Estimating Gradients of Programs with Discrete Randomness:

With Applications in Detector Design Optimization

Michael Kagan, SLAC
Lukas Heinrich, TUM

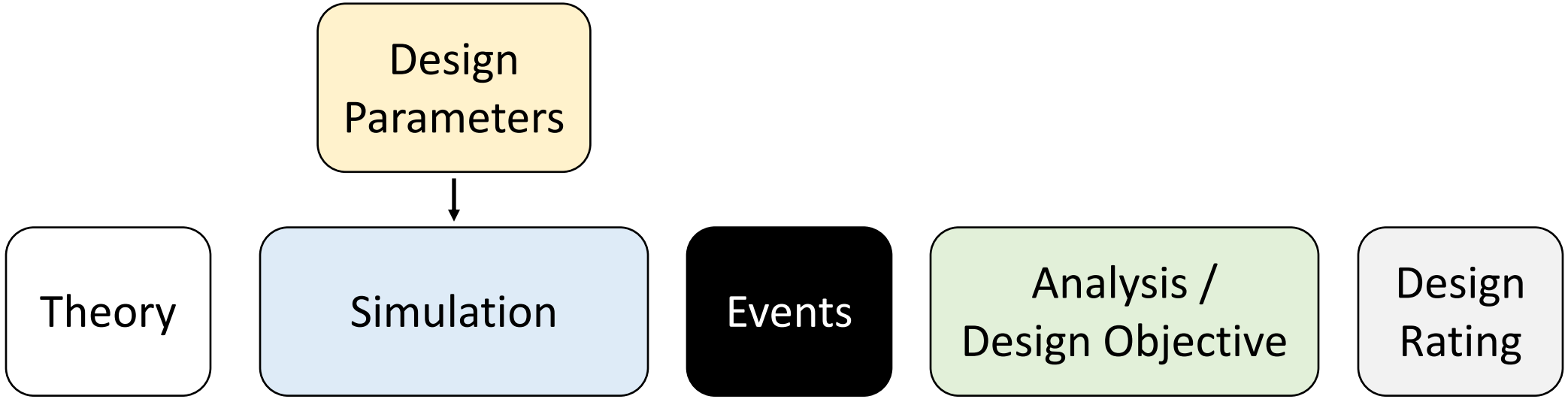
3rd MODE Workshop
July 26, 2023

SLAC NATIONAL ACCELERATOR LABORATORY

Stanford University | U.S. DEPARTMENT OF ENERGY

Detector Design Optimization

$$\min_{\phi} \mathbb{E}[f(x,\phi)] = \min_{\phi} \int f(x,\phi) p_{\phi}(x) dx$$



Detector Design Optimization

$$\min_{\phi} \mathbb{E}[f(x,\phi)] = \min_{\phi} \int f(x,\phi) p_{\phi}(x) dx$$

Design Objective

Realizations of measurements:

E.g. Simulations

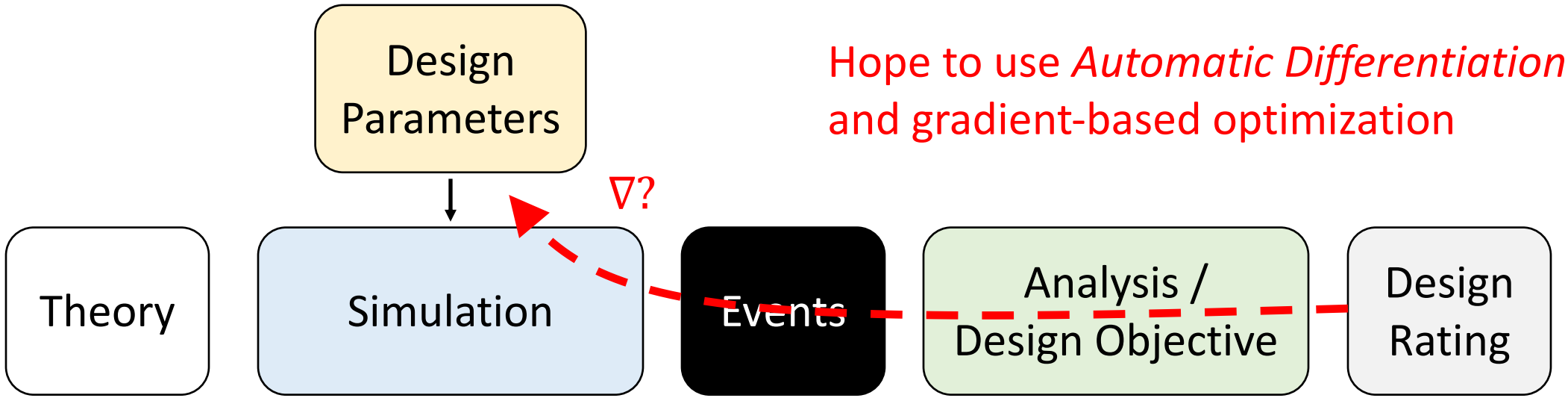
Design Parameters

Probability to see a measurement:

E.g. Interaction and detection probability

Detector Design Optimization

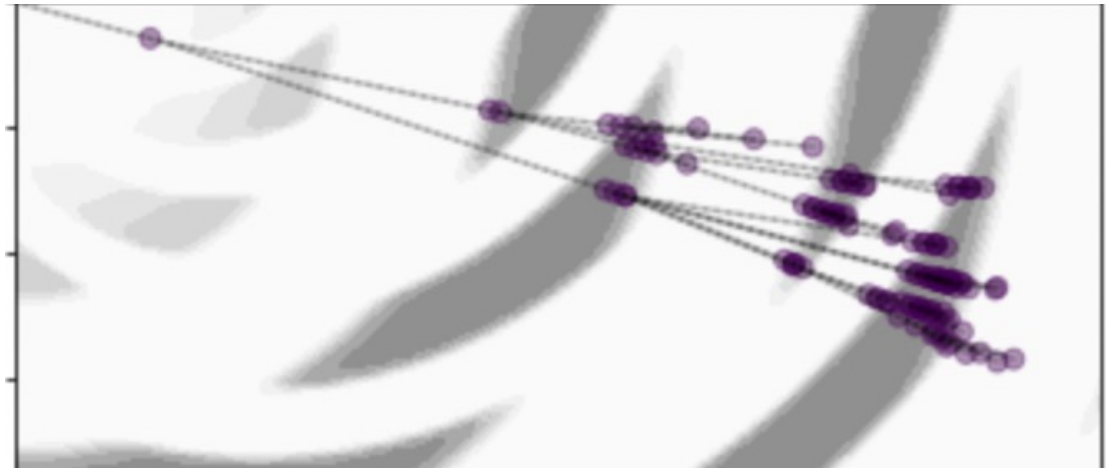
$$\min_{\phi} \mathbb{E}[f(x,\phi)] = \min_{\phi} \int f(x,\phi) p_{\phi}(x) dx$$



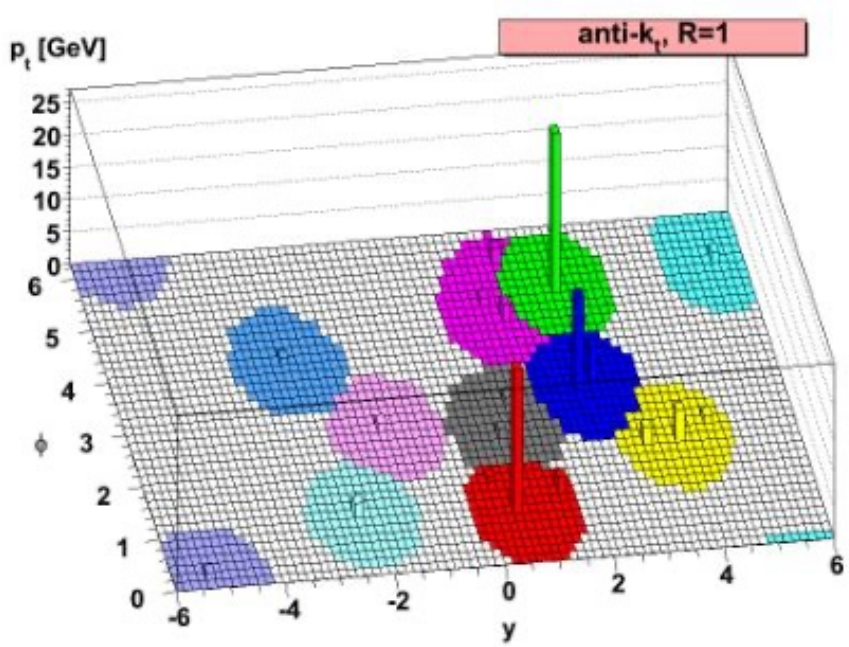
A Problem: Discrete Random Variables & Choices

Discrete random variables and discrete choices are all over HEP

Branching / Showering Processes



Clustering Algorithms

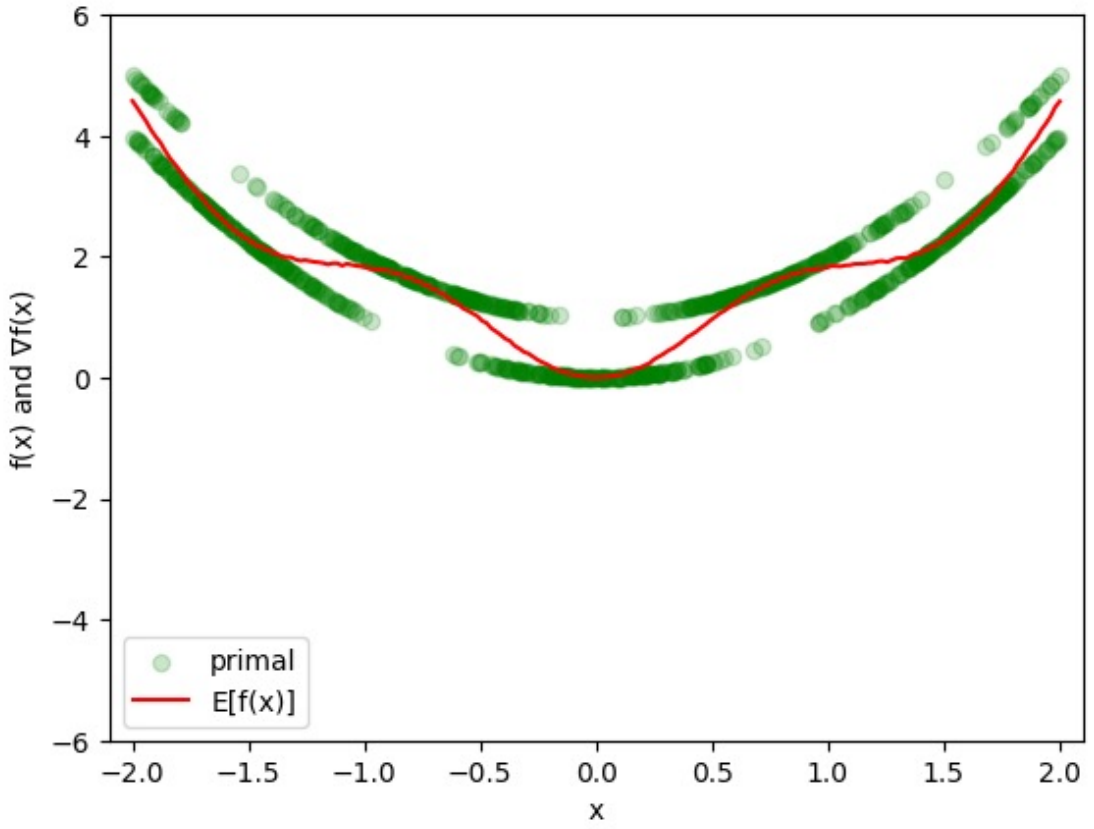


Programs with Discrete Randomness

$$f(x) = x^2 + b \qquad b \sim Bernoulli(\theta = \sin^2(2x))$$

```
def f(x):
    theta = (sin(2.*x))**2
    b = bernoulli(theta)
    g = x*x
    return g+b
```

Bernoulli parameter θ depends on x



Gradients of Expected Values

Even if a program contains discrete randomness, expected value of the program may be smooth and have a well-defined derivative

Simple example:

```
def f(theta):
    b = bernoulli(theta)
    return b
```

$$\mathbb{E}[f(\theta)] = \mathbb{E}_{b \sim Bern(\theta)}[b] = \theta$$

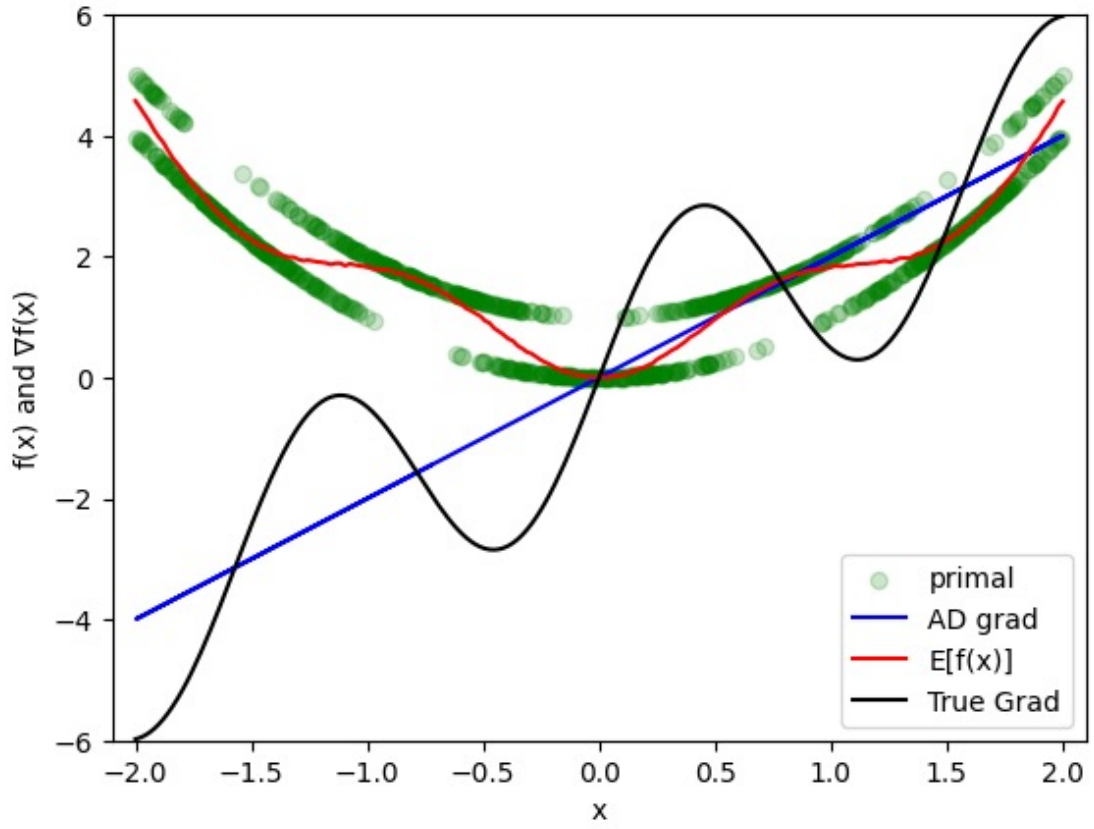
$$\nabla_\theta \mathbb{E}[f(\theta)] = 1$$

AD in Programs with Discrete Randomness


```
def f(x):
    theta = (sin(2.*x))**2
    b = bernoulli(theta)
    g = x*x
    return g+b
```

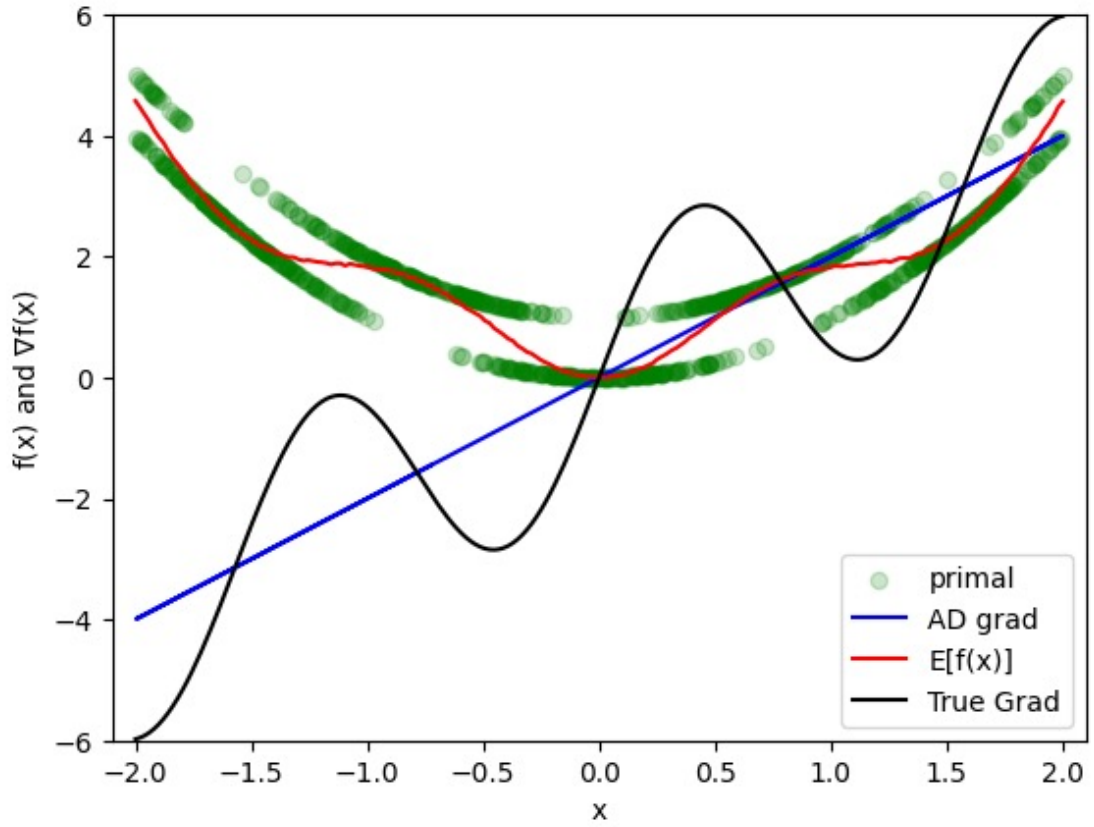
AD Gradient: $grad(f_i(x))$

True gradient: $\nabla_x E[f(x)]$



AD in Programs with Discrete Randomness


```
def f(x):
    theta = (sin(2.*x))**2
    b = bernoulli(theta)
    g = x*x
    return g+b
```



Standard AD tools don't know how to handle discrete randomness that depends on the parameter of differentiation

We need another approach

Derivatives for Discrete Randomness

Do Some Work, Get Better Derivatives

Approximate Derivatives

Don't Use Derivatives

Score Functions

Stochastic AD

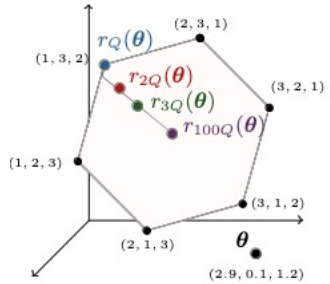
Smoothing / Relaxations

Surrogates

Gradient-Free Methods

Example:

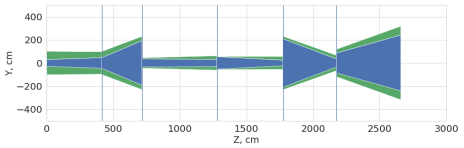
Differentiable ranking and sorting



2002.08871

HEP Example:

Surrogates for SHiP magnet design



MK, et al. 2002.04632

Bayesian Optimization, Genetic Algorithms, …

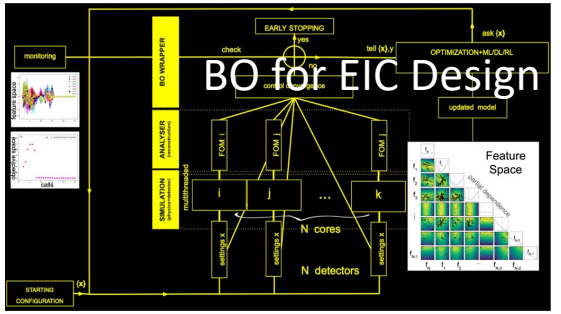


Figure credit: C. Fanelli

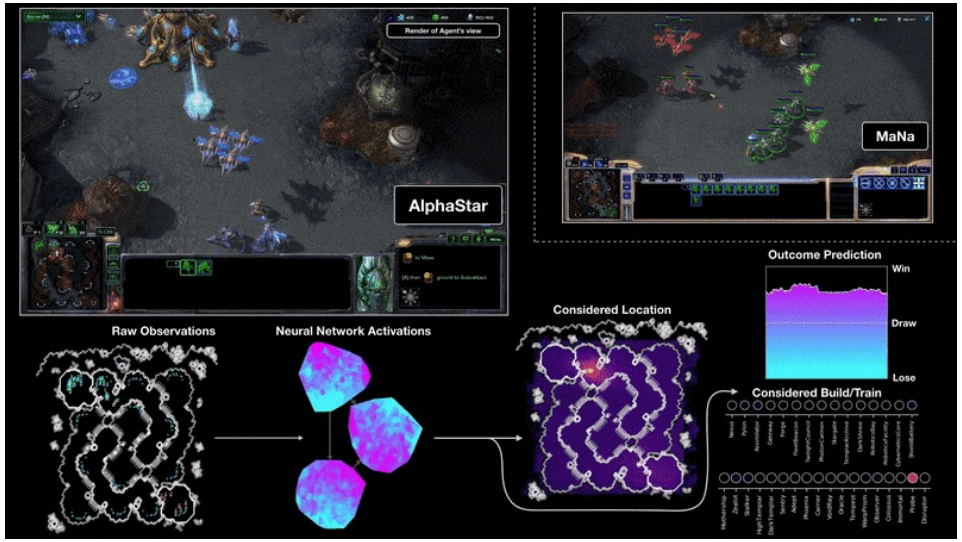
Score Functions / REINFORCE

$$\nabla_\theta \mathbb{E}_{p_\theta(x)}[f(x)] = \mathbb{E}_{p_\theta(x)}[f(x)\nabla_\theta \log p_\theta(x)]$$

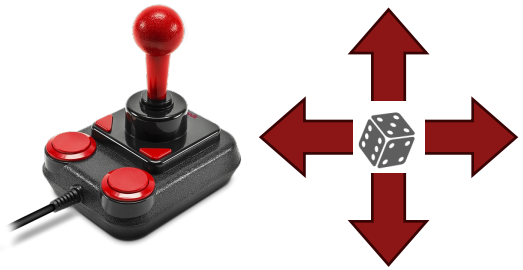
Gradient estimator used in Reinforcement Learning

Works with discrete x and even non-differentiable $f(\cdot)$

Requires tracking probabilities $\log p_\theta(x)$ throughout program



AlphaStar Vinyals et al. 2019



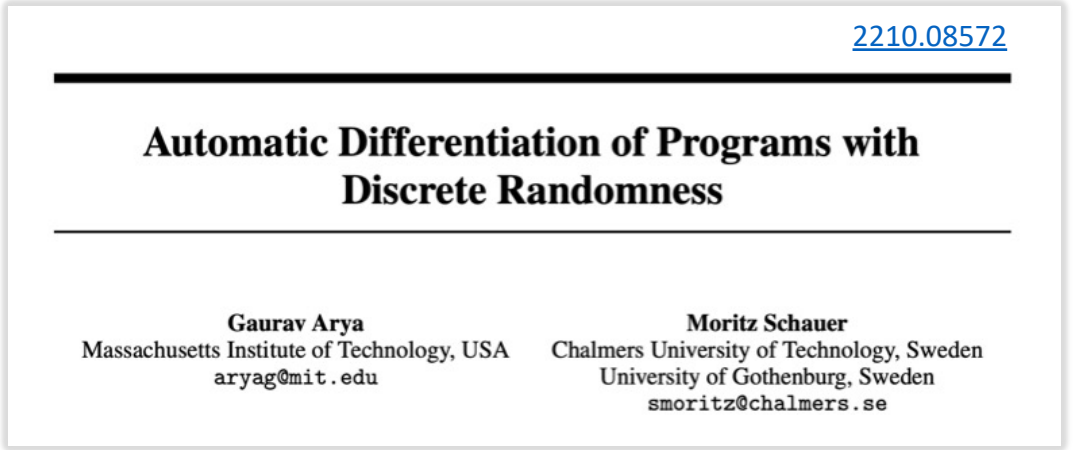
Stochastic AD

$$\frac{d}{d\theta}\mathbb{E}_{p_\theta}[f(x)] = \mathbb{E}_{p_\theta}[\delta + \beta(y - x)]$$

Standard AD Weight Alternative value of rv

Recently, *Arya et al.* extended fwd-mode AD to discrete-stochastic environments

Importantly, this includes a ***composition rule*** for how to combine weights β step-by-step along the computation chain



2210.08572

Automatic Differentiation of Programs with Discrete Randomness

Gaurav Arya
Massachusetts Institute of Technology, USA
aryag@mit.edu

Moritz Schauer
Chalmers University of Technology, Sweden
University of Gothenburg, Sweden
smoritz@chalmers.se

Quick Aside: Reparameterization Trick

$$\frac{d}{d\theta}\mathbb{E}_{p_\theta(x)}[f(x)] = \frac{d}{d\theta}\mathbb{E}_{p(\epsilon)}[f(g(\epsilon,\theta))] = \mathbb{E}_{p(\epsilon)}\left[\frac{df}{dg}\frac{dg}{d\theta}\right]$$

Common method for continuous rv's is the *reparameterization trick*

If $x \sim p_\theta(x) \rightarrow$ rewrite $x = g(\epsilon, \theta)$ with $\epsilon \sim p(\epsilon)$

Separate parameters from stochasticity

Example:

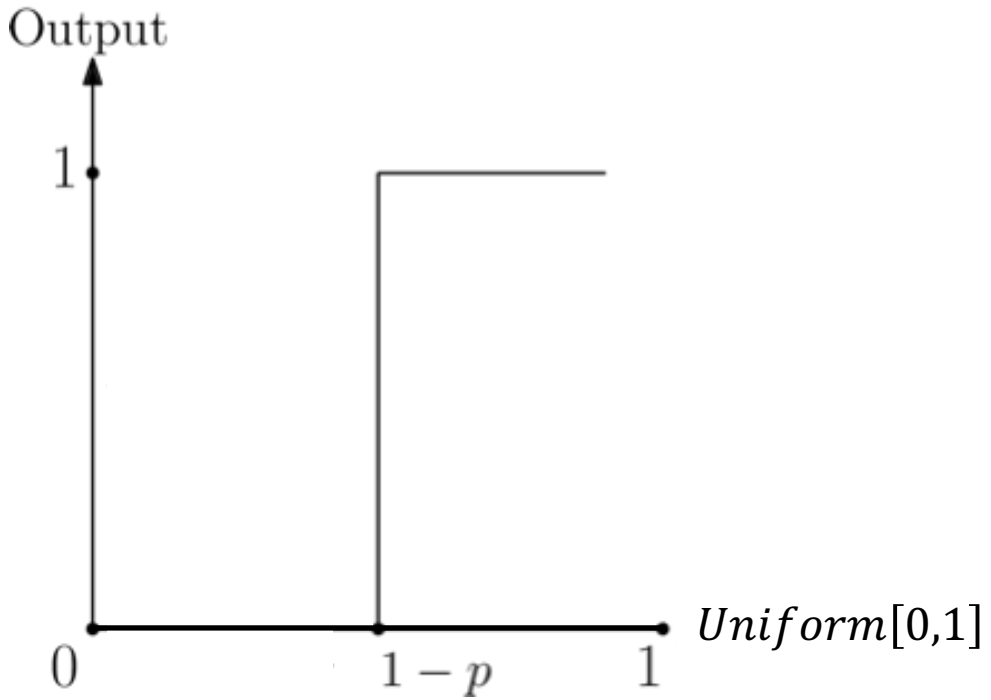
$$x \sim \mathcal{N}(\mu, \sigma) \quad \rightarrow \quad x = \epsilon * \sigma + \mu \quad \text{with } \epsilon \sim \mathcal{N}(0,1)$$

Intuition

Can use the inversion method to reparameterize discrete random variables

$$x \sim Bernoulli(p) \quad \rightarrow \quad \omega \sim Uniform[0,1]$$

$$x = \begin{cases} 1 \ \ if \ \omega > 1 - p \\ 0 \ \ Otherwise. \end{cases}$$



Arya et al. 2210.08572

Intuition

Can use the inversion method to reparameterize discrete random variables

$$x \sim Bernoulli(p) \quad \rightarrow \quad \omega \sim Uniform[0,1]$$

$$x = \begin{cases} 1 \ \ if \ \omega > 1 - p \\ 0 \ \ Otherwise. \end{cases}$$

$$\mathbb{E}[b] = \int 1_{[\omega > 1-p]} p(\omega) d\omega = \int_{1-p}^{1} d\omega$$

Intuition

Can use the inversion method to reparameterize discrete random variables

$$x \sim Bernoulli(p) \quad \rightarrow \quad \omega \sim Uniform[0,1]$$

$$x = \begin{cases} 1 \ \ if \ \omega > 1 - p \\ 0 \ \ Otherwise. \end{cases}$$

$$\mathbb{E}[b] = \int 1_{[\omega > 1-p]} p(\omega) d\omega = \int_{1-p}^{1} d\omega$$

Standard AD on Monte Carlo expectation of this program would still be wrong

$$grad_p \left(\frac{1}{N} \sum_i [1 \ if \ (\omega_i > 1 - p) \ else \ 0] \right) = 0$$

Intuition

Can use the inversion method to reparameterize discrete random variables

$$x \sim Bernoulli(p) \quad \rightarrow \quad \omega \sim Uniform[0,1]$$

$$x = \begin{cases} 1 & if\ \omega > 1 - p \\ 0 & Otherwise. \end{cases}$$

$$\mathbb{E}[b] = \int 1_{[\omega > 1-p]} p(\omega) d\omega = \int_{1-p}^{1} d\omega$$

Param. dependence in integration bounds

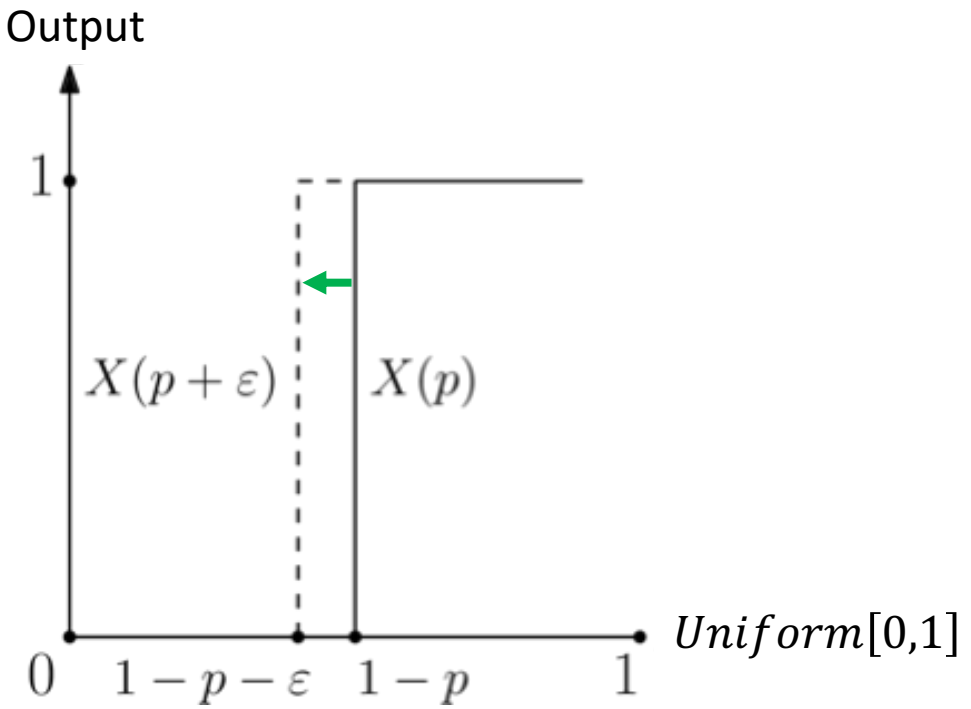
Correct derivative must account for boundary dependence → *Leibniz Rule*

Intuition

Can use the inversion method to reparameterize discrete random variables

$$x \sim Bernoulli(p) \quad \rightarrow \quad \omega \sim Uniform[0,1]$$

$$x = \begin{cases} 1 \ \ if\ \omega > 1-p \\ 0 \ \ Otherwise. \end{cases}$$



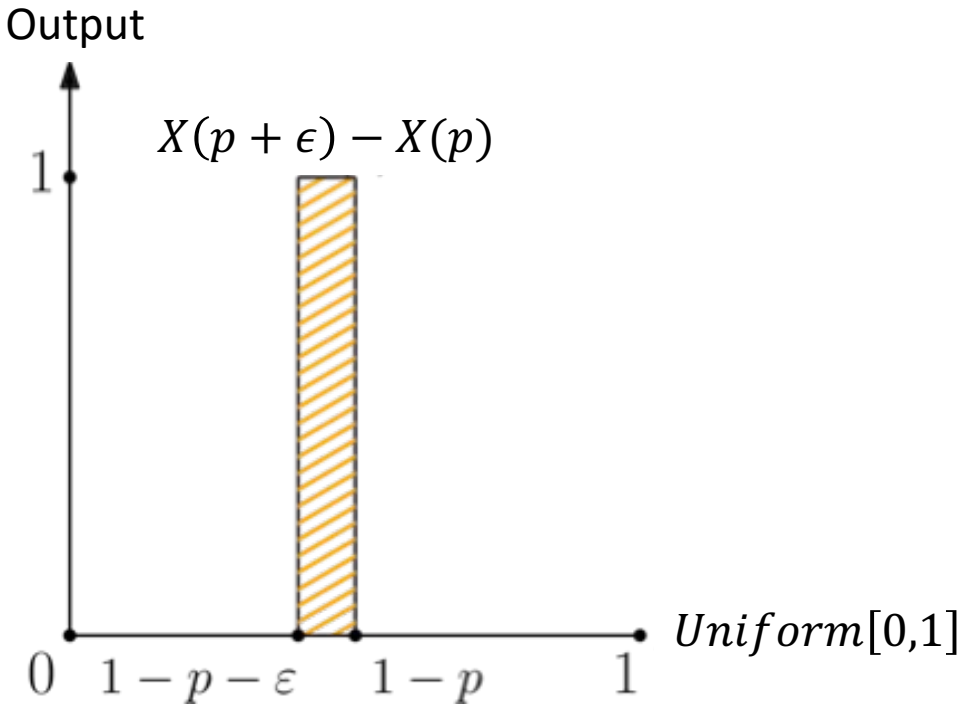
Arya et al. 2210.08572

Intuition

Can use the inversion method to reparameterize discrete random variables

$$x \sim Bernoulli(p) \quad \rightarrow \quad \omega \sim Uniform[0,1]$$

$$x = \begin{cases} 1 \ \ if \ \omega > 1 - p \\ 0 \ \ Otherwise. \end{cases}$$



Arya et al. 2210.08572

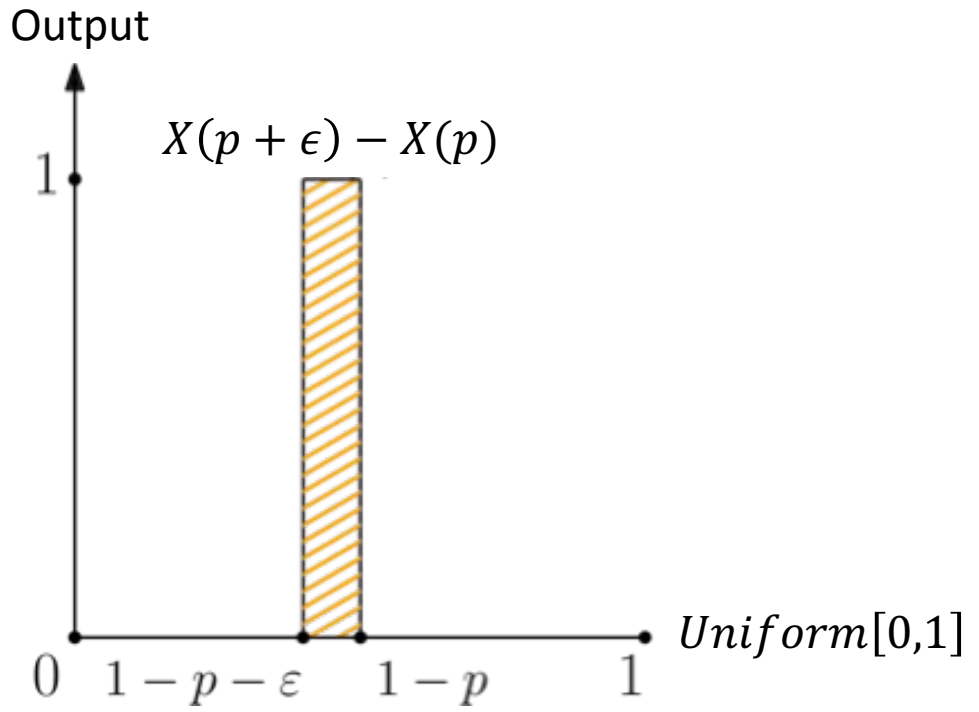
Stochastic AD

$$\frac{d}{d\theta}\mathbb{E}_{p_\theta}[f(x)] = \mathbb{E}_{p_\theta}[\delta + \beta(y - x)]$$

The weight β accounts for the derivative of the probability of a jump in program

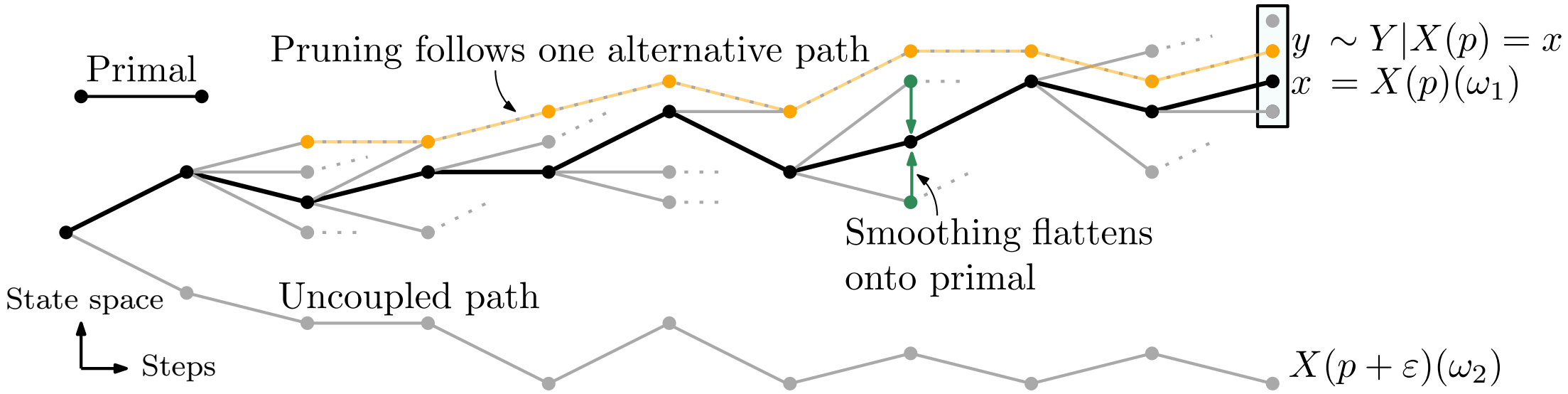
Equivalently, the weight accounts for the boundary derivative

In many cases: $\beta = \frac{\partial_\theta CDF_\theta(X(\theta))}{PDF_\theta(X(\theta))}$



Arya et al. 2210.08572

Automatic Differentiation



Correlated paths → low variance

$\mathcal{O}(1)$ unbiased forward mode AD

Arya et al. 2210.08572

Are these methods useful?

Toy Shower

Simplified particle shower:

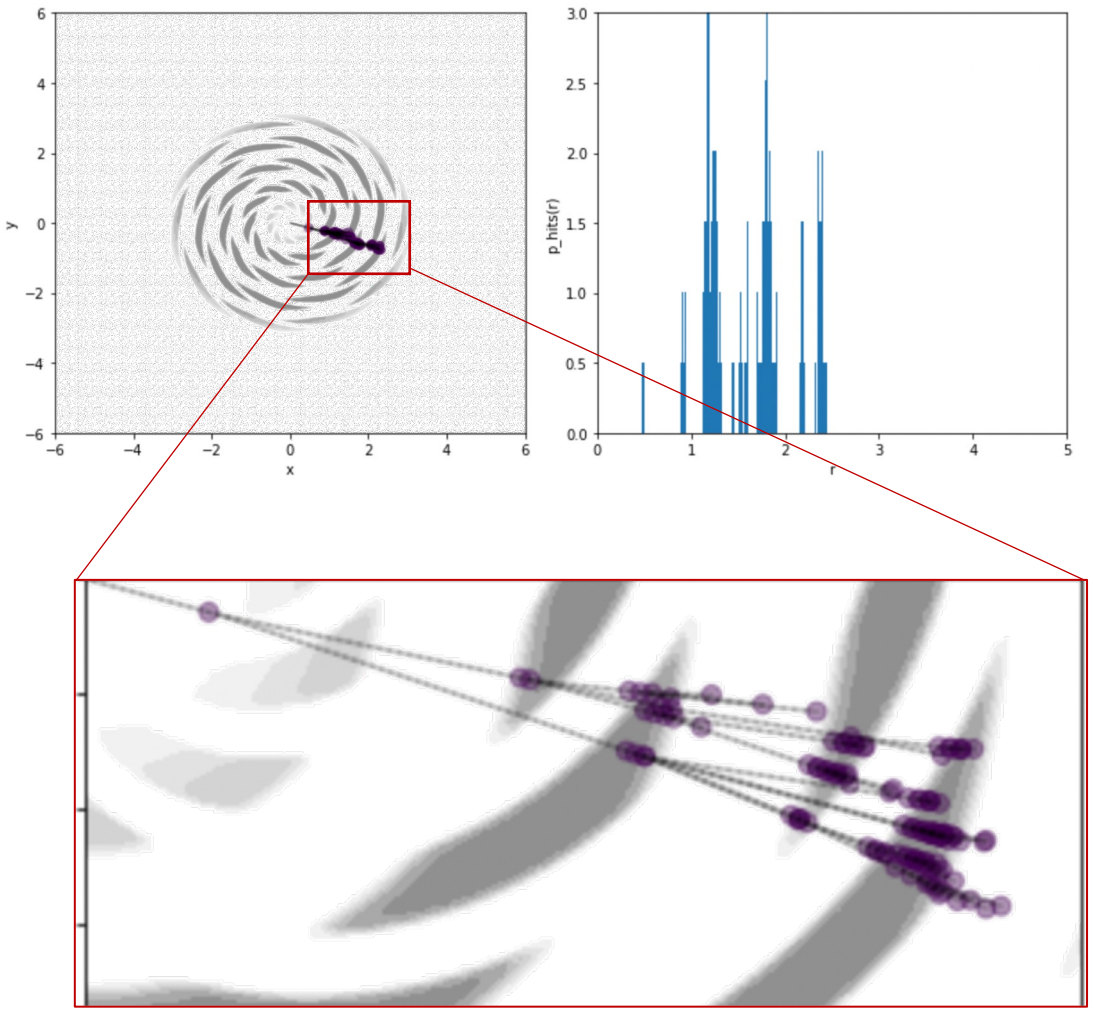
Including Energy loss and splitting

Design parameter:

Radial distance of material

Design goal:

Specify average shower depth



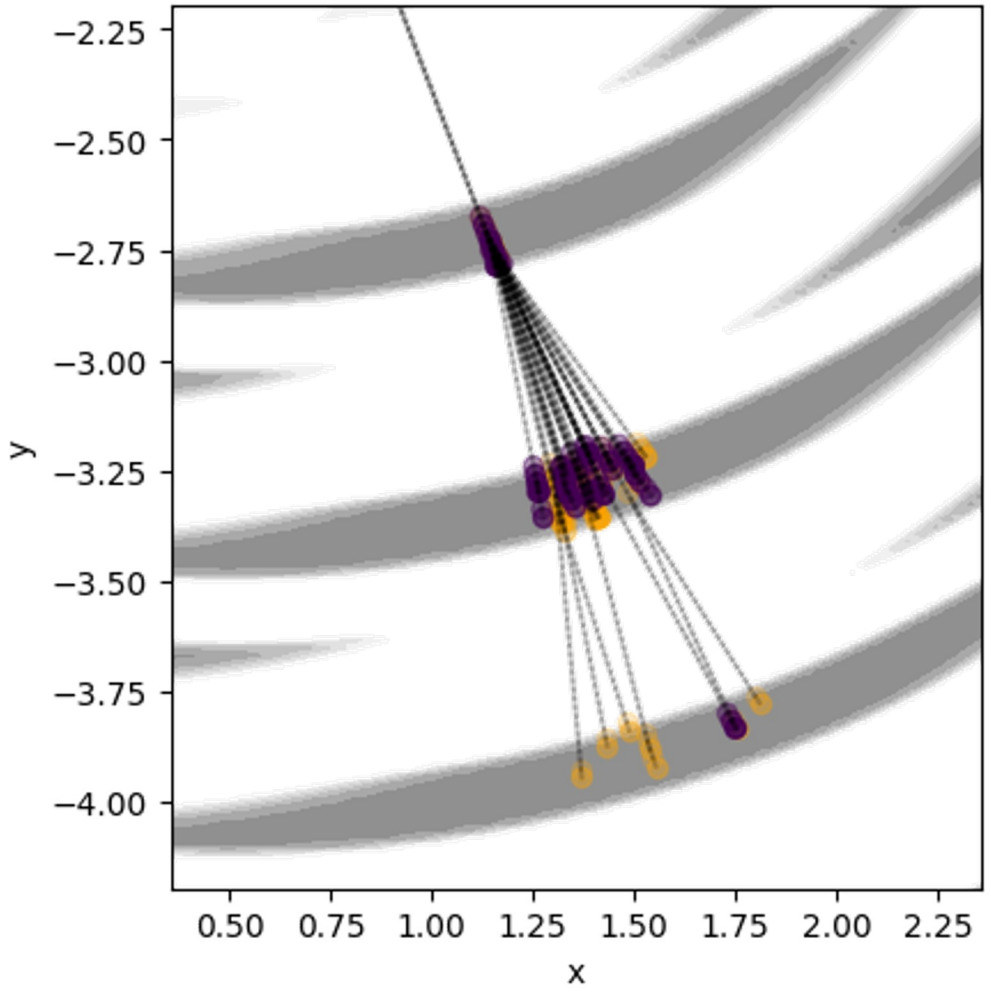
Toy Shower

Simplified particle shower:
Including Energy loss and splitting

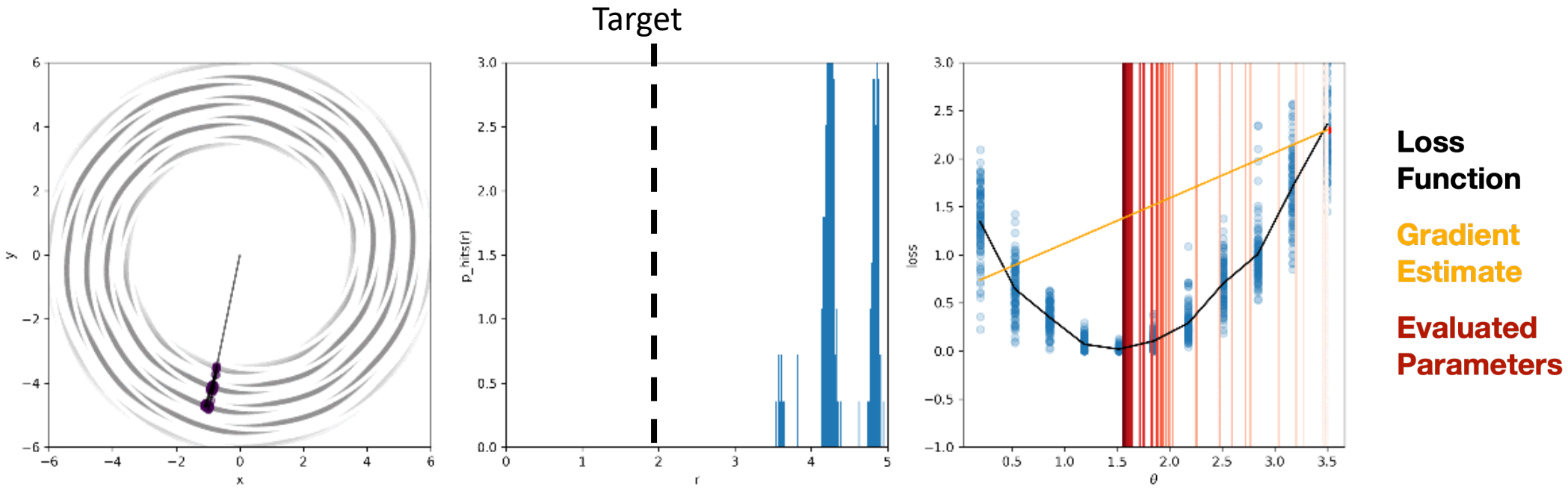
Design parameter:
Radial distance of material

Design goal:
Specify average shower depth

Dedicated implementation of Stochastic AD
→ Can generate “alternative showers”



Example Optimization

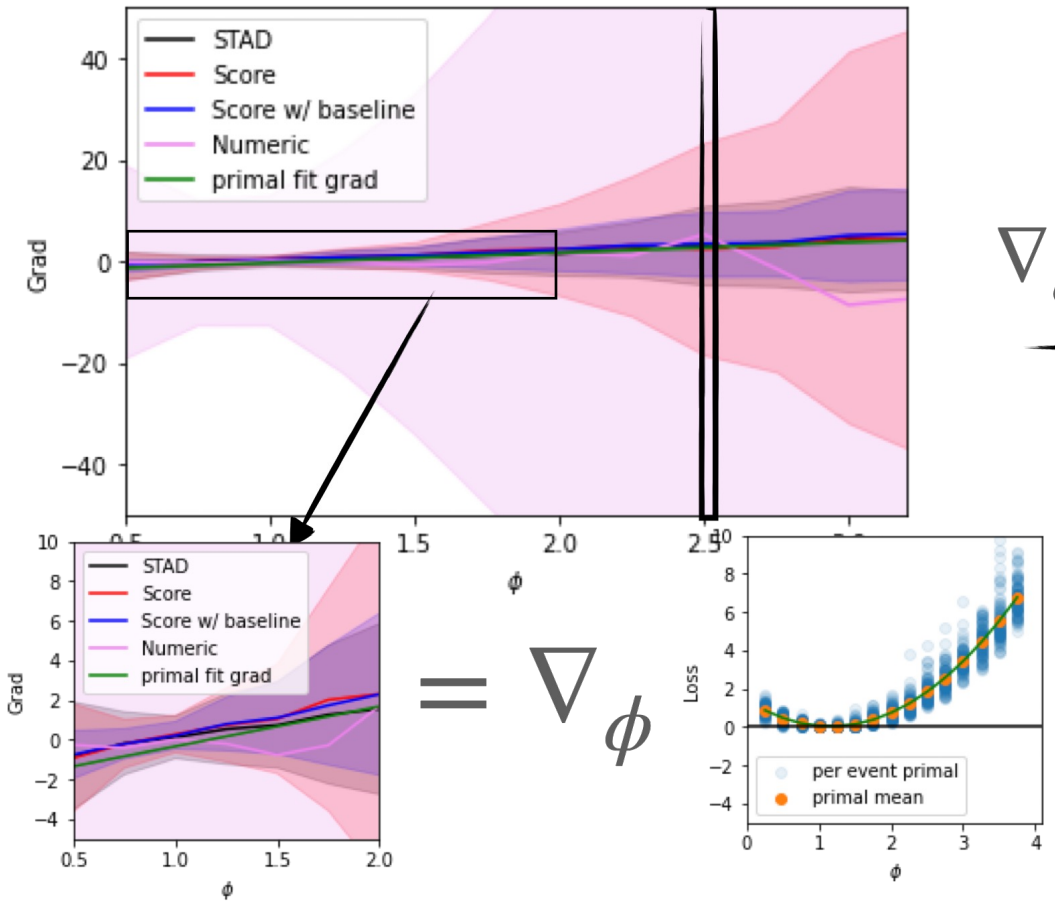


Gradients are noisy but in right direction (on average) → optimization works!

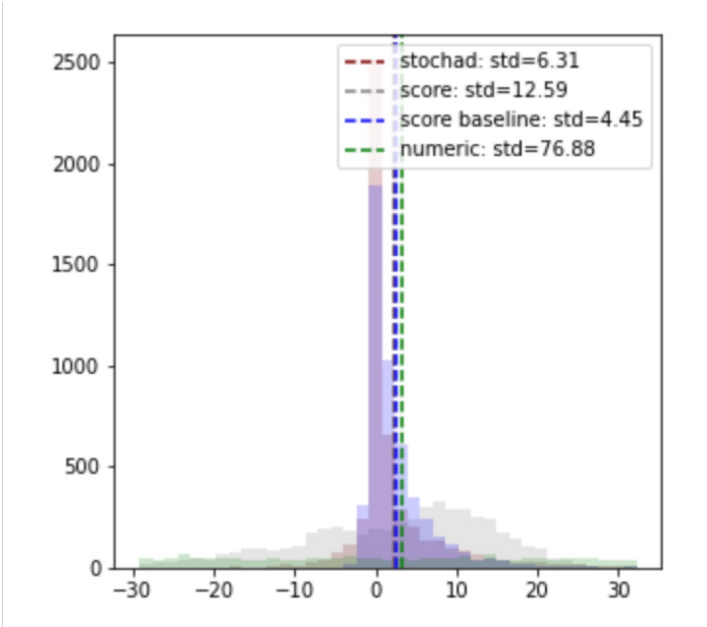
Comparisons

Both score function and Stochastic AD have reasonable variance gradients

Much to explore on how to *couple* primal & alternative programs to lower variance



$$\nabla_\phi L|_{\phi=2.5}$$



Summary

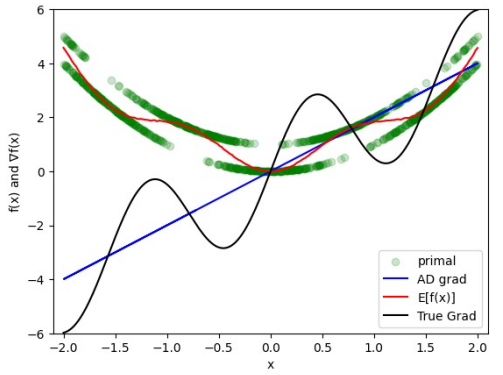
Programs with discrete stochasticity are all over HEP

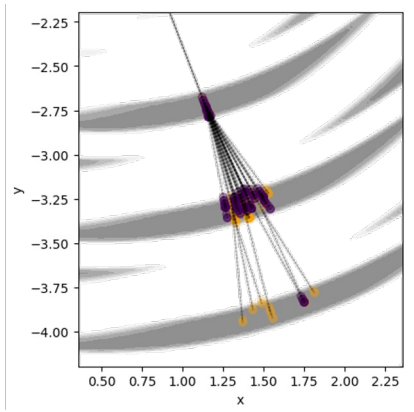
Discrete stochasticity is a problem for differentiation, but expected value of these programs (which is what we want anyway) may be differentiable

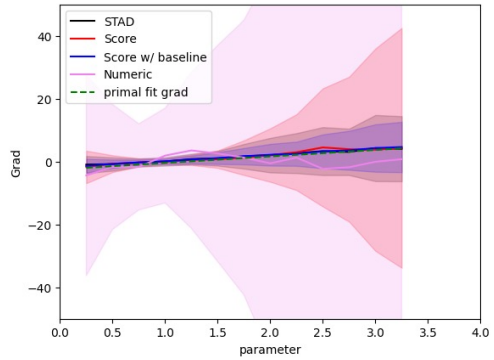
Variety of tools can potentially handle this, like score function or recent exciting work on Stochastic AD.

In a toy shower, we could successfully differentiate the program and perform optimization…

- Proof of principle… we may be able to scale this up!
- Still much work on reducing gradient variance







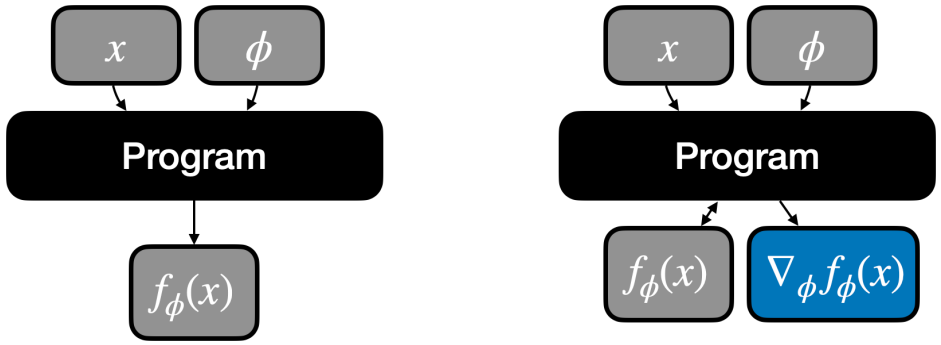
Backup

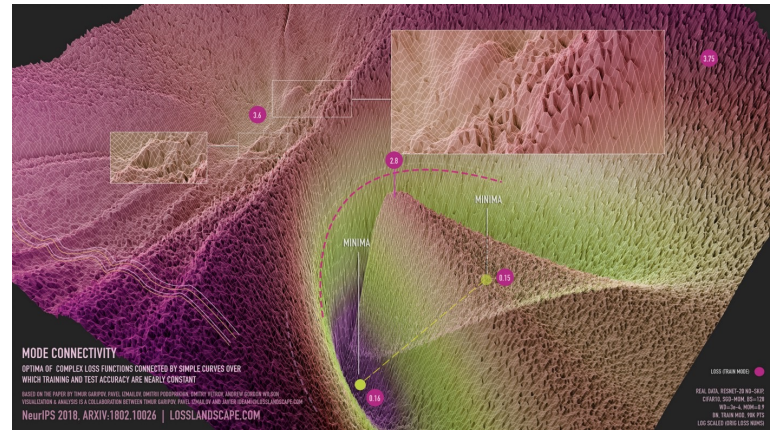
Differentiable Programming

Automatic differentiation super-charges code:

- Can compute gradients of numeric programs

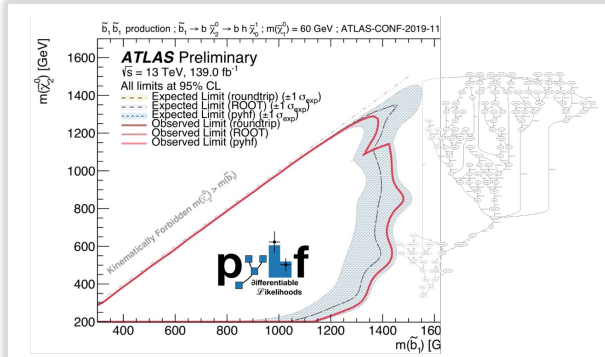
→ Major enabler of ML





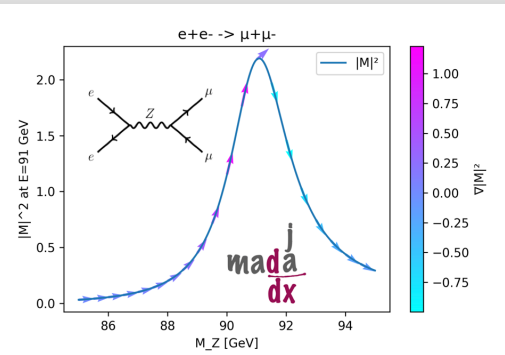
Differentiable programming applied to programs that are not (fully) NNs

→ powerful way to combine physics &ML



Differentiable Inference

10.21105/joss.02823



Differentiable Matrix Elements

2203.00057

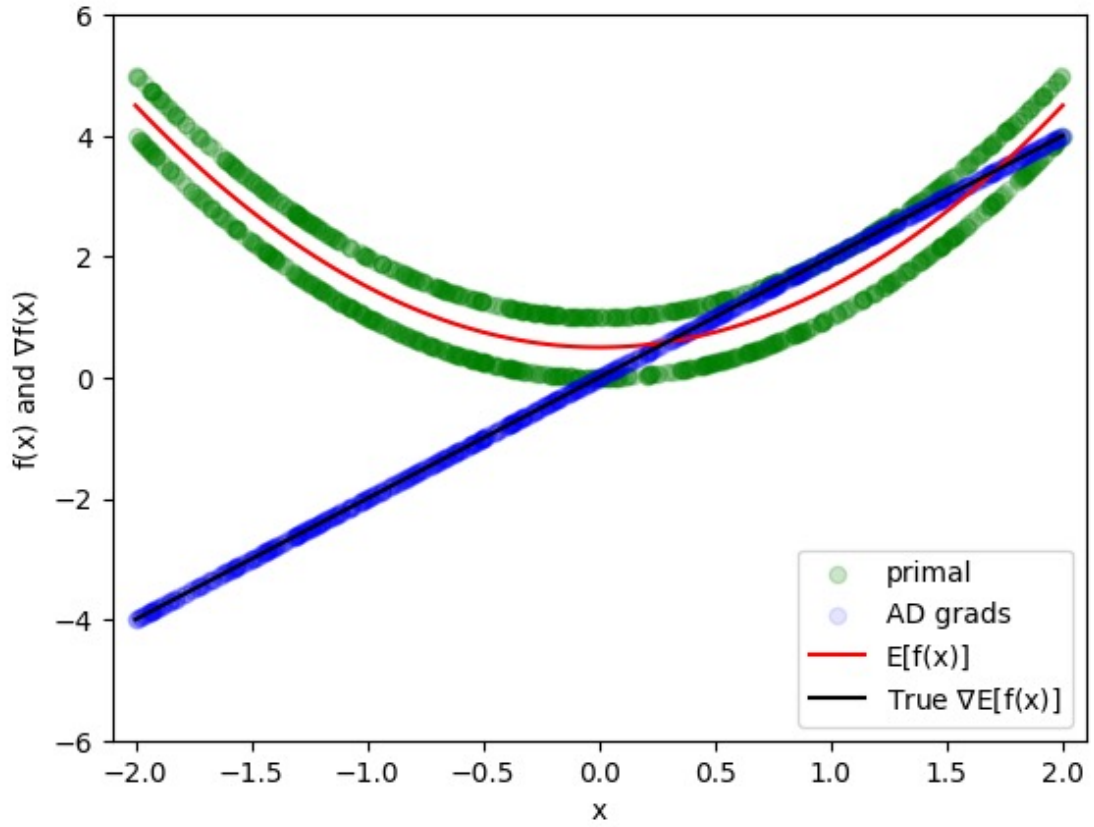
AD in Programs with Discrete Randomness


```
def f(x):
    theta = 0.5
    b = bernoulli(theta)
    g = x*x
    return g+b
```

AD Gradient: $grad(f(x_i))$

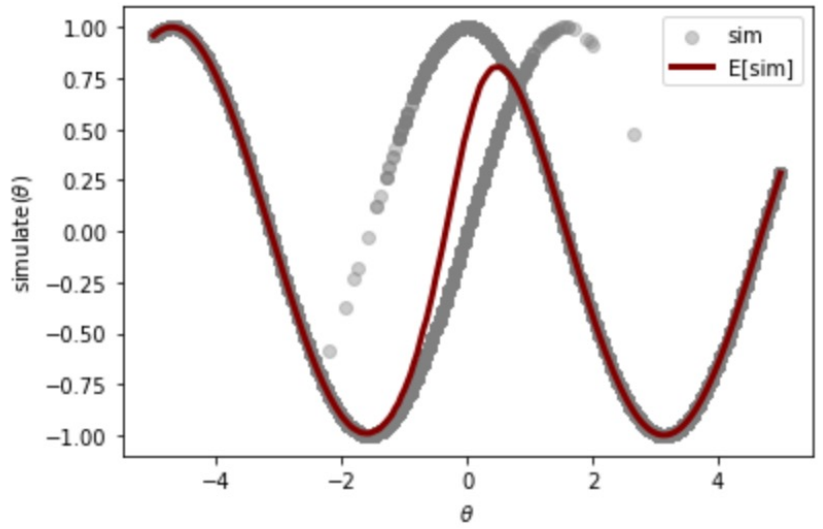
Expected value: $E[f(x)]$

True gradient: $\nabla_x E[f(x)]$

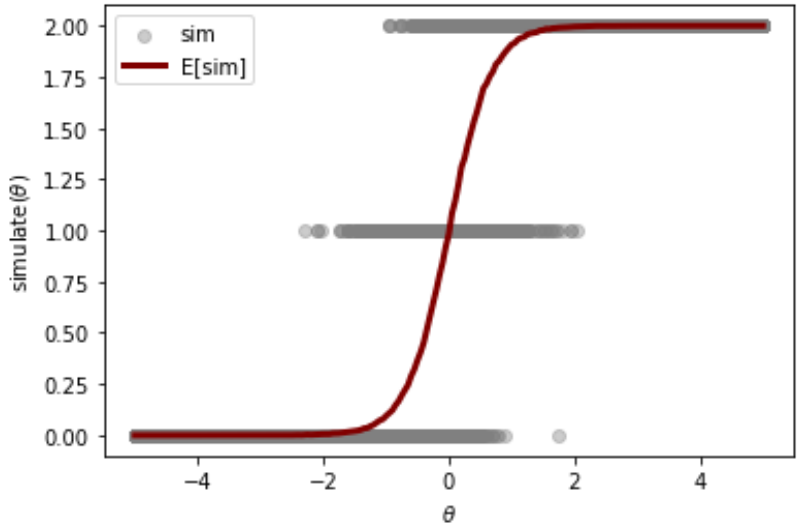


Programs with Discrete Randomness


```
def simulate(theta):
    p = sigmoid(theta)
    x = bernoulli(p) #0 or 1
    if x == 0:
        eval = sin(theta)
    else:
        eval = cos(theta)
    return eval
```




```
def program(theta):
    p = sigmoid(theta)
    x1 = bernoulli(p) #0/1
    x2 = bernoulli(p) #0/1
    eval = x1 + x2 # 0, 1 or 2
    return eval
```



Toy Shower


```
def simulate(t,ϕ){
    t' = propagate(t)
    dointeract ~ p(interact|material(x,y,z| ϕ))
    if dointeract:
        hits.append([x,y,z])
        dosplit ~ p(split|t')
        if dosplit:
            t1,t2 = split(t')
            simulate(t1), simulate(t2)
        else:
            E = (1-α) E // energy loss
            simulate(t')
}
```