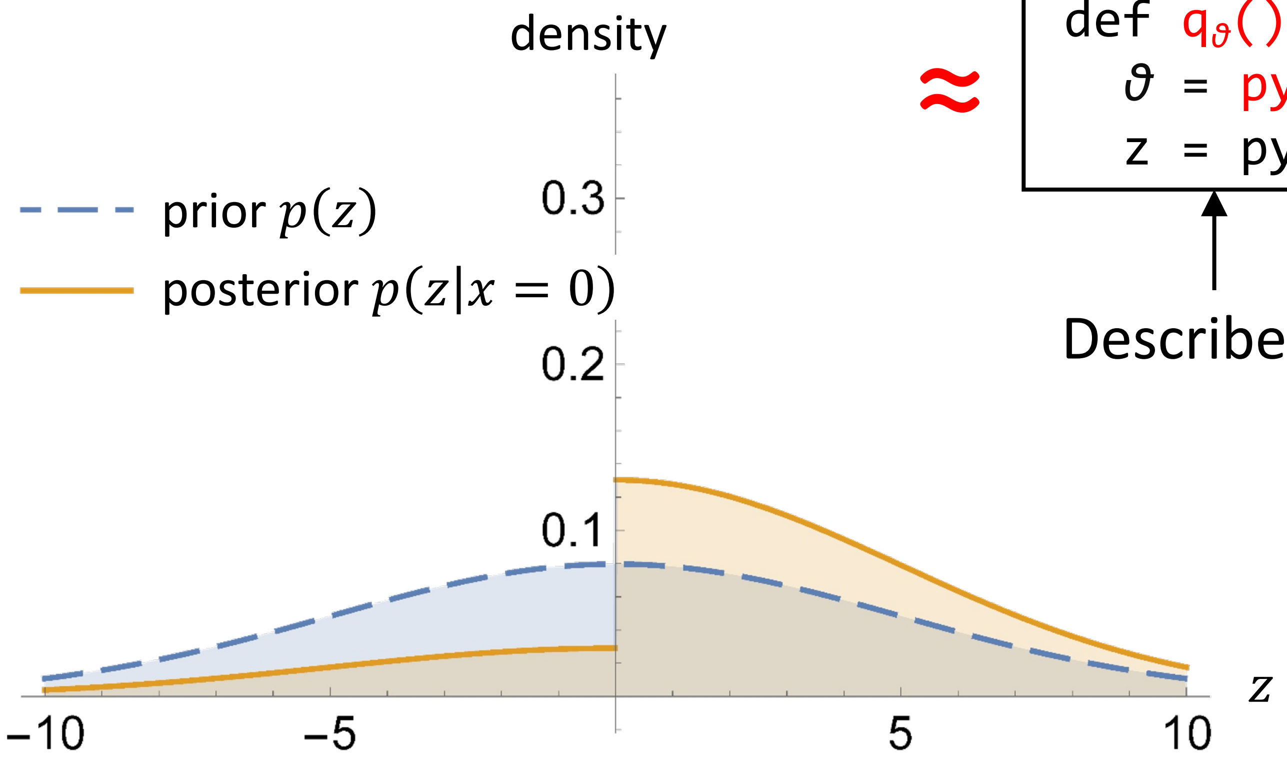
```
def p(): # model
  z = pyro.sample(“z”, Normal(0., 5.))
  if (z > 0):  pyro.sample(“x”, Normal( 1., 1.), obs=0.)
  else:        pyro.sample(“x”, Normal(-2., 1.), obs=0.)
```



# Variational Inference

- Example:

```
def p(): # model
  z = pyro.sample("z", Normal(0., 5.))
  if (z > 0):  pyro.sample("x", Normal( 1., 1.), obs=0.)
  else:        pyro.sample("x", Normal(-2., 1.), obs=0.)
```




≈

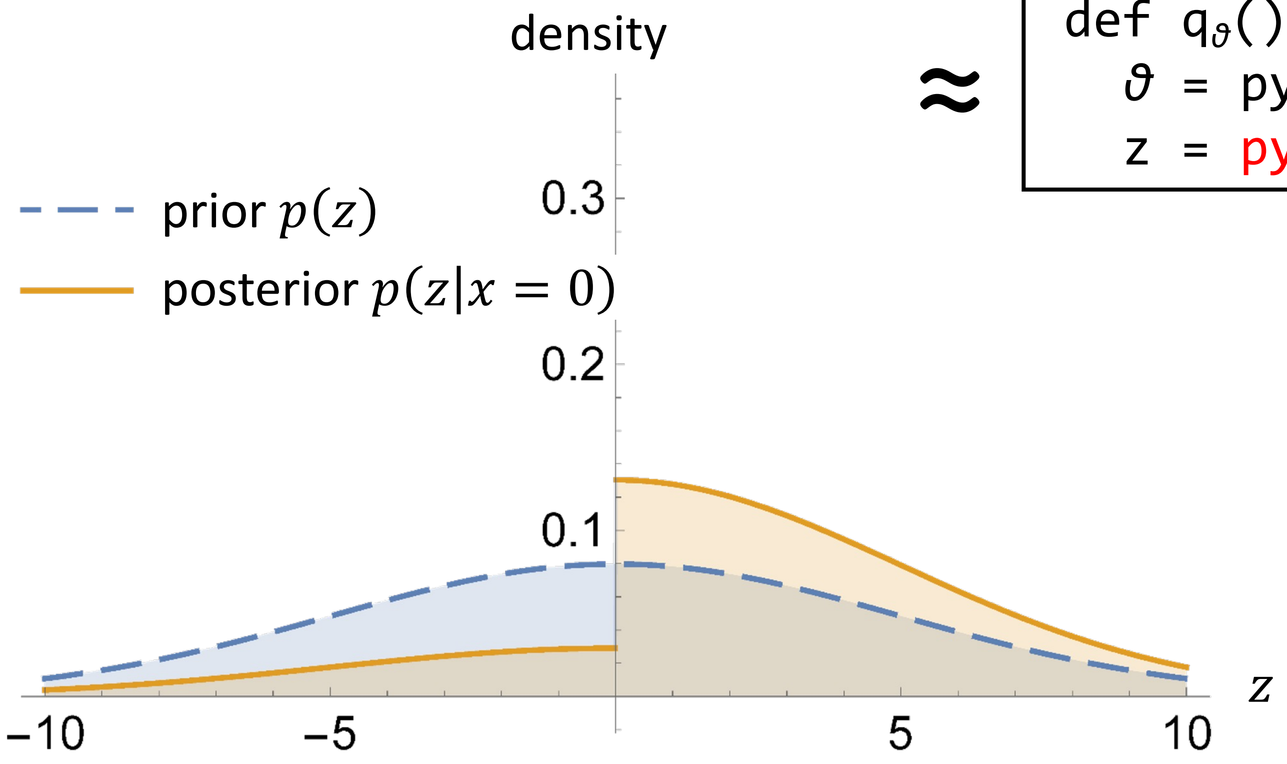
```
def qϑ(): # guide
  ϑ = pyro.param("ϑ", 1.)
  z = pyro.sample("z", Normal(ϑ, 1.))
```

Describes a family of approx. distributions.

# Variational Inference

- Example:

```
def p(): # model
  z = pyro.sample("z", Normal(0., 5.))
  if (z > 0):  pyro.sample("x", Normal( 1., 1.), obs=0.)
  else:        pyro.sample("x", Normal(-2., 1.), obs=0.)
```



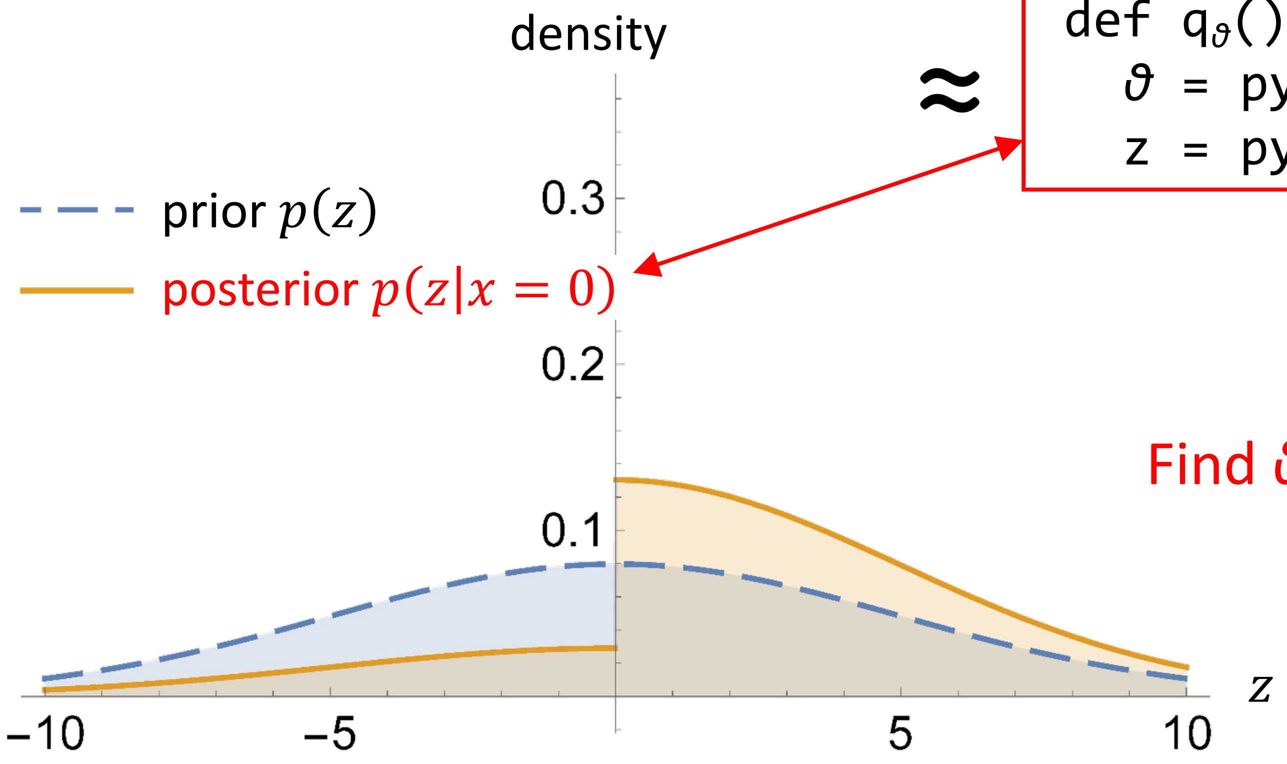

≈

```
def qϑ(): # guide
  ϑ = pyro.param("ϑ", 1.)
  z = pyro.sample("z", Normal(ϑ, 1.))
```

# Variational Inference

- Example:

```
def p(): # model
  z = pyro.sample("z", Normal(0., 5.))
  if (z > 0):  pyro.sample("x", Normal( 1., 1.), obs=0.)
  else:        pyro.sample("x", Normal(-2., 1.), obs=0.)
```




≈

```
def qϑ(): # guide
  ϑ = pyro.param("ϑ", 1.)
  z = pyro.sample("z", Normal(ϑ, 1.))
```

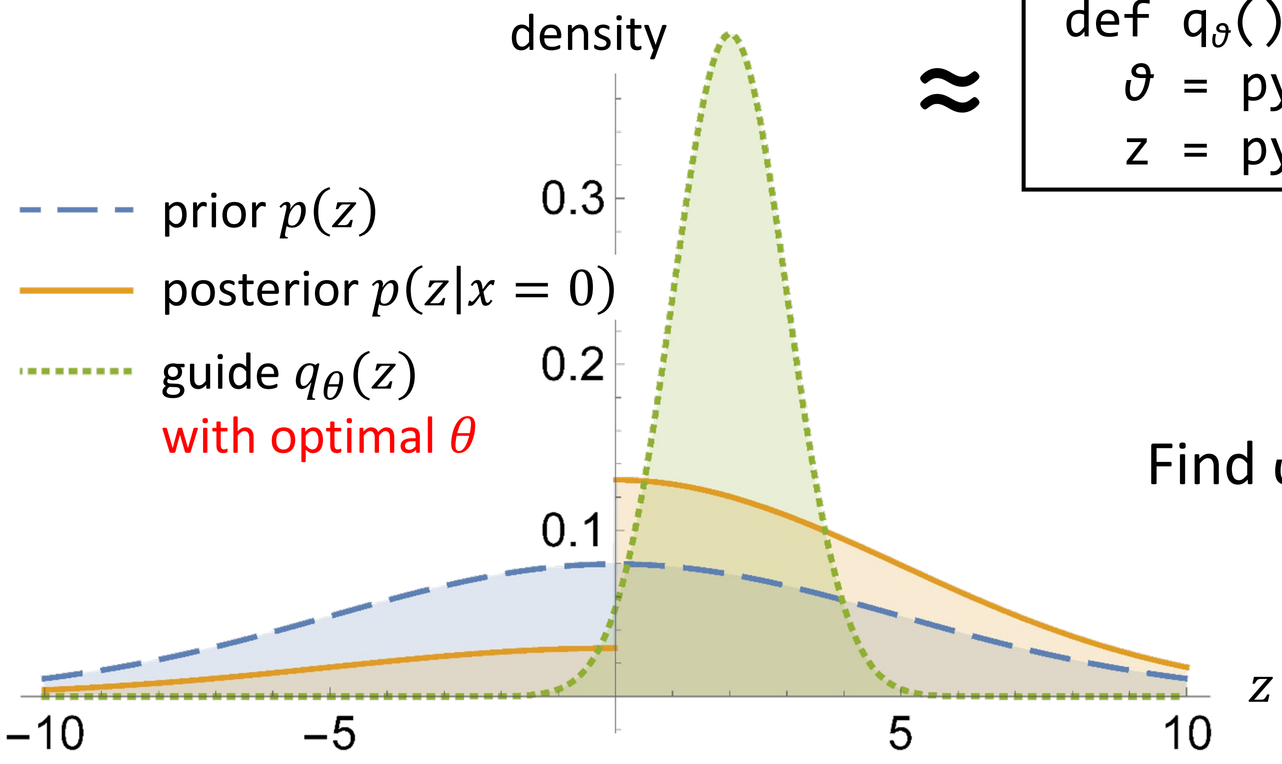
**Variational inference:**

Find $\vartheta$ that minimizes this distance.

# Variational Inference

- Example:

```
def p(): # model
  z = pyro.sample("z", Normal(0., 5.))
  if (z > 0):  pyro.sample("x", Normal( 1., 1.), obs=0.)
  else:        pyro.sample("x", Normal(-2., 1.), obs=0.)
```




$\approx$

```
def qϑ(): # guide
  ϑ = pyro.param("ϑ", 1.)
  z = pyro.sample("z", Normal(ϑ, 1.))
```

**Variational inference:**

Find $\vartheta$ that minimizes this distance.

# Variational Inference: Details

- Goal: Given a model $p(z, x)$ and a guide $q_\theta(z)$,

$$\text{minimize} \quad \mathcal{L}(\theta) \triangleq \underbrace{\mathbb{E}_{q_\theta(z)}\left[\log \frac{q_\theta(z)}{p(z,x)}\right]}_{= \,\mathrm{dist}(p, q_\theta) + \mathrm{const}} \quad \text{over } \theta \in \mathbb{R}^n.$$

# Variational Inference: Details

- Goal: Given a model $p(z, x)$ and a guide $q_\theta(z)$,

$$\text{minimize} \quad \mathcal{L}(\theta) \triangleq \underbrace{\mathbb{E}_{q_\theta(z)}\left[\log \frac{q_\theta(z)}{p(z,x)}\right]}_{= \operatorname{dist}(p, q_\theta) + \operatorname{const}} \quad \text{over } \theta \in \mathbb{R}^n.$$

- Typical approach: Apply a gradient descent algorithm.

$$\theta_{t+1} = \theta_t - \eta \cdot \nabla_\theta \mathcal{L}(\theta_t).$$

# Variational Inference: Details

- Goal: Given a model $p(z,x)$ and a guide $q_\theta(z)$,

$$\text{minimize} \quad \mathcal{L}(\theta) \triangleq \underbrace{\mathbb{E}_{q_\theta(z)}\left[\log \frac{q_\theta(z)}{p(z,x)}\right]}_{= \, \mathrm{dist}(p,\, q_\theta) + \mathrm{const}} \quad \text{over } \theta \in \mathbb{R}^n.$$

- Typical approach: Apply a gradient descent algorithm.

$$\theta_{t+1} = \theta_t - \eta \cdot \widehat{\nabla_\theta \mathcal{L}(\theta_t)}.$$

Difficult to compute exactly.

So we usually estimate it.

# Gradient Estimators

- Score estimator (≈ basic):

$$\widehat{\nabla_\theta \mathcal{L}(\theta)} = \cdots \qquad \text{for } z \sim q_\theta.$$

- Pathwise gradient estimator (≈ more accurate):

$$\widehat{\nabla_\theta \mathcal{L}(\theta)} = \cdots \qquad \text{for } z \sim r.$$

# Gradient Estimators

- Score estimator (≈ basic):

$$\widehat{\nabla_\theta \mathcal{L}(\theta)} = \log \frac{q_\theta(z)}{p(z,x)} \times \nabla_\theta (\log q_\theta(z)) \qquad \text{for } z \sim q_\theta.$$

- Pathwise gradient estimator (≈ more accurate):

$$\widehat{\nabla_\theta \mathcal{L}(\theta)} = \nabla_\theta \left(\log \frac{q_\theta(t_\theta(z))}{p(t_\theta(z),x)}\right) \qquad \text{for } z \sim r.$$

# Gradient Estimators

- Score estimator (≈ basic):

$$\widehat{\nabla_\theta \mathcal{L}(\theta)} = \log \frac{q_\theta(z)}{p(z,x)} \times \nabla_\theta(\log q_\theta(z)) \qquad \text{for } z \sim q_\theta.$$

  - Required: $q_\theta(z)$ should be differentiable in $\theta, \cdots$.

- Pathwise gradient estimator (≈ more accurate):

$$\widehat{\nabla_\theta \mathcal{L}(\theta)} = \nabla_\theta \left(\log \frac{q_\theta\big(t_\theta(z)\big)}{p(t_\theta(z),x)}\right) \qquad \text{for } z \sim r.$$

  - Required: $q_\theta(z)$ and $p(z, x)$ should be differentiable in $\theta$ and $z, \cdots$.

# Gradient Estimators

- Score estimator (≈ basic):

$$\widehat{\nabla_\theta \mathcal{L}(\theta)} = \log \frac{q_\theta(z)}{p(z,x)} \times \nabla_\theta(\log q_\theta(z)) \qquad \text{for } z \sim q_\theta.$$

  - Required: $q_\theta(z)$ should be differentiable in $\theta$, $\cdots$.

- Pathwise gradient estimator (≈ more accurate):

$$\widehat{\nabla_\theta \mathcal{L}(\theta)} = \nabla_\theta \left(\log \frac{q_\theta(t_\theta(z))}{p(t_\theta(z),x)}\right) \qquad \text{for } z \sim r.$$

  - Required: $q_\theta(z)$ and $p(z,x)$ should be differentiable in $\theta$ and $z$, $\cdots$.

- Soundness: $\mathbb{E}_z[\widehat{\nabla_\theta \mathcal{L}(\theta)}] = \nabla_\theta \mathcal{L}(\theta)$ if all requirements are satisfied.

# Gradient Estimators: General Case

expressed by programs (with if-else, ...)

- Score estimator (≈ basic):

$$\widehat{\nabla_\theta \mathcal{L}(\theta)} = \log \frac{q_\theta(z)}{p(z,x)} \times \nabla_\theta(\log q_\theta(z)) \qquad \text{for } z \sim q_\theta.$$

  - Required: $q_\theta(z)$ should be differentiable in $\theta$, $\cdots$.

- Pathwise gradient estimator (≈ more accurate):

$$\widehat{\nabla_\theta \mathcal{L}(\theta)} = \nabla_\theta \left(\log \frac{q_\theta(t_\theta(z))}{p(t_\theta(z),x)}\right) \qquad \text{for } z \sim r.$$

  - Required: $q_\theta(z)$ and $p(z,x)$ should be differentiable in $\theta$ and $z$, $\cdots$.

- Soundness: $\mathbb{E}_z[\widehat{\nabla_\theta \mathcal{L}(\theta)}] = \nabla_\theta \mathcal{L}(\theta)$ if all requirements are ~~satisfied~~.

# Gradient Estimators: General Case

expressed by programs (with if-else, ...)

- Score estimator (≈ basic):

$$\widehat{\nabla_\theta \mathcal{L}(\theta)} = \log \frac{q_\theta(z)}{p(z,x)} \times \nabla_\theta(\log q_\theta(z)) \qquad \text{for } z \sim q_\theta.$$

  - Required: $q_\theta(z)$ should be differentiable in $\theta$, $\cdots$.

- Pathwise gradient estimator (≈ more accurate):

$$\widehat{\nabla_\theta \cdots} \qquad \text{for } z \sim r.$$

Still holds?

  - Required: $q_\theta(z)$ and $p(z,x)$ should be differentiable in $\theta$ and $z$, $\cdots$.

- Soundness: $\mathbb{E}_z[\widehat{\nabla_\theta \mathcal{L}(\theta)}] = \nabla_\theta \mathcal{L}(\theta)$ if all requirements are ~~satisfied~~.

# Gradient Estimators: General Case

- Score estimator (≈ basic one):

**We studied this question in [NeurIPS'18]. Answer: No!**

  - Required: $q_\theta(z)$ should be differentiable in $\theta$, $\cdots$.

- Pathwise gradient estimator (≈ more accurate):

$\widehat{\nabla_\theta}$ 

**Still holds?**

for $z \sim r$.

  - Required: $q_\theta(z)$ and $p(z, x)$ should be differentiable in $\theta$ and $z$, $\cdots$.

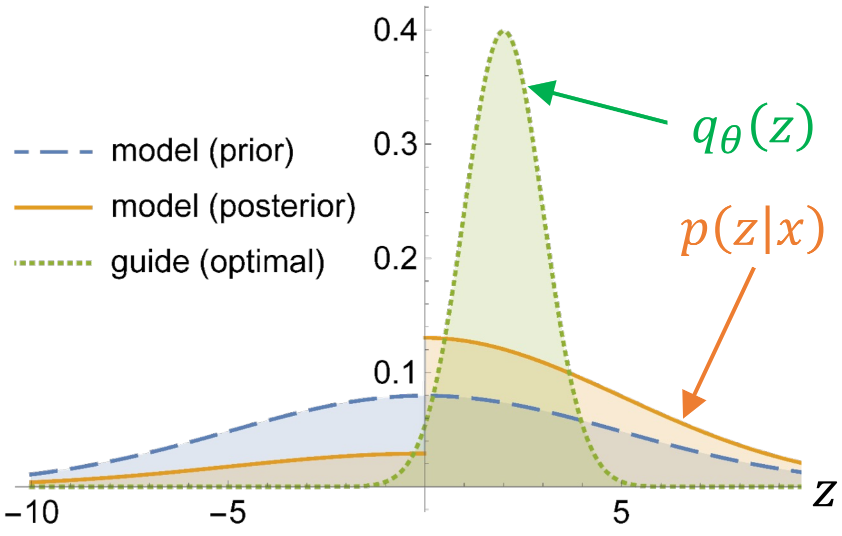
- Soundness: $\mathbb{E}_z[\widehat{\nabla_\theta \mathcal{L}(\theta)}] = \nabla_\theta \mathcal{L}(\theta)$ if all requirements are ~~satisfied~~.

# Gradient Estimators: General Case

- Previous example:

```
def p(): # model
  z = sample("z", Normal(0, 5))
  if (z > 0): sample("x", Normal( 1, 1), obs=0)
  else:       sample("x", Normal(-2, 1), obs=0)
```

```
def qϑ(): # guide
  ϑ = pyro.param("ϑ", 1)
  z = pyro.sample("z", Normal(ϑ, 1))
```

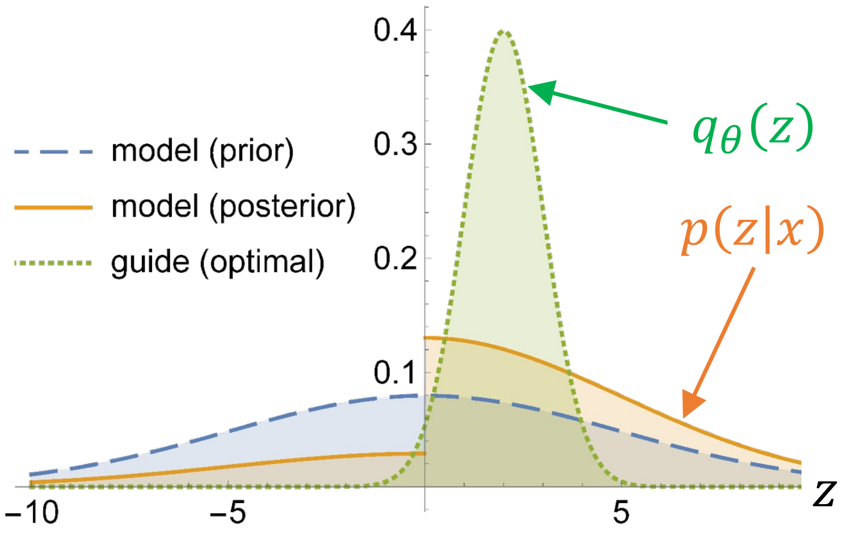


# Gradient Estimators: General Case

- Previous example:

```
def p(): # model
  z = sample("z", Normal(0, 5))
  if (z > 0): sample("x", Normal( 1, 1), obs=0)
  else:       sample("x", Normal(-2, 1), obs=0)
```

```
def qϑ(): # guide
  ϑ = pyro.param("ϑ", 1)
  z = pyro.sample("z", Normal(ϑ, 1))
```




$q_\theta(z)$: differentiable in $\theta$ and $z$.

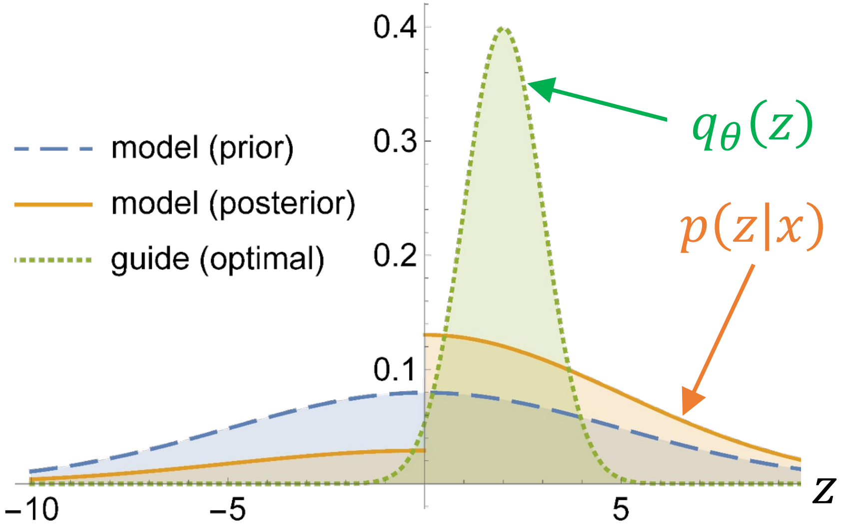
$p(z,x)$: non-differentiable in $z$.

# Gradient Estimators: General Case

- Previous example:

```
def p(): # model
  z = sample("z", Normal(0, 5))
  if (z > 0): sample("x", Normal( 1, 1), obs=0)
  else:       sample("x", Normal(-2, 1), obs=0)
```

```
def qϑ(): # guide
  ϑ = pyro.param("ϑ", 1)
  z = pyro.sample("z", Normal(ϑ, 1))
```




$q_\theta(z)$: differentiable in $\theta$ and $z$.

$p(z, x)$: non-differentiable in $z$.

$\Longrightarrow$

Score estimator: sound.

Pathwise gradient estimator: unsound.

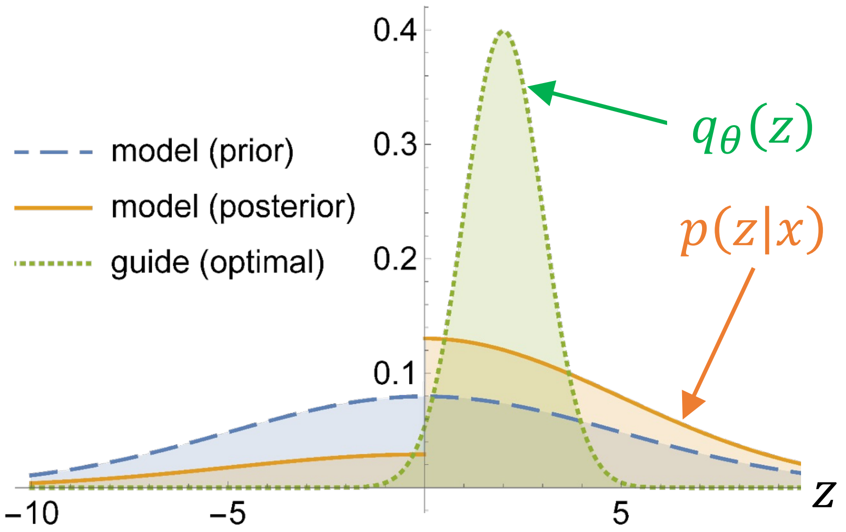
$\widehat{\nabla_\theta \mathcal{L}(\theta)} = 0$ for all $\theta$.

# Gradient Estimators: General Case

- Previous example:

If $f$ is non-differentiable in $\theta$, then in general

$$\nabla_\theta \int f_\theta(t)dt \neq \int \nabla_\theta f_\theta(t)dt\,.$$




$q_\theta(z)$: differentiable in $\theta$ and $z$.
$p(z,x)$: non-differentiable in $z$.

$\Longrightarrow$

Score estimator: sound.
Pathwise gradient estimator: unsound.

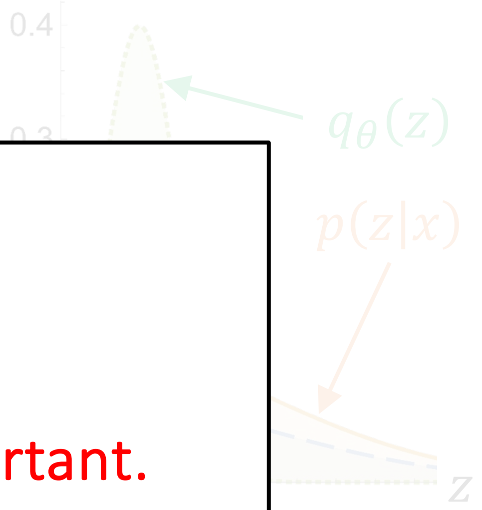
$\widehat{\nabla_\theta \mathcal{L}(\theta)} = 0$ for all $\theta$.

# Gradient Estimators: General Case

- Previous example:

```
def p(): # model
  z = sample("z", Normal(0, 5))
  if
  el
```

```
def (
  ϑ =
  z =
```




Score estimator: Accuracy ↓, Requirements ↓.

Path. grad. estimator: Accuracy ↑, Requirements ↑.

Differentiability requirement is important.

$q_\theta(z)$: differentiable in $\theta$ and $z$.

$p(z,x)$: non-differentiable in $z$.

Score estimator: correct.

Pathwise gradient estimator: incorrect.

$\widehat{\nabla_\theta \mathcal{L}(\theta)} = 0$ for all $\theta$. Pyro computes this.

# Gradient Estimators: General Case

How to maximize the use of path. grad. estimator,
while remaining **sound** for **general** programs?

# Gradient Estimators: General Case

How to maximize the use of path. grad. estimator,
while remaining **sound** for **general** programs?

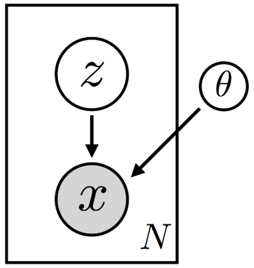
Our **automatic** approach [POPL'23]. (≈ [Lew+23])

1. **Smoothness analysis:** Identify differentiable parts of a model/guide.
2. **Selective application:** Apply path. grad. est. only to these parts.

Part 1: Smoothness in Probabilistic Programming
[NeurIPS'18]

→ Part 2: Smoothness Analysis of Programs
[POPL'23]

# Why Need Smoothness?




differentiable

probabilistic model

We can apply more efficient inference algorithms.

$$\nabla_{\boldsymbol{\theta}} \mathbb{E}_{p(\mathbf{x};\boldsymbol{\theta})}\left[f(\mathbf{x})\right]$$
$$= \mathbb{E}_{p(\boldsymbol{\epsilon})}\left[\nabla_{\boldsymbol{\theta}} f(g(\boldsymbol{\epsilon};\boldsymbol{\theta}))\right]$$

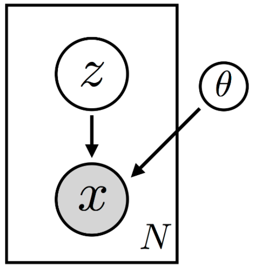
pathwise gradient estimator

$$\frac{dq_i}{dt} = \frac{\partial H}{\partial p_i}$$
$$\frac{dp_i}{dt} = -\frac{\partial H}{\partial q_i}$$

Hamiltonian Monte Carlo

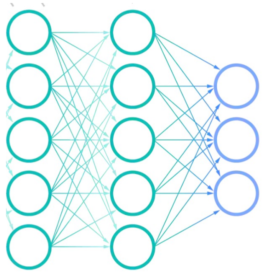
…

# Why Need Smoothness?




differentiable

probabilistic model



Lipschitz continuous

neural network

cyber-physical systems, ⋯

We can apply more efficient inference algorithms.

$$\nabla_{\boldsymbol{\theta}} \mathbb{E}_{p(\mathbf{x};\boldsymbol{\theta})}\left[f(\mathbf{x})\right]$$
$$= \mathbb{E}_{p(\boldsymbol{\epsilon})}\left[\nabla_{\boldsymbol{\theta}} f(g(\boldsymbol{\epsilon};\boldsymbol{\theta}))\right]$$

pathwise gradient estimator

$$\frac{dq_i}{dt} = \frac{\partial H}{\partial p_i}$$
$$\frac{dp_i}{dt} = -\frac{\partial H}{\partial q_i}$$

Hamiltonian Monte Carlo

⋯

We can provide provable robustness.

We can give guaranteed generalization bounds.

…

We can describe and solve differential equations.

…

# Why Need Smoothness?

Goal: Find out smoothness properties of **programs, automatically** and **soundly.**

probabilistic model

expressed by programs

neural network

cyber-physical systems, ⋯

$= \mathbb{E}_{p(\epsilon)}\left[\nabla_{\theta} f(g(\epsilon;\theta))\right]$

pathwise gradient estimator

Hamiltonian Monte Carlo

⋯

We can provide provable robustness.

We can give guaranteed generalization bounds.

⋯

We can describe and solve differential equations.

⋯

# Smoothness Properties of Programs

Smoothness = differentiability.   Programs = deterministic, imperative programs.

```
P ≜ ( y:=exp(x) ; if (x>0) {z:=1} else {z:=-1} )
```

# Smoothness Properties of Programs

Smoothness = differentiability. F real-valued terministic, imperative programs.

```
P ≜ ( y:=exp(x) ; if (x>0) {z:=1} else {z:=-1} )
```

- $[\![\mathrm{P}]\!] \quad : \quad \mathbb{R}^3 \quad \to \quad \mathbb{R}^3$

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} \mapsto \begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} x \\ \exp(x) \\ \mathrm{sgn}(x) \end{pmatrix}$$

# Smoothness Properties of Programs

Smoothness = differentiability.   Programs = deterministic, imperative programs.

```
P ≜ ( y:=exp(x) ; if (x>0) {z:=1} else {z:=-1} )
```

- $[\![\mathrm{P}]\!] \quad : \quad \mathbb{R}^3 \quad \rightarrow \quad \mathbb{R}^3$

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} \mapsto \begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} x \\ \exp(x) \\ \mathrm{sgn}(x) \end{pmatrix}$$

- P $\vdash$ $y$ is differentiable in $x$

$\iff$ …

# Smoothness Properties of Programs

Smoothness = differentiability. Programs = deterministic, imperative programs.

```
P ≜ ( y:=exp(x) ; if (x>0) {z:=1} else {z:=-1} )
```

- $[\![P]\!] \quad : \quad \mathbb{R}^3 \quad \to \quad \mathbb{R}^3$

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} \mapsto \begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} x \\ \exp(x) \\ \mathrm{sgn}(x) \end{pmatrix}$$

- P $\vdash$ $y$ is differentiable $\boxed{\text{in } x}$

$\Longleftrightarrow$

for all $y_0, z_0 \in \mathbb{R}$.

$$\begin{pmatrix} \boxed{x} \\ y_0 \\ z_0 \end{pmatrix} \xrightarrow{[\![P]\!]} \begin{pmatrix} x' \\ y' \\ z' \end{pmatrix}$$

# Smoothness Properties of Programs

Smoothness = differentiability. Programs = deterministic, imperative programs.

```
P ≜ ( y:=exp(x) ; if (x>0) {z:=1} else {z:=-1} )
```

- $[\![\mathrm{P}]\!] \quad : \quad \mathbb{R}^3 \quad \to \quad \mathbb{R}^3$

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} \mapsto \begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} x \\ \exp(x) \\ \mathrm{sgn}(x) \end{pmatrix}$$

- P $\vdash$ $\boxed{y \text{ is}}$ differentiable $\boxed{\text{in } x}$

$\iff f : \quad \mathbb{R} \quad \to \quad \mathbb{R}$ for all $y_0, z_0 \in \mathbb{R}$.

$$\begin{pmatrix} x \\ y_0 \\ z_0 \end{pmatrix} \overset{[\![\mathrm{P}]\!]}{\longmapsto} \begin{pmatrix} x' \\ \boxed{y'} \\ z' \end{pmatrix}$$

# Smoothness Properties of Programs

Smoothness = differentiability. Programs = deterministic, imperative programs.

```
P ≜ ( y:=exp(x) ; if (x>0) {z:=1} else {z:=-1} )
```

- $[\![\mathrm{P}]\!] \;:\; \mathbb{R}^3 \;\rightarrow\; \mathbb{R}^3$

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} \mapsto \begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} x \\ \exp(x) \\ \operatorname{sgn}(x) \end{pmatrix}$$

- P $\vdash$ $y$ is differentiable in $x$

$\Longleftrightarrow$ $f : \;\mathbb{R} \;\rightarrow\; \mathbb{R}$ is differentiable for all $y_0, z_0 \in \mathbb{R}$.

$$\begin{pmatrix} \boxed{x} \\ y_0 \\ z_0 \end{pmatrix} \overset{[\![\mathrm{P}]\!]}{\longmapsto} \begin{pmatrix} x' \\ \boxed{y'} \\ z' \end{pmatrix}$$

# Smoothness Properties of Programs

Smoothness = differentiability.   Programs = deterministic, imperative programs.

```
P ≜ ( y:=exp(x) ; if (x>0) {z:=1} else {z:=-1} )
```

- $[\![\mathrm{P}]\!] \ : \ \mathbb{R}^3 \ \rightarrow \ \mathbb{R}^3$

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} \mapsto \begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} x \\ \exp(x) \\ \operatorname{sgn}(x) \end{pmatrix}$$

- P $\vdash$ $y$ is differentiable in $x$.

  P $\nvdash$ $z$ is differentiable in $x$.

  …

# Smoothness Properties of Programs

Smoothness = differentiability. Programs = deterministic, imperative programs.

```
P ≜ ( y:=exp(x) ; if (x>0) {z:=1} else {z:=-1} )
```

It is surprisingly **subtle** to find out such smoothness properties

(1) **automatically,**

(2) **soundly,** and

(3) **precisely enough.**

# Differentiability Analysis

Want to check differentiability of a program in a **compositional way.**

**Case for seq-composition:** ⋯

**Case for if-else:** ⋯

**Case for while:** ⋯

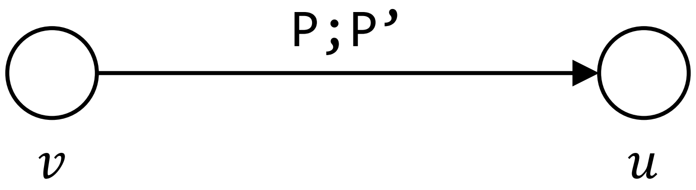
…

# Differentiability Analysis

Want to check differentiability of a program in a **compositional way.**

**Case for seq-composition:** Given P, P' and $U, V \subseteq \mathrm{Var}$,

$$\forall u \in U, \forall v \in V. \quad \text{P;P'} \vdash u \text{ is differentiable in } v \ ?$$




**Case for if-else:** $\cdots$

…

# Differentiability Analysis

Based on the chain rule:

$$\exists T \subseteq \mathrm{Var}. \quad \frac{\begin{array}{lll} \forall u \in U, \forall t \in T. & \mathrm{P'} & \vdash u \text{ is differentiable in } t \\ \forall t \in T, \forall v \in V. & \mathrm{P} & \vdash t \text{ is differentiable in } v \end{array}}{\begin{array}{lll} \forall u \in U, \forall v \in V. & \mathrm{P;P'} & \vdash u \text{ is differentiable in } v \end{array}}$$

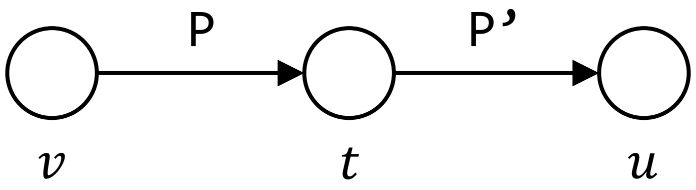
$$v \xrightarrow{\mathrm{P}} t \xrightarrow{\mathrm{P'}} u$$

# Differentiability Analysis

Looks sound by chain rule.
Previously proposed & studied
(e.g., [CACM’12] for continuity).

Based on the chain rule:

$$\exists T \subseteq \text{Var}. \quad \frac{\begin{array}{lrl} \forall u \in U, \forall t \in T. & \text{P'} & \vdash u \text{ is differentiable in } t \\ \forall t \in T, \forall v \in V. & \text{P} & \vdash t \text{ is differentiable in } v \end{array}}{\begin{array}{lrl} \forall u \in U, \forall v \in V. & \text{P};\text{P'} & \vdash u \text{ is differentiable in } v \end{array}} \;\text{Seq}$$



# Differentiability Analysis

Looks sound by chain rule.
Previously proposed & studied
(e.g., [CACM'12] for continuity).

Based on the chain rule:

$$\exists T \subseteq \mathrm{Var}. \quad \frac{\begin{array}{lll} \forall u \in U, \forall t \in T. & \mathrm{P'} & \vdash u \text{ is differentiable in } t \\ \forall t \in T, \forall v \in V. & \mathrm{P} & \vdash t \text{ is differentiable in } v \end{array}}{\begin{array}{lll} \forall u \in U, \forall v \in V. & \mathrm{P;P'} & \vdash u \text{ is differentiable in } v \end{array}} \ \text{Seq}$$

Is this rule indeed sound?

# Soundness Issue

`P;P’ ≜ (y:=sgn(x) ; z:=x+y).`

$$\dfrac{\begin{array}{lrl} \forall u \in U, \forall\, t \in T. & \mathrm{P'} & \vdash\ u \text{ is differentiable in } t \\ \forall\, t \in T, \forall v \in V. & \mathrm{P} & \vdash\ t \text{ is differentiable in } v \end{array}}{\begin{array}{lrl} \forall u \in U, \forall v \in V. & \mathrm{P;P'} & \vdash\ u \text{ is differentiable in } v \end{array}}\ \text{Seq}$$

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} \underset{[\![\mathrm{P}]\!]}{\longmapsto} \begin{pmatrix} x \\ \mathrm{sgn}(x) \\ z \end{pmatrix}, \qquad \begin{pmatrix} x \\ y \\ z \end{pmatrix} \underset{[\![\mathrm{P'}]\!]}{\longmapsto} \begin{pmatrix} x \\ y \\ x+y \end{pmatrix}$$

# Soundness Issue

`P;P' ≜ (y:=sgn(x) ; z:=x+y).` $U \triangleq \{z\}, T \triangleq \{x\}, V \triangleq \{x\}.$

$$\dfrac{\begin{array}{lrl} \forall u \in U, \forall t \in T. & \mathtt{P'} & \vdash u \text{ is differentiable in } t \\ \forall t \in T, \forall v \in V. & \mathtt{P} & \vdash t \text{ is differentiable in } v \end{array}}{\begin{array}{lrl} \forall u \in U, \forall v \in V. & \mathtt{P;P'} & \vdash u \text{ is differentiable in } v \end{array}}\ \text{Seq}$$

$$\begin{pmatrix} \boxed{x} \\ y \\ z \end{pmatrix} \xmapsto[[\![\mathtt{P}]\!]]{} \begin{pmatrix} \boxed{x} \\ \mathrm{sgn}(x) \\ z \end{pmatrix}, \quad \begin{pmatrix} \boxed{x} \\ y \\ z \end{pmatrix} \xmapsto[[\![\mathtt{P'}]\!]]{} \begin{pmatrix} x \\ y \\ \boxed{x+y} \end{pmatrix}$$

# Soundness Issue

P;P’ ≜ (y:=sgn(x) ; z:=x+y).  $U \triangleq \{z\}, T \triangleq \{x\}, V \triangleq \{x\}.$

$$\dfrac{\begin{array}{lll} \forall u \in U, \forall\, t \in T. & \text{P'} & \vdash u \text{ is differentiable in } t \\ \forall\, t \in T, \forall v \in V. & \text{P} & \vdash t \text{ is differentiable in } v \end{array}}{\begin{array}{lll} \forall u \in U, \forall v \in V. & \text{P;P'} & \vdash u \text{ is differentiable in } v \end{array}}\ \text{Seq}$$

$$\begin{pmatrix} \boxed{x} \\ y \\ z \end{pmatrix} \underset{[\![\text{P}]\!]}{\longmapsto} \begin{pmatrix} x \\ \mathrm{sgn}(x) \\ z \end{pmatrix} \underset{[\![\text{P'}]\!]}{\longmapsto} \begin{pmatrix} x \\ \mathrm{sgn}(x) \\ \boxed{x + \mathrm{sgn}(x)} \end{pmatrix}$$

# Soundness Issue

`P;P’ ≜ (y:=sgn(x) ; z:=x+y).` $U \triangleq \{z\}, T \triangleq \{x\}, V \triangleq \{x\}.$

$$
\dfrac{
\begin{array}{lrl}
\checkmark\ \forall u \in U, \forall t \in T. & \texttt{P'} & \vdash u \text{ is differentiable in } t \\
\checkmark\ \forall t \in T, \forall v \in V. & \texttt{P} & \vdash t \text{ is differentiable in } v
\end{array}
}{
\times\ \forall u \in U, \forall v \in V. \quad \texttt{P;P'} \vdash u \text{ is differentiable in } v
}\ \text{Seq}
$$

$$
\begin{pmatrix} \boxed{x} \\ y \\ z \end{pmatrix}
\xmapsto[\llbracket \mathrm{P} \rrbracket]{}
\begin{pmatrix} x \\ \mathrm{sgn}(x) \\ z \end{pmatrix}
\xmapsto[\llbracket \mathrm{P'} \rrbracket]{}
\begin{pmatrix} x \\ \mathrm{sgn}(x) \\ \boxed{x + \mathrm{sgn}(x)} \end{pmatrix}
$$

This rule is unsound! But why?

# Soundness Issue

`P;P’` ≜ `(y:=sgn(x) ; z:=x+y).` $U \triangleq \{z\}, T \triangleq \{x\}, V \triangleq \{x\}.$

✓ $\forall u \in U, \forall t \in T.$ P’ $\vdash$ $u$ is differentiable in $t$

✓ $\forall t \in T, \forall v \in V.$ P $\vdash$ $t$ is differentiable in $v$

---- Seq

✗ $\forall u \in U, \forall v \in V.$ P;P’ $\vdash$ $u$ is differentiable in $v$

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} \underset{[\![\text{P}]\!]}{\longmapsto} \begin{pmatrix} x \\ \operatorname{sgn}(x) \\ z \end{pmatrix} \underset{[\![\text{P'}]\!]}{\longmapsto} \begin{pmatrix} x \\ \operatorname{sgn}(x) \\ x + \operatorname{sgn}(x) \end{pmatrix}$$

# Soundness Issue

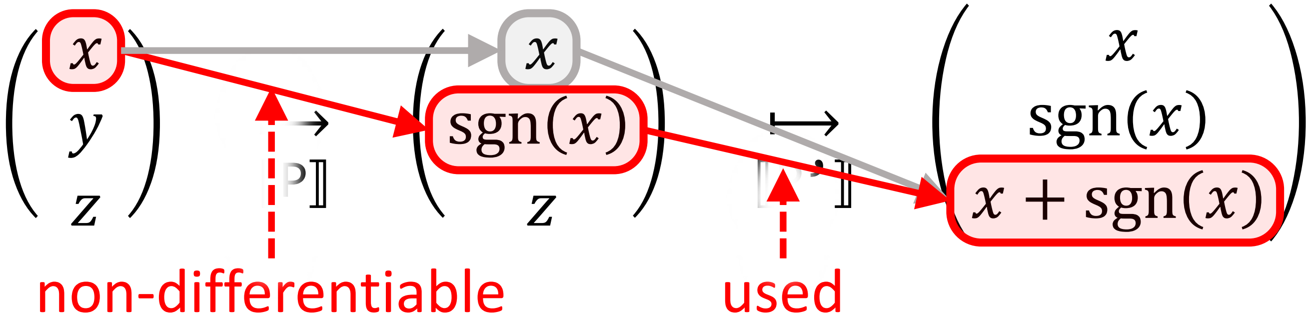
P;P’ $\triangleq$ (y:=sgn(x) ; z:=x+y). $U \triangleq \{z\}, T \triangleq \{x\}, V \triangleq \{x\}.$

✓ $\forall u \in U, \forall t \in T.$ P’ $\vdash$ $u$ is differentiable in $t$

✓ $\forall t \in T, \forall v \in V.$ P $\vdash$ $t$ is differentiable in $v$

───────────────────────────── Seq

✗ $\forall u \in U, \forall v \in V.$ P;P’ $\vdash$ $u$ is differentiable in $v$



# Soundness Issue

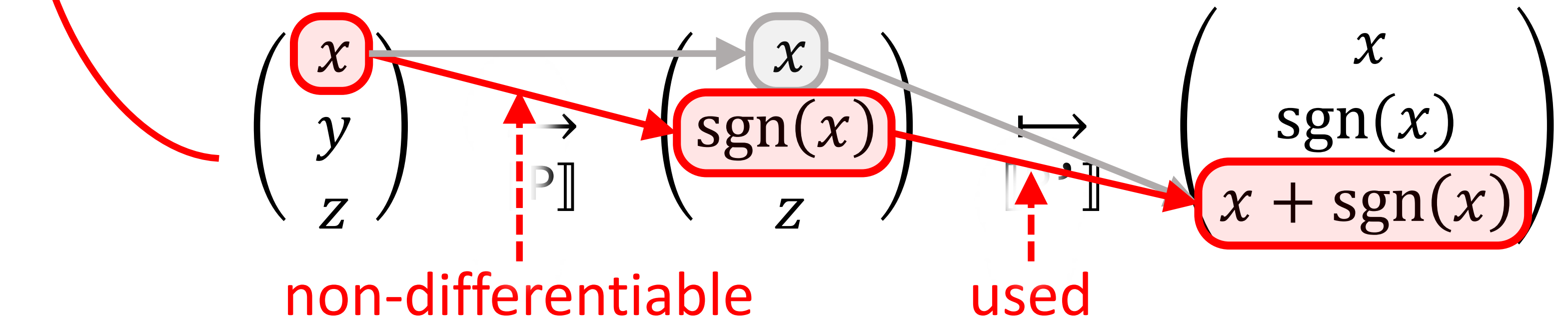
`P;P’` $\triangleq$ `(y:=sgn(x) ; z:=x+y).` $\quad U \triangleq \{z\}, T \triangleq \{x\}, V \triangleq \{x\}.$

✓ $\forall u \in U, \forall t \in T.$ P’ $\vdash$ $u$ is differentiable in $t$

✓ $\forall t \in T, \forall v \in V.$ P $\vdash$ $t$ is differentiable in $v$

---------------------------------------------------------------- Seq

✗ $\forall u \in U, \forall v \in V.$ P;P’ $\vdash$ $u$ is differentiable in $v$



# Soundness Issue

P;P' ≜ (y:=sgn(x) ; z:=x+y).   $U \triangleq \{z\}, T \triangleq \{x\}, V \triangleq \{x\}.$

$$\dfrac{\begin{array}{lrl} \forall u \in U, \forall t \in T. & \text{P'} & \vdash u \text{ is differentiable in } t \\ \forall t \in T, \forall v \in V. & \text{P} & \vdash t \text{ is differentiable in } v \end{array}}{\begin{array}{lrl} \forall u \in U, \forall v \in V. & \text{P;P'} & \vdash u \text{ is differentiable in } v \end{array}} \text{ Seq}$$

**Lesson: Need to consider dependency between variables.**

non-differentiable    used

# Possible Fix

P;P' $\triangleq$ (y:=sgn(x) ; z:=x+y). $\quad U \triangleq \{z\}$, $\cancel{T \triangleq \{x\}}$, $V \triangleq \{x\}$.

Var

$$\dfrac{\begin{array}{lll} \forall u \in U, \forall t \in \underbrace{\cancel{T}}. & \texttt{P'} & \vdash u \text{ is differentiable in } t \\ \forall t \in \underbrace{\cancel{T}}, \forall v \in V. & \texttt{P} & \vdash t \text{ is differentiable in } v \end{array}}{\forall u \in U, \forall v \in V. \quad \texttt{P;P'} \vdash u \text{ is differentiable in } v}\ \text{Seq'}$$

✗

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} \underset{[\![\texttt{P}]\!]}{\longmapsto} \begin{pmatrix} x \\ \mathrm{sgn}(x) \\ z \end{pmatrix} \underset{[\![\texttt{P'}]\!]}{\longmapsto} \begin{pmatrix} x \\ \mathrm{sgn}(x) \\ x + \mathrm{sgn}(x) \end{pmatrix}$$

# Possible Fix

`P;P'` ≜ `(y:=sgn(x) ; z:=x+y).` $U \triangleq \{z\}, \cancel{T \triangleq \{x\}}, V \triangleq \{x\}.$

Var

$$\frac{\begin{array}{lll}\forall u \in U, \forall t \in \cancel{T}. & \mathtt{P'} & \vdash u \text{ is differentiable in } t \\ \forall t \in \cancel{T}, \forall v \in V. & \mathtt{P} & \vdash t \text{ is differentiable in } v\end{array}}{\begin{array}{lll}\forall u \in U, \forall v \in V. & \mathtt{P;P'} & \vdash u \text{ is differentiable in } v\end{array}}\ \text{Seq'}$$

$$\begin{pmatrix} \boxed{x} \\ y \\ z \end{pmatrix} \xmapsto{[\![\mathtt{P}]\!]} \begin{pmatrix} x \\ \boxed{\mathrm{sgn}(x)} \\ z \end{pmatrix} \xmapsto{[\![\mathtt{P'}]\!]} \begin{pmatrix} x \\ \mathrm{sgn}(x) \\ x + \mathrm{sgn}(x) \end{pmatrix}$$

non-differentiable

# Possible Fix

`P;P’ ≜ (y:=sgn(x) ; z:=x+y).` $U \triangleq \{z\}, \cancel{T \triangleq \{x\}}, V \triangleq \{x\}.$

Var

$$\frac{\begin{array}{lll} \forall u \in U, \forall t \in \cancel{T}. & \texttt{P'} & \vdash u \text{ is differentiable in } t \\ ✕\ \forall t \in \cancel{T}, \forall v \in V. & \texttt{P} & \vdash t \text{ is differentiable in } v \end{array}}{✕\ \forall u \in U, \forall v \in V. \quad \texttt{P;P'} \vdash u \text{ is differentiable in } v}\ \text{Seq'}$$

$$\begin{pmatrix} \boxed{x} \\ y \\ z \end{pmatrix} \xmapsto{[\![\texttt{P}]\!]} \begin{pmatrix} x \\ \boxed{\mathrm{sgn}(x)} \\ z \end{pmatrix} \xmapsto{[\![\texttt{P'}]\!]} \begin{pmatrix} x \\ \mathrm{sgn}(x) \\ x + \mathrm{sgn}(x) \end{pmatrix}$$

non-differentiable

This rule now looks sound. Right...?

# Soundness Issue (Again)

`P;P’ ≜ (y:=x ; z:=f(x,y))` for $f(x,y) \triangleq \begin{cases} xy/(x^2+y^2) & \text{if } (x,y) \neq (0,0) \\ 0 & \text{if } (x,y) = (0,0). \end{cases}$

$$\dfrac{\begin{array}{lll} \forall u \in U,\ \forall t \in \mathrm{Var}. & \texttt{P'} & \vdash u \text{ is differentiable in } t \\ \forall t \in \mathrm{Var}, \forall v \in V. & \texttt{P} & \vdash t \text{ is differentiable in } v \end{array}}{\forall u \in U, \forall v \in V.\quad \texttt{P;P'} \vdash u \text{ is differentiable in } v}\ \text{Seq'}$$

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} \xmapsto[\llbracket \mathrm{P} \rrbracket]{} \begin{pmatrix} x \\ x \\ z \end{pmatrix} \qquad \begin{pmatrix} x \\ y \\ z \end{pmatrix} \xmapsto[\llbracket \mathrm{P'} \rrbracket]{} \begin{pmatrix} x \\ y \\ f(x,y) \end{pmatrix}$$

# Soundness Issue (Again)

P;P’ $\triangleq$ (y:=x ; z:=f(x,y)) for $f(x,y) \triangleq \begin{cases} xy/(x^2+y^2) & \text{if } (x,y) \neq (0,0) \\ 0 & \text{if } (x,y) = (0,0). \end{cases}$

$U \triangleq \{z\}, V \triangleq \{x\}.$

$$\dfrac{\begin{array}{llll} \checkmark & \forall u \in U,\ \ \forall t \in \mathrm{Var}. & \text{P'} & \vdash u \text{ is differentiable in } t \\ \checkmark & \forall t \in \mathrm{Var}, \forall v \in V. & \text{P} & \vdash t \text{ is differentiable in } v \end{array}}{\forall u \in U, \forall v \in V. \quad \text{P;P'} \vdash u \text{ is differentiable in } v}\ \text{Seq'}$$

$$\begin{pmatrix} \boxed{x} \\ y \\ z \end{pmatrix} \underset{[\![\text{P}]\!]}{\longmapsto} \begin{pmatrix} \boxed{\begin{matrix} x \\ x \\ z \end{matrix}} \end{pmatrix} \qquad \begin{pmatrix} \boxed{\begin{matrix} x \\ y \\ z \end{matrix}} \end{pmatrix} \underset{[\![\text{P'}]\!]}{\longmapsto} \begin{pmatrix} x \\ y \\ \boxed{f(x,y)} \end{pmatrix}$$

# Soundness Issue (Again)

$\texttt{P;P'} \triangleq \texttt{(y:=x ; z:=f(x,y))}$ for $f(x,y) \triangleq \begin{cases} xy/(x^2+y^2) & \text{if } (x,y) \neq (0,0) \\ 0 & \text{if } (x,y) = (0,0). \end{cases}$

$U \triangleq \{z\}, V \triangleq \{x\}.$

$$\frac{\begin{array}{ll} \checkmark & \forall u \in U,\ \forall t \in \mathrm{Var}.\quad \texttt{P'} \vdash u \text{ is differentiable in } t \\ \checkmark & \forall t \in \mathrm{Var}, \forall v \in V.\quad \texttt{P} \vdash t \text{ is differentiable in } v \end{array}}{\times \quad \forall u \in U, \forall v \in V.\quad \texttt{P;P'} \vdash u \text{ is differentiable in } v}\ \text{Seq'}$$

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} \xmapsto[{[\![\mathrm{P}]\!]}]{} \begin{pmatrix} x \\ x \\ z \end{pmatrix} \xmapsto[{[\![\mathrm{P'}]\!]}]{} \begin{pmatrix} x \\ x \\ \boxed{1[x \neq 0] \cdot \frac{1}{2}} \end{pmatrix}$$

# Soundness Issue (Again)

P;P' $\triangleq$ (y:=x ; z:=f(x,y)) for $f(x,y) \triangleq \begin{cases} xy/(x^2+y^2) & \text{if } (x,y) \neq (0,0) \\ 0 & \text{if } (x,y) = (0,0). \end{cases}$

$U \triangleq \{z\}, V \triangleq \{x\}.$

$$\frac{\begin{array}{lll} \checkmark\ \forall u \in U,\ \forall t \in \mathrm{Var}. & \text{P'} & \vdash u \text{ is differentiable in } t \\ \checkmark\ \forall t \in \mathrm{Var}, \forall v \in V. & \text{P} & \vdash t \text{ is differentiable in } v \end{array}}{\times\ \forall u \in U, \forall v \in V. \quad \text{P;P'} \vdash u \text{ is differentiable in } v}\ \text{Seq'}$$

$$\begin{pmatrix} \boxed{x} \\ y \\ z \end{pmatrix} \xrightarrow{[\![\text{P}]\!]} \begin{pmatrix} x \\ x \\ z \end{pmatrix} \xrightarrow{[\![\text{P'}]\!]} \begin{pmatrix} x \\ x \\ \boxed{1[x \neq 0] \cdot \frac{1}{2}} \end{pmatrix}$$

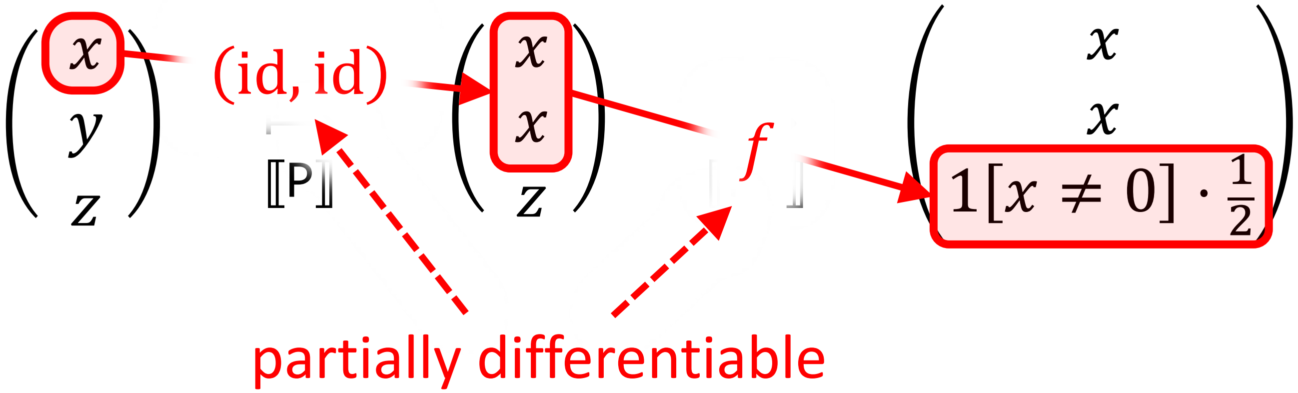
This rule is still unsound! But why?

# Soundness Issue (Again)

P;P' $\triangleq$ (y:=x ; z:=f(x,y)) for $f(x,y) \triangleq \begin{cases} xy/(x^2+y^2) & \text{if } (x,y) \neq (0,0) \\ 0 & \text{if } (x,y) = (0,0). \end{cases}$

$U \triangleq \{z\}, V \triangleq \{x\}$.

$$\dfrac{\begin{array}{ll} ✓\ \forall u \in U,\ \ \forall t \in \mathrm{Var}. & \texttt{P'} \vdash u \text{ is differentiable in } t \\ ✓\ \forall t \in \mathrm{Var}, \forall v \in V. & \texttt{P} \vdash t \text{ is differentiable in } v \end{array}}{✗\ \forall u \in U, \forall v \in V. \quad \texttt{P;P'} \vdash u \text{ is differentiable in } v}\ \text{Seq'}$$



# Soundness Issue (Again)

P;P' $\equiv$ … $\begin{cases} xy/(x^2+y^2) & \text{if } (x,y) \neq (0,0) \\ \ldots & \ldots = (0,0). \end{cases}$

$U \triangleq \{z$ …

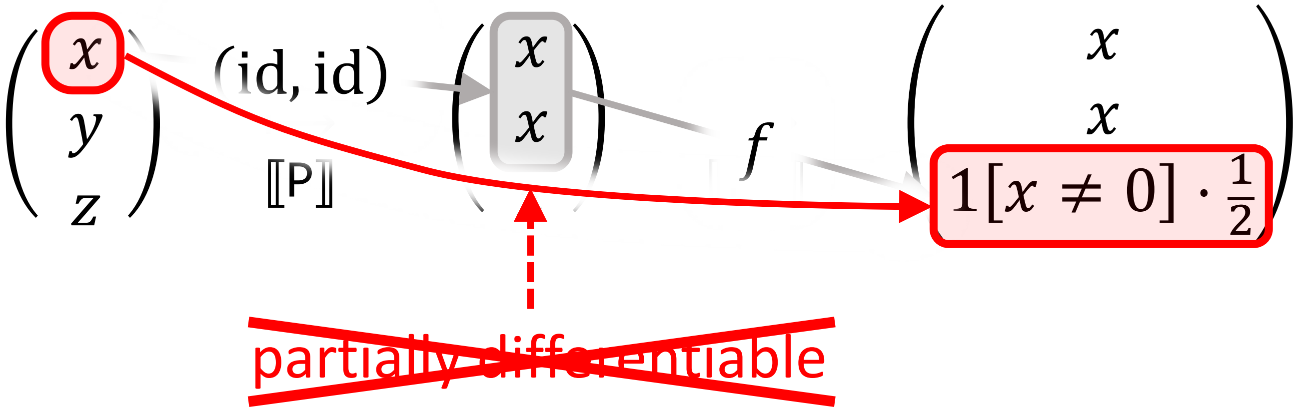
Fact: $g, h$ are partially differentiable $\not\Rightarrow$ $g \circ h$ does so.

✓ $\forall u \in U, \quad \forall t \in \mathrm{Var}.$ P' $\vdash$ $u$ is differentiable in $t$

✓ $\forall t \in \mathrm{Var}, \forall v \in V.$ P $\vdash$ $t$ is differentiable in $v$

——————————————————————— Seq'

✗ $\forall u \in U, \forall v \in V.$ P;P' $\vdash$ $u$ is differentiable in $v$



# Soundness Issue (Again)

P;P’ $\triangleq$ (y: … if $(x, y) \neq (0{,}0)$

$U \triangleq \{z\}, V \triangleq$ … $= (0{,}0)$.

$g, h$ are partially differentiable $\Longrightarrow$ $g \circ h$ does so.

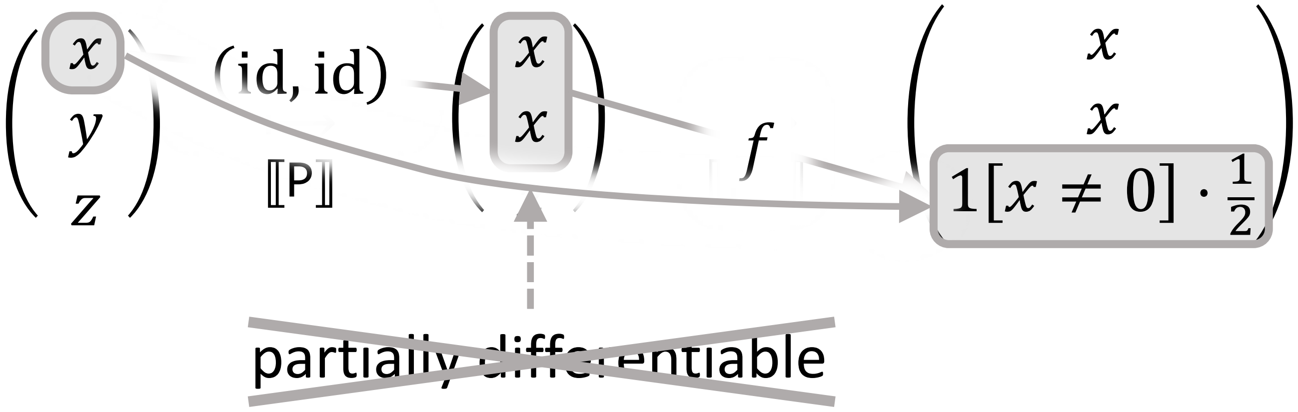
assumed implicitly, but invalid.

✓ $\forall u \in U, \quad \forall t \in \mathrm{Var}.$ P’ $\vdash$ $u$ is differentiable in $t$

✓ $\forall t \in \mathrm{Var}, \forall v \in V.$ P $\vdash$ $t$ is differentiable in $v$

───────────────────────── Seq’

✗ $\forall u \in U, \forall v \in V.$ P;P’ $\vdash$ $u$ is differentiable in $v$



# Soundness Issue (Again)

P;P' ≜ (y: ... $\begin{cases} xy/(x^2+y^2) & \text{if } (x,y) \neq (0,0) \\ & \ldots = (0,0). \end{cases}$

$U \triangleq \{z\}, V \triangleq$ ...

$g, h$ are partially differentiable $\Longrightarrow$ $g \circ h$ does so.

assumed implicitly, but invalid.

$$\dfrac{\begin{array}{lll} \forall u \in U,\ \forall t \in \mathrm{Var}. & \mathrm{P'} & \vdash u \text{ is differentiable in } t \\ \forall t \in \mathrm{Var}, \forall v \in V. & \mathrm{P} & \vdash t \text{ is differentiable in } v \end{array}}{\begin{array}{lll} \forall u \in U, \forall v \in V. & \mathrm{P;P'} & \vdash u \text{ is differentiable in } v \end{array}}\ \text{Seq'}$$

~~partially differentiable~~

**Lesson: Need to identify & check assumptions on target smoothness.**

It is subtle to do smoothness analysis, soundly (and precisely).

→ Our approach for smoothness analysis

# Our Approach: Smoothness Property

$\mathrm{P} \;\vdash\; U$ is $\phi$-smooth in $V \qquad (U, V \subseteq \mathrm{Var})$

# Our Approach: Smoothness Property

$$\mathrm{P} \;\vdash\; U \text{ is } \phi\text{-smooth in } V \qquad (U, V \subseteq \mathrm{Var})$$

- $\phi \subseteq \{f : \mathbb{R}^n \to \mathbb{R}^m\}$: any set of functions we consider "smooth".
  - E.g., $\{f$: partially differentiable$\}$.

# Our Approach: Smoothness Property

$$\mathsf{P}\ \vdash\ U \text{ is } \phi\text{-smooth in } V \qquad (U, V \subseteq \mathrm{Var})$$

- $\phi \subseteq \{f : \mathbb{R}^n \to \mathbb{R}^m\}$: any set of functions we consider "smooth".
  - E.g., $\{f$: partially differentiable$\}$, $\{f$: jointly differentiable$\}$, $\{f$: continuous$\}$, $\cdots$.

# Our Approach: Smoothness Property

$$\text{P} \;\vdash\; U \text{ is } \phi\text{-smooth in } V \qquad (U, V \subseteq \mathrm{Var})$$

- $\phi \subseteq \{f : \mathbb{R}^n \to \mathbb{R}^m\}$: any set of functions we consider "smooth".
  - E.g., $\{f$: partially differentiable$\}$, $\{f$: jointly differentiable$\}$, $\{f$: continuous$\}$, $\cdots$.

- P $\vdash$ $\{x, y\}$ is $\phi$-smooth in $\{y, z\}$

  $\iff$ $\cdots$

# Our Approach: Smoothness Property

$$\mathsf{P} \;\vdash\; U \text{ is } \phi\text{-smooth in } V \qquad (U, V \subseteq \mathrm{Var})$$

- $\phi \subseteq \{f : \mathbb{R}^n \to \mathbb{R}^m\}$: any set of functions we consider "smooth".
  - E.g., $\{f$: partially differentiable$\}$, $\{f$: jointly differentiable$\}$, $\{f$: continuous$\}$, $\cdots$.

- P $\vdash$ $\{x, y\}$ is $\phi$-smooth $\boxed{\text{in } \{y, z\}}$

$\Longleftrightarrow$ for any $x_0 \in \mathbb{R}$.

$$\begin{pmatrix} x_0 \\ \boxed{\begin{matrix} y \\ z \end{matrix}} \end{pmatrix} \overset{[\![\mathsf{P}]\!]}{\longmapsto} \begin{pmatrix} x' \\ y' \\ z' \end{pmatrix}$$

# Our Approach: Smoothness Property

$$\text{P} \;\vdash\; U \text{ is } \phi\text{-smooth in } V \qquad (U, V \subseteq \mathrm{Var})$$

- $\phi \subseteq \{f : \mathbb{R}^n \to \mathbb{R}^m\}$: any set of functions we consider "smooth".
  - E.g., $\{f$: partially differentiable$\}$, $\{f$: jointly differentiable$\}$, $\{f$: continuous$\}$, $\cdots$.

- P $\vdash$ $\boxed{\{x, y\} \text{ is}}$ $\phi$-smooth $\boxed{\text{in } \{y, z\}}$

$\iff$ $f : \mathbb{R}^2 \to \mathbb{R}^2$ for any $x_0 \in \mathbb{R}$.

$$\begin{pmatrix} x_0 \\ \boxed{\begin{matrix} y \\ z \end{matrix}} \end{pmatrix} \xmapsto{[\![\text{P}]\!]} \begin{pmatrix} \boxed{\begin{matrix} x' \\ y' \end{matrix}} \\ z' \end{pmatrix}$$

# Our Approach: Smoothness Property

$$\mathrm{P} \ \vdash \ U \text{ is } \phi\text{-smooth in } V \qquad (U, V \subseteq \mathrm{Var})$$

- $\phi \subseteq \{f : \mathbb{R}^n \to \mathbb{R}^m\}$: any set of functions we consider "smooth".
    - E.g., $\{f$: partially differentiable$\}$, $\{f$: jointly differentiable$\}$, $\{f$: continuous$\}$, $\cdots$.

- $\mathrm{P} \ \vdash \ \boxed{\{x, y\} \text{ is}}\ \phi\text{-smooth } \boxed{\text{in } \{y, z\}}$

$$\Longleftrightarrow \quad f : \mathbb{R}^2 \to \mathbb{R}^2 \in \phi \quad \text{for any } x_0 \in \mathbb{R}.$$

$$\begin{pmatrix} x_0 \\ \boxed{\begin{matrix} y \\ z \end{matrix}} \end{pmatrix} \overset{[\![\mathrm{P}]\!]}{\longmapsto} \begin{pmatrix} \boxed{\begin{matrix} x' \\ y' \end{matrix}} \\ z' \end{pmatrix}$$

# Our Approach: Smoothness Analysis

$$d_{\mathrm{P}},\ s_{\mathrm{P}}\ :\ 2^{\mathrm{Var}} \to 2^{\mathrm{Var}}$$

dependency info of P

smoothness info of P

# Our Approach: Smoothness Analysis

$$d_{\mathrm{P}},\ s_{\mathrm{P}}\ :\ 2^{\mathrm{Var}} \to 2^{\mathrm{Var}}$$

dependency info of P

smoothness info of P

- Invariants: P $\vdash$ $V$ is dependent at most on $d_{\mathrm{P}}(V)$.

  P $\vdash$ $V$ is $\phi$-smooth in $s_{\mathrm{P}}(V)$.

# Our Approach: Smoothness Analysis

$$d_{\mathrm{P}},\ s_{\mathrm{P}}\ :\ 2^{\mathrm{Var}} \to 2^{\mathrm{Var}}$$

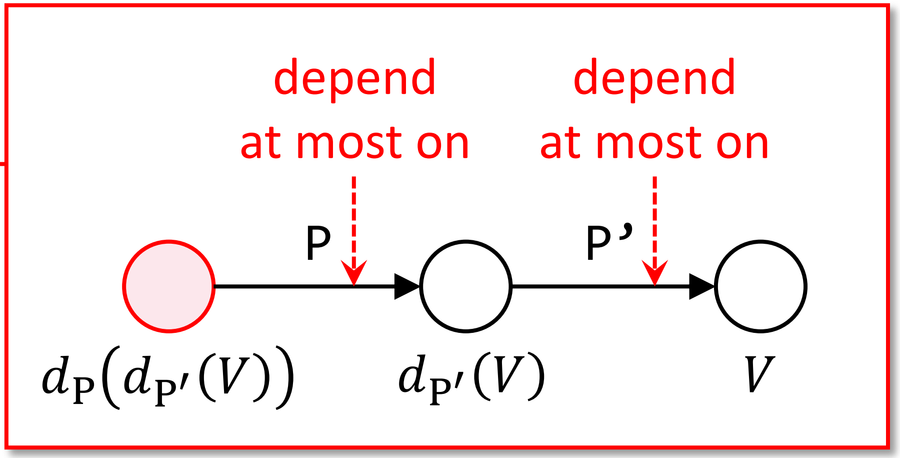
dependency info of P

smoothness info of P

- Invariants: P $\vdash$ $V$ is dependent at most on $d_{\mathrm{P}}(V)$.

  P $\vdash$ $V$ is $\phi$-smooth in $s_{\mathrm{P}}(V)$.

- Rules: $d_{\mathrm{P;P'}}(V) \triangleq d_{\mathrm{P}}(d_{\mathrm{P'}}(V))$.



# Our Approach: Smoothness Analysis

$$d_{\mathrm{P}},\ s_{\mathrm{P}}\ :\ 2^{\mathrm{Var}} \to 2^{\mathrm{Var}}$$

dependency info of P

smoothness info of P

- Invariants: P $\vdash$ $V$ is dependent at most on $d_{\mathrm{P}}(V)$.

  P $\vdash$ $V$ is $\phi$-smooth in $s_{\mathrm{P}}(V)$.

- Rules: $d_{\mathrm{P;P'}}(V) \triangleq d_{\mathrm{P}}\big(d_{\mathrm{P'}}(V)\big)$.

  $s_{\mathrm{P;P'}}(V) \triangleq s_{\mathrm{P}}\big(d_{\mathrm{P'}}(V)\big)\ \cap\ d_{\mathrm{P}}(s_{\mathrm{P'}}(V)^c)^c$.

# Our Approach: Smoothness Analysis

$$\underset{\substack{\uparrow \\ \text{dependency info of P}}}{d_{\mathrm{P}}},\ \underset{\substack{\uparrow \\ \text{smoothness info of P}}}{s_{\mathrm{P}}} \ :\ 2^{\mathrm{Var}} \to 2^{\mathrm{Var}}$$

- Invariants: P $\vdash$ $V$ is dependent at most on $d_{\mathrm{P}}(V)$.

  P $\vdash$ $V$ is $\phi$-smooth in $s_{\mathrm{P}}(V)$.

- Rules: $d_{\mathrm{P;P'}}(V) \triangleq d_{\mathrm{P}}\big(d_{\mathrm{P'}}(V)\big)$.

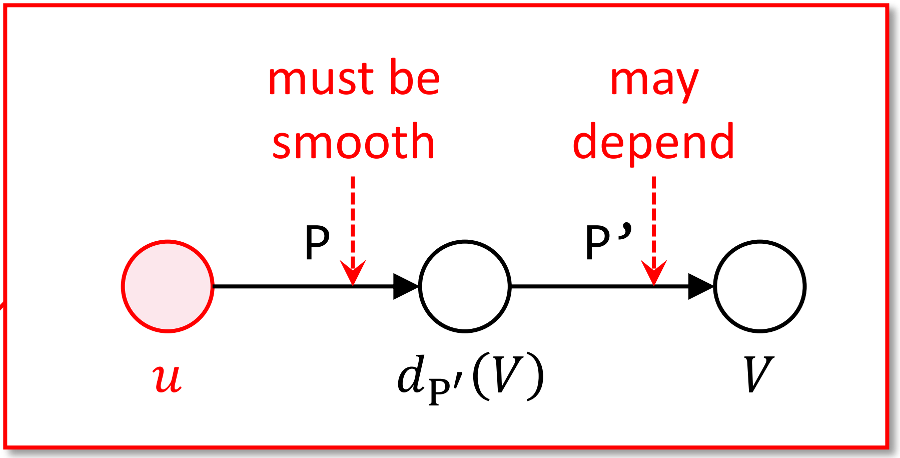
  $\underbrace{s_{\mathrm{P;P'}}(V)}_{u\,\in} \triangleq \underbrace{s_{\mathrm{P}}\big(d_{\mathrm{P'}}(V)\big)}_{u\,\in} \underset{\wedge}{\cap} \underbrace{d_{\mathrm{P}}(s_{\mathrm{P'}}(V)^c)^c}_{u\,\in}$.

# Our Approach: Smoothness Analysis

$$d_{\mathrm{P}},\ s_{\mathrm{P}} \ :\ 2^{\mathrm{Var}} \to 2^{\mathrm{Var}}$$

dependency info of P

smoothness info of P




- Invariants: P $\vdash$ $V$ is dependent at most on $d_{\mathrm{P}}(V)$.

  P $\vdash$ $V$ is $\phi$-smooth in $s_{\mathrm{P}}(V)$.

- Rules: $d_{\mathrm{P;P'}}(V) \triangleq d_{\mathrm{P}}\big(d_{\mathrm{P'}}(V)\big)$.

$$\underbrace{s_{\mathrm{P;P'}}(V)}_{u\,\in} \triangleq \underbrace{s_{\mathrm{P}}\big(d_{\mathrm{P'}}(V)\big)}_{u\,\in} \underset{\wedge}{\cap} \underbrace{d_{\mathrm{P}}(s_{\mathrm{P'}}(V)^c)^c}_{u\,\in}.$$

# Our Approach: Smoothness Analysis

$$d_{\mathrm{P}},\ s_{\mathrm{P}}\ :\ 2^{\mathrm{Var}} \to 2^{\mathrm{Var}}$$

dependency info of P

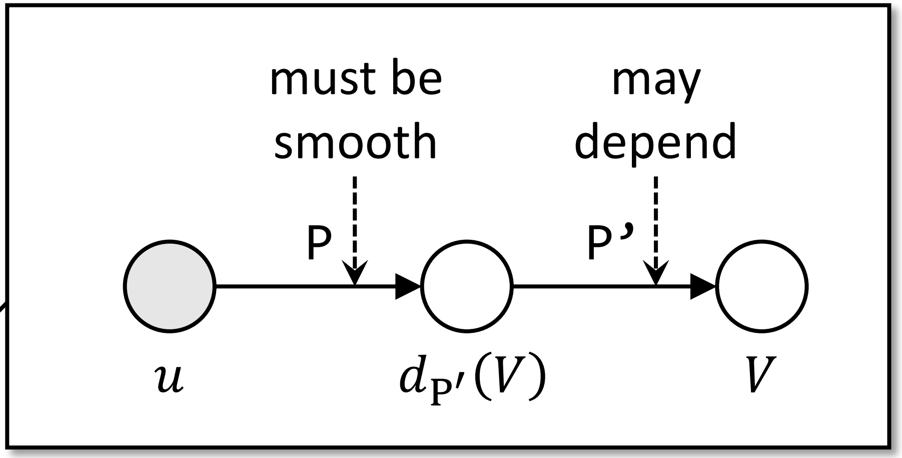
smoothness info of P

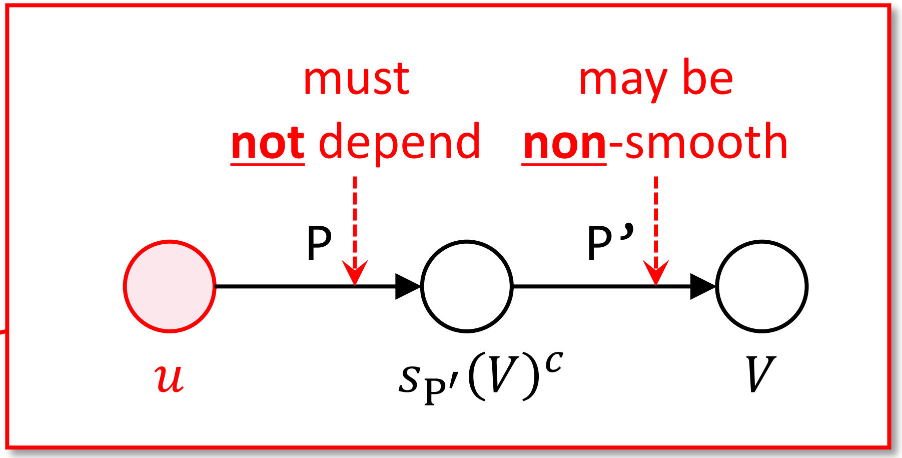
- Invariants: P $\vdash$ $V$ is dependent at most on $d_{\mathrm{P}}(V)$.

  P $\vdash$ $V$ is $\phi$-smooth in $s_{\mathrm{P}}(V)$.

- Rules: $d_{\mathrm{P;P'}}(V) \triangleq d_{\mathrm{P}}\big(d_{\mathrm{P'}}(V)\big)$.

$$\underbrace{s_{\mathrm{P;P'}}(V)}_{u\,\in} \triangleq \underbrace{s_{\mathrm{P}}\big(d_{\mathrm{P'}}(V)\big)}_{u\,\in} \underset{\wedge}{\cap} \underbrace{d_{\mathrm{P}}(s_{\mathrm{P'}}(V)^c)^c}_{u\,\in}.$$



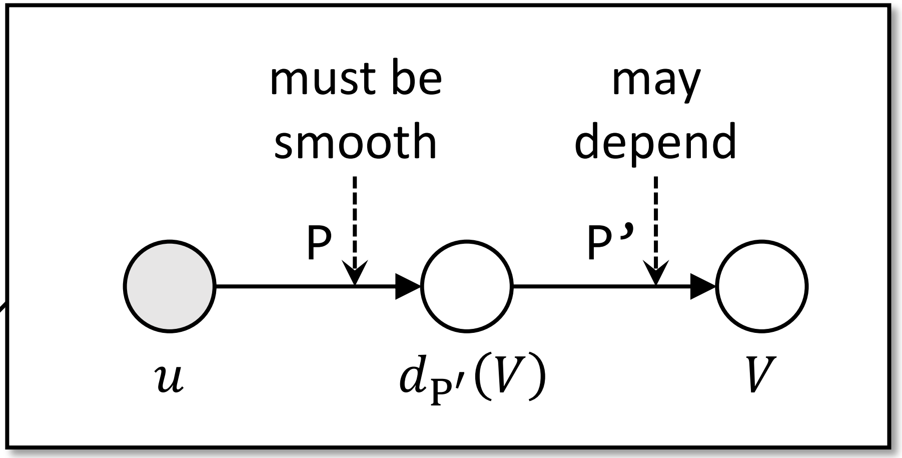



# Our Approach: Smoothness Analysis

$$d_{\mathrm{P}},\ s_{\mathrm{P}}\ :\ 2^{\mathrm{Var}} \to 2^{\mathrm{Var}}$$

dependency info of P

smoothness info of P



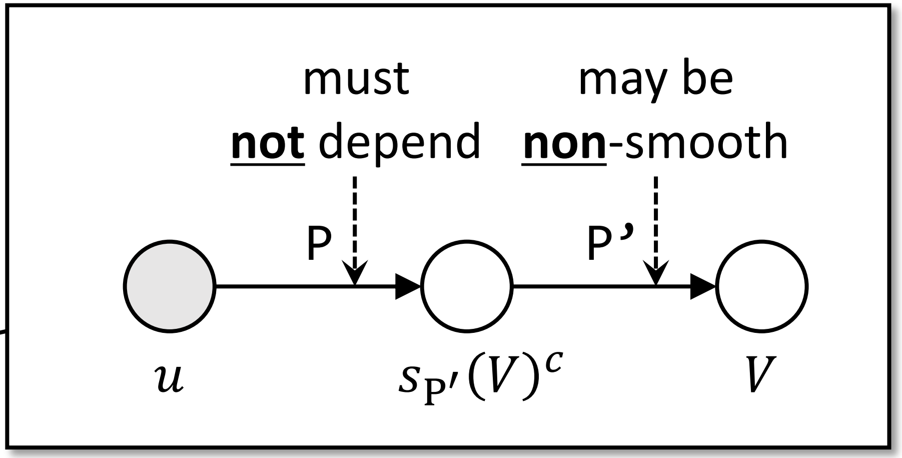

- Invariants: P $\vdash$ $V$ is dependent at most on $d_{\mathrm{P}}(V)$.

  P $\vdash$ $V$ is $\phi$-smooth in $s_{\mathrm{P}}(V)$.

- Rules: $d_{\mathrm{P;P'}}(V) \triangleq d_{\mathrm{P}}\big(d_{\mathrm{P'}}(V)\big)$.

$$\underbrace{s_{\mathrm{P;P'}}(V)}_{u\,\in} \triangleq \underbrace{s_{\mathrm{P}}\big(d_{\mathrm{P'}}(V)\big)}_{u\,\in} \cap \underbrace{d_{\mathrm{P}}\big(s_{\mathrm{P'}}(V)^c\big)^c}_{u\,\in}.$$

$\wedge$




$\Longrightarrow$ can apply the chain rule.

(not obvious; proved in our soundness theorem)

# Our Approach: Smoothness Analysis (Details)

$$d_{\mathrm{P}},\ s_{\mathrm{P}}\ :\ 2^{\mathrm{Var}} \to 2^{\mathrm{Var}}$$

dependency info of P

smoothness info of P

- Invariants: P $\vdash$ $V$ is dependent at most on $d_{\mathrm{P}}(V)$.

  P $\vdash$ $V$ is $\phi$-smooth in $s_{\mathrm{P}}(V)$.

- Rules: $d_{\mathrm{P;P'}}(V) \triangleq d_{\mathrm{P}}\big(d_{\mathrm{P'}}(V)\big)$.

  $s_{\mathrm{P;P'}}(V) \triangleq s_{\mathrm{P}}\big(d_{\mathrm{P'}}(V)\big)\ \cap\ d_{\mathrm{P}}(s_{\mathrm{P'}}(V)^c)^c$.

P : first-order, imperative PPL
(if-else, while, sample, observe, ...)

# Our Approach: Smoothness Analysis (Details)

$$d_{\mathrm{P}},\ s_{\mathrm{P}}\ :\ \cancel{2^{\mathrm{Var}} \to 2^{\mathrm{Var}}}$$

$\mathrm{Var} \to 2^{\mathrm{Var}}$

dependency info of P

smoothness info of P

$$d_{\mathrm{P}}(V) \coloneqq \bigcup_{v \in V} d_{\mathrm{P}}(v)$$

$$s_{\mathrm{P}}(V) \coloneqq \bigcap_{v \in V} s_{\mathrm{P}}(v)$$

- Invariants: P $\vdash$ $V$ is dependent at most on $d_{\mathrm{P}}(V)$.

  P $\vdash$ $V$ is $\phi$-smooth in $s_{\mathrm{P}}(V)$.

P : first-order, imperative PPL
(if-else, while, sample, observe, ...)

- Rules: $d_{\mathrm{P;P'}}(V) \triangleq d_{\mathrm{P}}(d_{\mathrm{P'}}(V))$.

  $s_{\mathrm{P;P'}}(V) \triangleq s_{\mathrm{P}}(d_{\mathrm{P'}}(V)) \cap d_{\mathrm{P}}(s_{\mathrm{P'}}(V)^c)^c$

More details are in our paper [POPL'23].

# Our Approration: Soundness

# Our Approach: Soundness

Theorem Our $\phi$-smoothness analysis is sound if $\phi$ satisfies five assumptions:

- Composition: $\forall f: \mathbb{R}^n \rightharpoonup \mathbb{R}^m, g: \mathbb{R}^m \rightharpoonup \mathbb{R}^l.\quad f, g \in \phi \implies (g \circ f) \in \phi.$ ⟵ chain rule
- Pairing: $\forall f: \mathbb{R}^n \rightharpoonup \mathbb{R}^m, g: \mathbb{R}^n \rightharpoonup \mathbb{R}^l.\quad f, g \in \phi \implies (f, g) \in \phi.$ ⟵ merging
- Restriction: $\forall f: \mathbb{R}^n \rightharpoonup \mathbb{R}^m, x \in \mathbb{R}^k (k \leq n).\ f \in \phi \implies f(x, -) \in \phi.$ ⟵ weakening
- Projection: $\forall f = \mathrm{proj}_{n \to m} : \mathbb{R}^n \rightharpoonup \mathbb{R}^m.\quad f \in \phi.$ ⟵ assignmt
- Strictness: $\forall f = \lambda x. \perp : \mathbb{R}^n \rightharpoonup \mathbb{R}^m.\quad f \in \phi.$ ⟵ while

# Our Approach: Soundness

| Target smoothness property | A3 (proj.) | A4 (pair.) | A5 (rest.) | A6 (comp.) | A7 (stri.) |
|---|---|---|---|---|---|
| cont. ($C^0$) | ○ | ○ | ○ | ○ | ○ |
| locally Lipschitz ($= \phi^{(l)}$) | ○ | ○ | ○ | ○ | ○ |
| uniformly cont. | ○ | ○ | ○ | ○ | ○ |
| Lipschitz cont. | ○ | ○ | ○ | ○ | ○ |
| jointly diff. | ○ | ○ | ○ | ○ | ○ |
| continuously diff. ($C^1$) | ○ | ○ | ○ | ○ | ○ |
| smooth ($C^\infty$) | ○ | ○ | ○ | ○ | ○ |
| real analytic ($C^\omega$) | ○ | ○ | ○ | ○ | ○ |
| partially cont. ($= \phi^{(pc)}$) | ○ | ○ | ○ | × | ○ |
| partially diff. ($= \phi^{(pd)}$) | ○ | ○ | ○ | × | ○ |
| almost-everywhere cont. | ○ | ○ | × | × | ○ |
| almost-everywhere diff. | ○ | ○ | × | × | ○ |
| coordinatewise non-decreasing | ○ | ○ | ○ | ○ | ○ |
| locally bounded | ○ | ○ | ○ | ○ | ○ |
| bounded | × | ○ | ○ | ○ | ○ |
| Borel measurable | ○ | ○ | ○ | ○ | ○ |
| locally integrable | ○ | ○ | × | × | ○ |
| integrable | × | ○ | × | × | ○ |

# Our Approach: Soundness

| Target smoothness property | A3 (proj.) | A4 (pair.) | A5 (rest.) | A6 (comp.) | A7 (stri.) |
|---|---|---|---|---|---|
| cont. ($C^0$) | O | O | O | O | O |
| locally Lipschitz ($= \phi^{(l)}$) | O | O | O | O | O |
| uniformly cont. | O | O | O | O | O |
| Lipschitz cont. | O | O | O | O | O |
| jointly diff. | O | O | O | O | O |
| continuously diff. ($C^1$) | O | O | O | O | O |
| smooth ($C^\infty$) | O | O | O | O | O |
| real analytic ($C^\omega$) | O | O | O | O | O |
| partially cont. ($= \phi^{(pc)}$) | O | O | O | X | O |
| partially diff. ($= \phi^{(pd)}$) | O | O | O | X | O |
| almost-everywhere cont. | O | O | X | X | O |
| almost-everywhere diff. | O | O | X | X | O |
| coordinatewise non-decreasing | O | O | O | O | O |
| locally bounded | O | O | O | O | O |
| bounded | X | O | O | O | O |
| Borel measurable | O | O | O | O | O |
| locally integrable | O | O | X | X | O |
| integrable | X | O | X | X | O |

# Our Approach: Soundness

| Target smoothness property | A3 (proj.) | A4 (pair.) | A5 (rest.) | A6 (comp.) | A7 (stri.) |
|---|---|---|---|---|---|
| cont. ($C^0$) | O | O | O | O | O |
| locally Lipschitz ($= \phi^{(l)}$) | O | O | O | O | O |
| uniformly cont. | O | O | O | O | O |
| Lipschitz cont. | O | O | O | O | O |
| jointly diff. | O | O | O | O | O |
| continuously diff. ($C^1$) | O | O | O | O | O |
| smooth ($C^\infty$) | O | O | O | O | O |
| real analytic ($C^\omega$) | O | O | O | O | O |
| partially cont. ($= \phi^{(pc)}$) | O | O | O | X | O |
| partially diff. ($= \phi^{(pd)}$) | O | O | O | X | O |
| almost-everywhere cont. | O | O | X | X | O |
| almost-everywhere diff. | O | O | X | X | O |
| coordinatewise non-decreasing | O | O | O | O | O |
| locally bounded | O | O | O | O | O |
| bounded | X | O | O | O | O |
| Borel measurable | O | O | O | O | O |
| locally integrable | O | O | X | X | O |
| integrable | X | O | X | X | O |

# Evaluation

- We implemented a static smoothness analyzer for Pyro programs.
  - It can analyze differentiability (and locally Lipschitz continuity).

# Evaluation

- We implemented a static smoothness analyzer for Pyro programs.
  - It can analyze differentiability (and locally Lipschitz continuity).

- We implemented our gradient estimator using our analyzer.

  (1) Identify differentiable parts of model $p(z, x)$ and guide $q_\theta(z)$.
  (2) Find $S \subseteq \{z_1, \ldots, z_n\}$ that satisfies diff'ty req's of path. grad. estimator.
  (3) Apply path. grad. estimator to $S$, and score estimator to $S^c$.

# Evaluation Results

random variables $z_i$
to which path. grad. estimator is applied

| Example in Pyro | LoC | Our estimator | Pyro's default estimator | |
|---|---|---|---|---|
| | | # rv (Sound) | # rv (Sound) | # rv (Unsound) |
| Splitting normal | 16 | | | |
| ⋯ (7 examples omitted) | ⋯ | | | |
| Deep exponential family | 105 | | | |
| Deep Markov model | 112 | | | |
| Hidden Markov model | 137 | | | |
| Single-cell annotation | 147 | | | |
| Attend-infer-repeat | 174 | | | |
| Conditional VAE | 205 | | | |

# Evaluation Results

random variables $z_i$
to which path. grad. estimator is applied

| Example in Pyro | LoC | Our estimator | Pyro's default estimator | |
|---|---|---|---|---|
| | | # rv (Sound) | # rv (Sound) | # rv (Unsound) |
| Splitting normal | 16 | | | |
| ⋯ (7 examples omitted) | ⋯ | | | |
| Deep exponential family | 105 | | | |
| Deep Markov model | 112 | | | |
| Hidden Markov model | 137 | | | |
| Single-cell annotation | 147 | | | |
| Attend-infer-repeat | 174 | | | |
| Conditional VAE | 205 | | | |

- Ours checks diff'ty req's using smoothness analysis.
- Pyro does not check diff'ty req's, so it can be unsound.

# Evaluation Results

random variables $z_i$
to which path. grad. estimator is applied

| Example in Pyro | LoC | Our estimator | Pyro's default estimator | |
|---|---|---|---|---|
| | | # rv (Sound) | # rv (Sound) | # rv (Unsound) |
| Splitting normal | 16 | 1 | 1 | 1 |
| ⋯ (7 examples omitted) | ⋯ | ⋯ | ⋯ | ⋯ |
| Deep exponential family | 105 | 6 | 6 | 0 |
| Deep Markov model | 112 | 1 | 1 | 0 |
| Hidden Markov model | 137 | 2 | 2 | 0 |
| Single-cell annotation | 147 | 3 | 3 | 0 |
| Attend-infer-repeat | 174 | 1 | 1 | 1 |
| Conditional VAE | 205 | 1 | 1 | 0 |

# Evaluation Results

random variables $z_i$
to which path. grad. estimator is applied

| Example in Pyro | LoC | Our estimator | Pyro's default estimator | |
|---|---|---|---|---|
| | | # rv (Sound) | # rv (Sound) | # rv (Unsound) |
| Splitting normal | 16 | 1 | 1 | 1 |
| ⋯ (7 examples omitted) | ⋯ | ⋯ | ⋯ | ⋯ |
| Deep exponential family | 105 | 6 | 6 | 0 |
| Deep Markov model | 112 | 1 | 1 | 0 |
| Hidden Markov model | 137 | 2 | 2 | 0 |
| Single-cell annotation | 147 | 3 | 3 | 0 |
| Attend-infer-repeat | 174 | 1 | 1 | 1 |
| Conditional VAE | 205 | 1 | 1 | 0 |

due to if-else

due to div-by-0

# High-Level Messages

- Smoothnes properties play an important role in prob. prog. (and other areas).
  - Example: Pathwise gradient estimator.

- It is subtle to design a proper smoothness analysis (sound and precise).
  - Reason: Make assumptions on target smoothness which are easily violated.

# High-Level Messages

- Smoothnes properties play an important role in prob. prog. (and other areas).
  - Example: Pathwise gradient estimator.
- It is subtle to design a proper smoothness analysis (sound and precise).
  - Reason: Make assumptions on target smoothness which are easily violated.
- There are some PL research opportunities for ML (which are less explored).
  - Example: Static analysis for automatic planning of inference algorithms.

# Gradient Estimators: General Case

Our **automatic** approach [POPL'23]. (≈ [Lew+23])

1. **Smoothness analysis:** Identify differentiable parts of a model/guide.
2. **Selective application:** Apply path. grad. est. only to these parts.

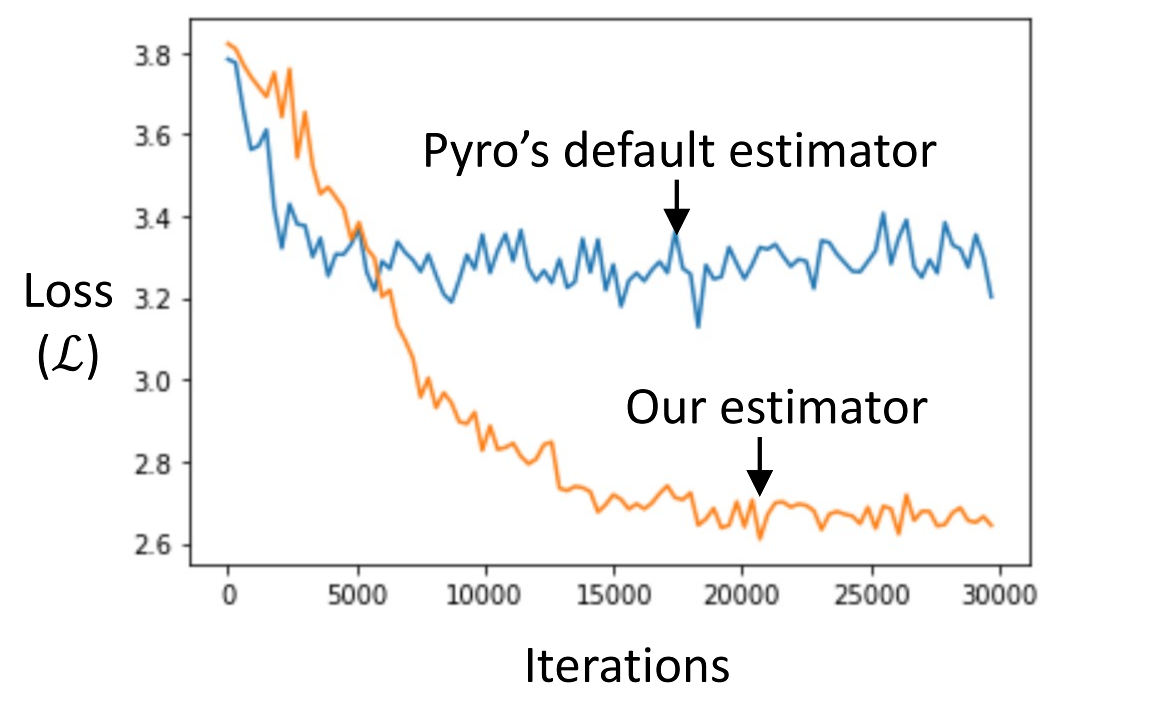
Other approaches.

- **Generalization** of path. grad. est.: [Lee+18], [Bangaru+21], ...
- **Smoothing** of input programs: [Khajwal+23], [Wagner+24], ...
- ...

# Evaluation Results

random variables $z_i$
to which path. grad. estimator is applied

| Example in Pyro | LoC | Our estimator | Pyro's default estimator | |
|---|---|---|---|---|
| | | # rv (Sound) | # rv (Sound) | # rv (Unsound) |
| Splitting normal | | | | 1 |
| ⋯ (7 examples o | | | | ⋯ |
| Deep exponentia | | | | 0 |
| Deep Markov mo | | | | 0 |
| Hidden Markov r | | | | 0 |
| Single-cell annot | | | | 0 |
| Attend-infer-repe | | | | 1 |
| Conditional VAE | | | | 0 |




due to branching

due to div-by-0