Differentiation stochastic programs and parametric discontinuities

UCSD CSE 291 Differentiable Programming
Tzu-Mao Li
Today: derivatives of stochastic programs

\[ \nabla_\theta E_{x' \sim p(x|\theta)}[f(x', \theta)] \]
Why do we want to differentiate stochastic programs?
Reinforcement learning

get “reward”
$E_{x' \sim p(x|\theta)}[f(x')]$

take action
$x' \sim p(x|\theta)$

$\theta$: parameters of the robot controller
Financial engineering

stock price= $E_{x' \sim p(x|\theta)}[f(x', \theta)]$

$\theta$: buy and sell stocks

Inverse rendering

image color = \( E_{x' \sim p(x|\theta)}[f(x', \theta)] \)

optimization video  target
Variational Bayesian inference

want to match two distributions using their KL divergence

more about this later

https://gregorygunderson.com/blog/2021/04/16/variational-inference/
Anything else?
Let’s focus on the probability density first

$$\nabla_{\theta} E_{x' \sim p(x|\theta)}[f(x')]$$

this is often called “policy gradient” in reinforcement learning
Goal: want to find an “estimator” $Y$ of the gradient of expectation

$$Y \approx \nabla_\theta E_{x' \sim p(x|\theta)}[f(x')]$$

what kind of properties should $Y$ hold?
Goal: want to find an “estimator” Y of the gradient of expectation

\[ Y \approx \nabla_{\theta} E_{x' \sim p(x|\theta)}[f(x')] \]

what kind of properties should Y hold?

1. unbiasedness: \( E[Y] = \nabla_{\theta} E_{x' \sim p(x|\theta)}[f(x')] \)
Goal: want to find an "estimator" $Y$ of the gradient of expectation

$$Y \approx \nabla_{\theta} E_{x' \sim p(x|\theta)}[f(x')]$$

what kind of properties should $Y$ hold?

1. unbiasedness: $E[Y] = \nabla_{\theta} E_{x' \sim p(x|\theta)}[f(x')]$
2. we want the variance $\text{Var}[Y]$ to be small
Expectation = integration

$$\nabla_{\theta} \int f(x') p(x' | \theta) dx'$$
Swap integral and gradient

$$\int f(x') \nabla_\theta p(x' | \theta) dx'$$

when can we do this?
Multiply by "1"

\[ \int f(x') \frac{\nabla_\theta p(x' | \theta)}{p(x' | \theta)} p(x' | \theta) dx' \]
Rewrite into an expectation

\[ E_{x' \sim p(x|\theta)} \left[ f(x') \frac{\nabla_{\theta} p(x'|\theta)}{p(x'|\theta)} \right] \]
Rewrite into an expectation

\[ E_{x' \sim p(x|\theta)} \left[ f(x') \frac{\nabla_{\theta} p(x'|\theta)}{p(x'|\theta)} \right] \]

this is called the “score function”, and is often written as

\[ \nabla_{\theta} \log p(x'|\theta) \]
We get our unbiased estimator $Y$!

$$Y_{score} = f(x') \frac{\nabla_\theta p(x' | \theta)}{p(x' | \theta)} \quad x' \sim p(x | \theta)$$

this estimator has been rediscovered many times and thus has many names: score (function) estimator, REINFORCE, likelihood ratio estimator, or zeroth-order estimator
A few notes about score estimators

- it does not require derivatives of $f$ — it is **model free**

- $f$ can be discontinuous

- $f$ can even be **unknown**, e.g., it can be a result of real-world interactions

- $Y_{\text{score}}$ can be seen as a **stochastic finite difference** scheme of $f$

- hence, when the dimensionality of $x'$ is high, $Y_{\text{score}}$ can have very high variance (in many cases it grows exponentially to the dimension)
Score estimator is related to “Evolutionary Strategies” a class of derivative-free optimization methods

\[
\frac{\partial}{\partial \theta} f(\theta) \approx \frac{1}{\sigma^2} \mathbb{E}_{\epsilon \sim \mathcal{N}(0, \sigma^2)} \epsilon f(\theta + \epsilon)
\]

https://www.inference.vc/evolutionary-strategies-embarrassingly-parallelizable-optimization/
An extension to the score estimator

\[ Y_{\text{score2}} = f(x') \frac{\nabla_{\theta} p(x' | \theta)}{q(x' | \theta)} \quad x' \sim q(x | \theta) \]

we don’t have to use \( p \) as the sampling distribution for \( x' \), this can improve variance
however,

1) choosing a good \( q \) is hard, and
2) you can’t use the same samples as your forward model
Can we design an estimator that actually uses derivatives of $f$?

$$\nabla_\theta E_{x' \sim p(x|\theta)}[f(x')]$$
Going back to the integral

$$\nabla_\theta \int f(x') p(x' | \theta) dx'$$
Going back to the integral

$$\nabla_\theta \int f(x') p(x' | \theta) dx'$$

apply a change of variable s.t. \(x' = T(z, \theta)\)

and \(\left| \frac{dz}{dx'} \right| = \frac{p(z)}{p(x'|\theta)}\)

= \nabla_\theta \int f(T(z, \theta))p(z)dz

example, for normal distribution \(N(\theta, I)\),

\(T = z + \theta\), and \(z \sim N(0,I)\)
Swap integral and gradient

when can we do this?

$$\int \nabla_\theta f(T(z, \theta)) p(z) dz$$
Apply chain rule

\[ \int \frac{\partial f}{\partial T} \frac{\partial T}{\partial \theta} p(z) \, dz \]
Back to expectation

$$E_{z \sim p(z)} \left[ \frac{\partial f}{\partial T} \frac{\partial T}{\partial \theta} p(z) \right]$$
We get another unbiased estimator!

\[ Y_{\text{pathwise}} = \frac{\partial f}{\partial T} \frac{\partial T}{\partial \theta} \quad x' = T(z, \theta) \]

\[ z \sim p(z) \]

this estimator also has many names: pathwise estimator, reparameterization trick, or first-order estimator.
Score vs pathwise estimators

- pathwise estimator can be seen as taking expectation of $f$’s derivatives
- usually, pathwise estimator has much lower variance when $f$’s dimensionality is high (autodiff vs finite differences)
- not always true!!
- pathwise estimator assumes
  - knowledge of $f$ (it is **model-based**)
  - continuity of $f$

\[
Y_{\text{score}} = f(x') \frac{\nabla_{\theta} p(x' | \theta)}{p(x' | \theta)}
\]

\[
Y_{\text{pathwise}} = \frac{\partial f}{\partial T} \frac{\partial T}{\partial \theta}
\]

if $p$ is normal distribution and $\theta$ is the mean, then the pathwise estimator is simply computing the gradient of $f$!
Score vs pathwise estimators

in high-dimensional cases, pathwise estimators seem to be almost always better than score estimators (I could be wrong)

ChainQueen: A Real-Time Differentiable Physical Simulator for Soft Robotics

Yuanming Hu, Jiancheng Liu*, Andrew Spielberg*, Joshua B. Tenenbaum, William T. Freeman, Jiajun Wu, Daniela Rus, Wojciech Matusik¹,²
in high-dimensional cases, pathwise estimators seem to be almost always better than score estimators

(I could be wrong)
Score vs pathwise estimators

in low-dimensional cases, when f is very noisy (e.g., chaos in physical systems),
taking it’s derivative can amplify the variance
(not super well understood)

Gradients are Not All You Need

Google Research, Brain Team
{lmetz, cdfreeman, schsam}@google.com

Tal Kachman
Radboud University
Donders Institute for Brain, Cognition and Behaviour
tal.kachman@donders.ru.nl
Score vs pathwise estimators

pathwise estimators fail when there are discontinuities
PL perspectives

**Input Loss as a Probabilistic Program**

\[
L = \lambda \theta : \mathbb{I}. \mathbb{E} \{ \begin{align*}
& b \leftarrow \text{flip} \theta \\
& \text{if } b \text{ then} \\
& \quad \text{return } 0 \\
& \text{else} \\
& \quad \text{return } -(\theta + 2)
\end{align*} \}
\]

**AD on deterministic parts only (incorrect)**

\[
L' = \lambda \theta : \mathbb{I}. \mathbb{E} \{ \begin{align*}
& b \leftarrow \text{flip} \theta \\
& \text{if } b \text{ then} \\
& \quad \text{return } 0 \\
& \text{else} \\
& \quad \text{return } -1/2
\end{align*} \}
\]

**ADEV (correct derivative)**

\[
L'' = \lambda \theta : \mathbb{I}. \mathbb{E} \{ \begin{align*}
& b \leftarrow \text{flip} \theta \\
& \text{if } b \text{ then} \\
& \quad \text{return } 0 \\
& \text{else} \\
& \quad \text{let } \delta \theta = 1/(\theta - 1) \\
& \quad \text{let } \delta l = -1/2 \\
& \quad \text{let } l = -\theta/2 \\
& \quad \text{return } \delta l + l \times \delta \theta
\end{align*} \}
\]

**Proof of correctness**

**Smoothness Analysis for Probabilistic Programs with Application to Optimised Variational Inference**

**ADEV: Sound Automatic Differentiation of Expected Values of Probabilistic Programs**

ALEXANDER K. LEW*, MIT, USA
MATHIEU HUOT*, Oxford University, UK
SAM STATON, Oxford University, UK
VIKASH K. MANSINGHKA, MIT, USA

**WONYEOL LEE**, Stanford University, USA
XAVIER RIVAL, INRIA Paris, France and ENS, CNRS, and PSL University, Paris, France
HONGSEOK YANG, KAIST, South Korea and Institute for Basic Science (IBS), South Korea

\[
c_m = \begin{cases} 
    (x_1 := \text{sam}(z_1^n, \text{dist}_N(0,5), \lambda y.y); \\
    & x_2 := \text{sam}(z_2^n, \text{dist}_N(x_1, 3), \lambda y.y); \\
    & \text{if } (x_2 > 0) \{ \text{obs}(\text{dist}_N(1,1), 0) \} \\
    & \text{else} \{ \text{obs}(\text{dist}_N(-2,1), 0) \}
\end{cases}
\]

Fig. 1. A model \( c_m \) and a guide \( c_g \) in a PPL. Here \( \text{dist}_N(a, b) \) is the distribution expression, and denotes the normal distribution with mean \( a \) and variance \( b \).

\[
c_g' = \begin{cases} 
    (x_1 := \text{sam}(z_1^n, \text{dist}_N(0,1), \lambda y.y + \theta_1); \\
    & x_2 := \text{sam}(z_2^n, \text{dist}_N(0,1), \lambda y.y + \theta_2); \\
\end{cases}
\]

\[
c_g'' = \begin{cases} 
    (x_1 := \text{sam}(z_1^n, \text{dist}_N(0,1), \lambda y.y + \theta_1); \\
    & x_2 := \text{sam}(z_2^n, \text{dist}_N(0,1), \lambda y.y); \\
\end{cases}
\]

Fig. 2. A fully (or selectively) reparameterised guide \( c_g' \) (or \( c_g'' \)).

prove of correctness

decompose programs into smooth / non-smooth parts,
only apply pathwise estimators to smooth parts
Let’s see some applications in action

PLASTICINE-LAB: A SOFT-BODY MANIPULATION BENCHMARK WITH DIFFERENTIABLE PHYSICS

Zhao Huang UC San Diego zhao@eng.ucsd.edu
Yuming Hu MIT yuming@mit.edu
Siyuan Zhou Peking University siyuanzhou@pku.edu.cn
Hao Su UC San Diego haosu@eng.ucsd.edu
Chuang Gan MIT-RM WANSON AI Lab ganchuang@csail.mit.edu
Tao Du MIT tsdu@csail.mit.edu
Joshua B. Tenenbaum MIT LCS, CSAIL, AI Lab jbt@mit.edu
Let’s see some applications in action

Material and lighting reconstruction for complex indoor scenes with texture-space differentiable rendering

Merlin Nimier-David¹,² Zhao Dong¹ Wenzel Jakob² Anton Kaplanyan¹

¹Facebook Reality Labs ²Realistic Graphics Lab, EPFL

EGSR 2021 - Supplemental video
Bayesian inference

given observations $\mathbf{X}$, we want to infer the probability distribution $p(\mathbf{Z} | \mathbf{X})$ of latent variables $\mathbf{Z}$

from Bayes, we know that $p(\mathbf{Z} | \mathbf{X}) \propto p(\mathbf{X} | \mathbf{Z})p(\mathbf{Z})$
Bayesian inference

given observations $\mathbf{X}$, we want to infer the probability distribution $p(\mathbf{Z} | \mathbf{X})$ of latent variables $\mathbf{Z}$

from Bayes, we know that $p(\mathbf{Z} | \mathbf{X}) \propto p(\mathbf{X} | \mathbf{Z})p(\mathbf{Z})$

e.g., given observation of an image $\mathbf{X}$, we want to infer the distribution of 3D scenes $\mathbf{Z}$ that will render to $\mathbf{X}$

we can set $p(\mathbf{X} | \mathbf{Z}) = \mathcal{N}($render$(\mathbf{Z}), \sigma^2 I)$, and $p(\mathbf{Z})$ to be some prior we believe the 3D scene has
(e.g., the geometry should be smooth, etc)

knowing $p(\mathbf{Z} | \mathbf{X})$ allows us to find all 3D scenes $\mathbf{Z}$ (not just the most likely one) that will render to an image $\mathbf{X}$, and their likelihood
Bayesian inference

given observations $\mathbf{X}$, we want to infer the probability distribution $p(\mathbf{Z} | \mathbf{X})$ of latent variables $\mathbf{Z}$

from Bayes, we know that $p(\mathbf{Z} | \mathbf{X}) \propto p(\mathbf{X} | \mathbf{Z})p(\mathbf{Z})$

e.g., black hole imaging (find the distribution of the parameters of the black hole PDE)

https://blackholecam.org/research/bhshadow/vlbi/
Bayesian inference

given observations $X$, we want to infer the probability distribution $p(Z|X)$ of latent variables $Z$

from Bayes, we know that $p(Z|X) \propto p(X|Z)p(Z)$

e.g., from your social media interactions ($X$), we want to find your political inclinations ($Z$)
Bayesian inference

given observations $\mathbf{X}$, we want to infer the probability distribution $p(\mathbf{Z} | \mathbf{X})$ of latent variables $\mathbf{Z}$

\[
p(\mathbf{Z} | \mathbf{X}) \propto p(\mathbf{X} | \mathbf{Z}) p(\mathbf{Z})
\]

challenge 1: we usually don’t know how to compute the normalization factor $p(\mathbf{X})$
(it’s uniquely determined once you specify $p(\mathbf{X} | \mathbf{Z})$ and $p(\mathbf{Z})$)
challenge 2: even if we know $p(\mathbf{X})$, we still don’t know how to sample from $p(\mathbf{Z} | \mathbf{X})$
Variational inference

given observations $\mathbf{X}$, we want to infer the probability distribution $p(\mathbf{Z} | \mathbf{X})$ of latent variables $\mathbf{Z}$

idea: define a parametric family of distribution $Q$ that is easy to sample
find $q(\mathbf{Z}, \theta^*) \in Q$ s.t. $q$ is the closest to $p$

https://gregorygundersen.com/blog/2021/04/16/variational-inference/
Variational inference

given observations $\mathbf{X}$, we want to infer the probability distribution $p(\mathbf{Z} | \mathbf{X})$ of latent variables $\mathbf{Z}$

idea: define a parametric family of distribution $Q$ that is easy to sample
find $q(\mathbf{Z}, \theta^*) \in Q$ s.t. $q$ is the closest to $p$

the measure of closeness is defined by the KL divergence
$D_{KL}[q(\mathbf{Z}, \theta) || p(\mathbf{Z} | \mathbf{X})]$

which turns out to be the same as minimizing ELBO (Evidence Lower BOund)

$E_{\mathbf{Z} \sim q(\mathbf{Z}, \theta)}[\log q(\mathbf{Z}, \theta) - \log p(\mathbf{Z} | \mathbf{X})]$
Let’s move to a different topic: parametric discontinuities

let’s first merge $f$ and $p$ into one

$$\nabla_\theta \int f(x', \theta)p(x' | \theta)dx' = \nabla_\theta \int g(x', \theta)dx'$$
Let’s move to a different topic: parametric discontinuities

\[ \nabla_\theta \int g(x', \theta)dx' \]

what if \( g \) is discontinuous?
Systematically Differentiating Parametric Discontinuities

SAI PRAVEEN BANGARU*, MIT CSAIL
JESSE MICHEL*, MIT CSAIL
KEVIN MU, MIT CSAIL
GILBERT BERNSTEIN, UC Berkeley and MIT CSAIL
TZU-MAO LI, MIT CSAIL
JONATHAN RAGAN-KELLEY, MIT CSAIL

Distributions for Compositionally Differentiating Parametric Discontinuities

JESSE MICHEL, Massachusetts Institute of Technology, USA
KEVIN MU, University of Washington, USA
XUANDA YANG, University of California, San Diego, USA
SAI PRAVEEN BANGARU, Massachusetts Institute of Technology, USA
ELIAS ROJAS COLLINS, Massachusetts Institute of Technology, USA
GILBERT BERNSTEIN, University of Washington, USA
JONATHAN RAGAN-KELLEY, Massachusetts Institute of Technology, USA
MICHAEL CARBIN, Massachusetts Institute of Technology, USA
TZU-MAO LI, University of California, San Diego, USA

Computations in physical simulation, computer graphics, and probabilistic inference often require the differentiation of discontinuous processes due to contact, occlusion, and changes at a point in time. Popular differentiable programming languages, such as PyTorch and JAX, ignore discontinuities during differentiation. This is incorrect for parametric discontinuities—conditions containing at least one real-valued parameter and at least one variable of integration. We introduce Potto, the first differentiable first-order programming language to soundly differentiate parametric discontinuities. We present a denotational semantics for programs and program derivatives and show the two accord. We describe the implementation of Potto, which enables separate compilation of programs. Our prototype implementation overcomes previous compile-time bottlenecks achieving an 88.1x and 441.2x speed up in compile time and a 2.3x and 7.9x speed up in runtime, respectively, on two increasingly large image stylization benchmarks. We showcase Potto by implementing a prototype differentiable renderer with separately compiled shaders.

CCS Concepts: • Theory of computation → Denotational semantics; Operational semantics; • Computing methodologies → Rendering; • Mathematics of computing → Functional analysis.

Additional Key Words and Phrases: Differentiable Programming, Denotational Semantics, Differentiable Rendering, Distribution Theory, Probabilistic Programming
A 1D example of parametric discontinuity

\[ \frac{d}{dt} \int_0^1 \{ x < t \} \, dx \]
A 1D example of parametric discontinuity

\[ \int_0^1 [x < t] dx = \begin{cases} 
0 & \text{if } t \leq 0 \\
t & \text{if } 0 < t < 1 \\
1 & \text{if } 1 \geq t 
\end{cases} \]

\[ \frac{d}{dt} \int_0^1 [x < t] dx = \begin{cases} 
0 & \text{if } t \leq 0 \\
1 & \text{if } 0 < t < 1 \\
0 & \text{if } 1 \geq t 
\end{cases} \]
Writing it as a program

```python
def g(t):
    sum = 0
    for i in range(N):
        if rand() < t:
            sum += 1
    return sum / N
```

goal: want to compute \( \frac{\partial g}{\partial t} \)
A toy program for demonstration

def g(t):
    sum = 0
    for i in range(N):
        if rand() < t:
            sum += 1
    return sum / N

def dg(t, dt):
    sum = 0
    dsum = 0
    for i in range(N):
        if rand() < t:
            sum += 1
            dsum += 0
    return dsum / N

goal: want to compute $\frac{\partial g}{\partial t}$

standard forward mode:
A toy program for demonstration

```python
def g(t):
    sum = 0
    for i in range(N):
        if rand() < t:
            sum += 1
    return sum / N

def dg(t, dt):
    sum = 0
    dsum = 0
    for i in range(N):
        if rand() < t:
            sum += 1
            dsum += 0
    return dsum / N
```

goal: want to compute $\frac{\partial g}{\partial t}$

standard forward mode:

classical autodiff doesn’t know we’re computing an integral!!

$$\frac{d}{dt} \int_0^1 [x < t] dx$$
Key: we need to differentiate before discretize

\[ \int_0^1 [x < t] \, dx \]

\[ \int_0^1 \delta(t - x) \, dx \]
The sifting property of Dirac delta

\[ \int_{0}^{1} \delta(t-x) \, dx = \int_{t}^{t-1} \delta(x')(\, dx') = \int_{t-1}^{t} \delta(x') \, dx' = [0 < t < 1] \]
The sifting property of Dirac delta

essentially, we want to find \( x \) s.t. \( t - x = 0 \), and evaluate the integral only at the root

\[
\int_0^1 \delta(t - x) \, dx = \int_t^{t-1} \delta(x') \, (-dx') = \int_{t-1}^t \delta(x') \, dx' = [0 < t < 1]
\]
Differentiate -> discretize ->

feed in $x = t$ -> correct result

\[
\int_0^1 [x < t] dx
\]

\[
\int_0^1 \delta(t - x) dx
\]

$[0 < t < 1]$
General case

\[ \nabla_\theta \int_D g(x, \theta)[c(x, \theta) < 0] \, dx \]
General case

\[ \nabla_\theta \int_D g(x, \theta) [c(x, \theta) < 0] \, dx = \int_D (\nabla_\theta g(x, \theta)) [c(x, \theta) < 0] \, dx \]

derivatives of the smooth part
General case

\[ \nabla_\theta \int_D g(x, \theta) \left[ c(x, \theta) < 0 \right] dx \]

= \[ \int_D \left( \nabla_\theta g(x, \theta) \right) \left[ c(x, \theta) < 0 \right] dx \]

+ \[ \int_D g(x, \theta) \delta(c(x, \theta)) dx \]

derivatives of the smooth part

derivatives at the discontinuities
General case

\[ \nabla_\theta \int_D g(x, \theta)[c(x, \theta) < 0] \, dx = \int_D (\nabla_\theta g(x, \theta))[c(x, \theta) < 0] \, dx + \int_{c(x', \theta) = 0} \frac{g(x', \theta)}{\lVert \nabla c(x', \theta) \rVert} \, dx' \]

derivatives of the smooth part

derivatives at the discontinuities
Key message: AD systems should know that we are integrating something

@integrate1D
def g(t):
    sum = 0
    for i in range(N):
        if rand() < t:
            sum += 1
    return sum / N
We are building differentiable programming languages with integrals currently, only supports simplistic constructs:

\[
\begin{align*}
C & \quad \bar{X} \\
\begin{align*}
e_1 & \quad e_2 \\
\int_{x=a}^{b} e & \quad [\phi(\bar{x}) > 0] \\
f(e) & 
\end{align*}
\end{align*}
\]
Applications: inverse shader design

- thresholding Perlin noise leads to discontinuities

Threshold: \[ \text{Threshold} \quad \ldots \ast [\text{noise} > t] \ast \ldots \]
Applications: inverse shader design

target image

our shader optimization

Bangaru*, Michel*, Mu, Bernstein, Li, Ragan-Kelley, 2021
Applications: inverse shader design

- ignoring discontinuities lead to worse/incorrect results

target image  our shader optimization  naive autodiff

Bangaru*, Michel*, Mu, Bernstein, Li, Ragan-Kelley, 2021
Applications: inverse shader design

- ignoring discontinuities lead to worse/incorrect results

target image  our shader optimization  naive autodiff

*Bangaru*, Michel*, Mu, Bernstein, Li, Ragan-Kelley, 2021
Applications: animation design/motion planning

- Optimized trajectory
- In the presence of
- Time discontinuities (Windmill)
- Space discontinuities (Contact)
- Friction
Applications: animation design/motion planning
Applications: optimizing a discontinuous bungee

\[ m\ddot{x} = mg - s(x) \]

\[ s(x) = \begin{cases} 
  \frac{k_1 x_1 + k_2 x_2}{2} & \text{if } x_1 \leq l_1, x_2 \leq l_2 \\
  \frac{\alpha k_1 l_1 + k_2 (x - l_1)}{2} & \text{if } x_1 > l_1, x_2 < l_2 \\
  \frac{\alpha k_2 l_2 + k_1 (x - l_2)}{2} & \text{if } x_1 < l_1, x_2 > l_2 \\
  g & \text{if } x_1 \geq l_1, x_2 \geq l_2
\end{cases} \]

minimize time to fall:

\[ \int_{x=x_0}^{x_1} 2 \left( \int_{x=0}^{\dot{x}} g - \frac{s(x)}{m} \right)^{\frac{1}{2}} \]
Applications: optimizing a discontinuous bungee
Applications: inverse rendering

photos

3D reconstruction

Sun, Cai, Li, Yan, Zhang, Marshall, Huang, Zhao, Dong, 2023