CSC2547:
Learning to Search

Lecture 2: Background and gradient esitmators
Sept 20, 2019

Admin

- Course email: learn.search.2547@gmail.com
- Piazza: piazza.com/utoronto.ca/fall2019/csc2547hf
    - Good place to find project partners
        - leaves a paper trail of being engaged in course, asking good questions, bring helpful or knowledgable (for letters of rec)
- Project sizes: Groups of up to 4 are fine.
- My office hours: Mondays 3-4pm in Pratt room 384
    - TA office hours will have its own calendar

Due dates

- Assignment 1: Released Sept 24th, due Oct 3
- Project proposals: Due Oct 17th, returned Oct 28th
- Drop date: Nov 4th
- Project presentations: Nov 22nd and 29th
- Project due: Dec 10th

FAQs

- Q: I'm not registered / on the waitlist / auditing, can I still participate in projects or presentations?
    - A: Yes, as long as it doesn't make more grading + are paired with someone enrolled. Use piazza to find partners.
- Q: How can I make my long-term PhD project into a class project?
    - A: By making an initial proof of concept, possibly with fake / toy data

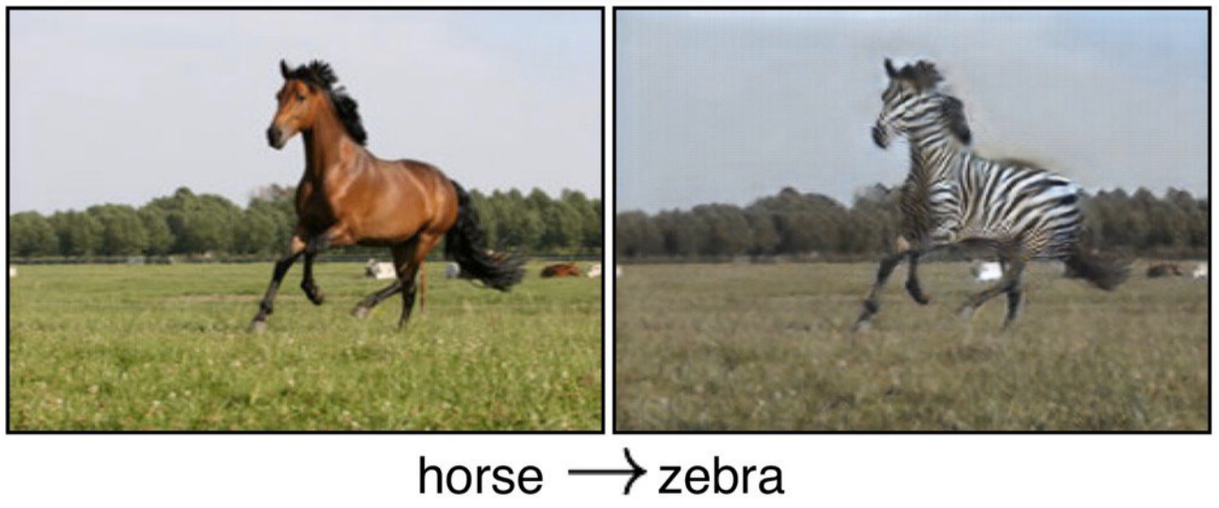
This week:
Course outline, and where we're stuck

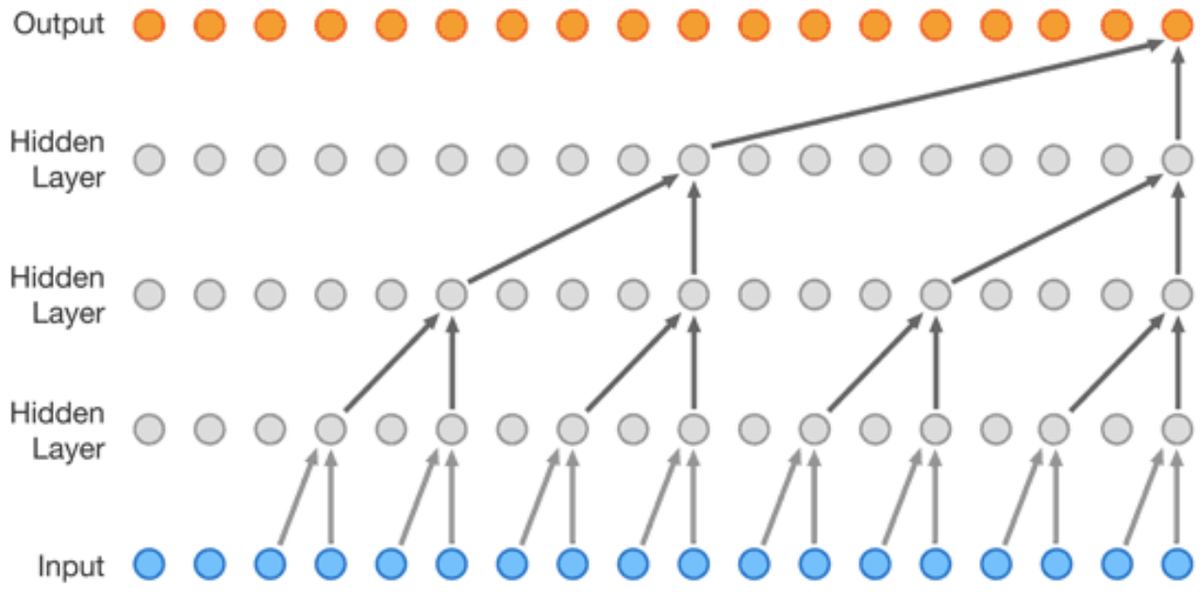
- The Joy of Gradients
  - Places we can't use them
- Outline of what we'll cover in course and why
- Detailed look at one approach to 'learning to search': RELAX, discuss where and why it stalled out

What recently became easy in machine learning?

- Training models with continuous intermediate quantities (hidden units, latent variables) to model or produce high-dimensional data (images, sounds, text)

- Discrete output mostly OK, Discrete hiddens or parameters are a no-no





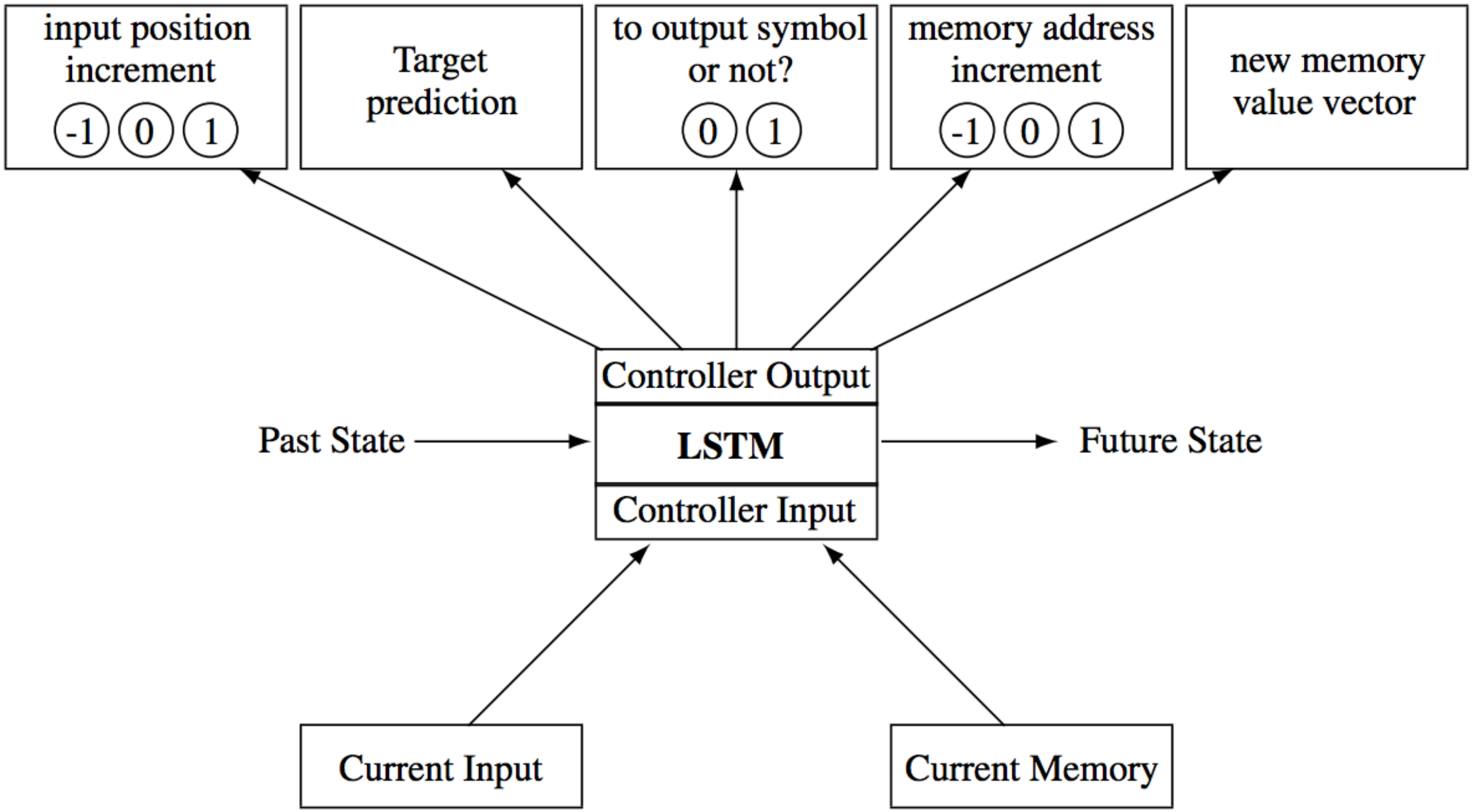
What is still hard?

- Training GANs to generate text
- Training VAEs with discrete latent variables
- Training agents to communicate with each other using words
- Training agent or programs to decide which discrete action to take.
- Training generative models of structured objects of arbitrary size, like programs, graphs, or large texts.

Level	Model	PTB	CMU-SE
Word	LSTM	what everything they take everything away from . may tea bill is the best chocolate from emergency . can you show show if any fish left inside . room service , have my dinner please .	<s>will you have two moment ? </s> <s>i need to understand deposit length . </s> <s>how is the another headache ? </s> <s>how there , is the restaurant popular this cheese ? </s>
	CNN	meanwhile henderson said that it has to bounce for. I'm at the missouri burning the indexing manufacturing and through .	<s>i 'd like to fax a newspaper . </s> <s>cruise pay the next in my replacement . </s> <s>what 's in the friday food ? ? </s>

Table 4: Word level generations on the Penn Treebank and CMU-SE datasets

Adversarial Generation of Natural Language.
Sai Rajeswar, Sandeep Subramanian, Francis Dutil,
Christopher Pal, Aaron Courville, 2017



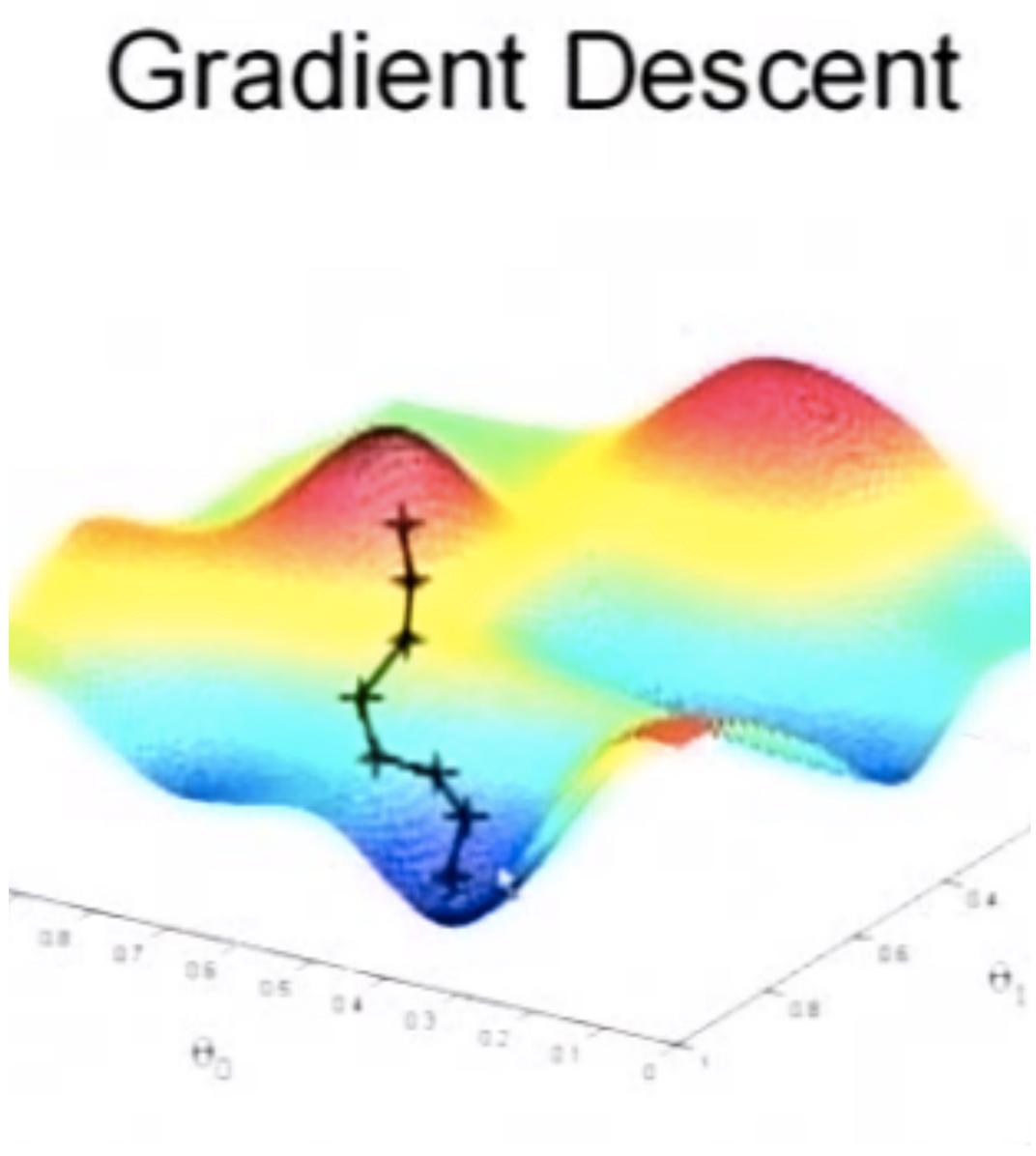
"We successfully trained the RL-NTM to solve a number of algorithmic tasks that are simpler than the ones solvable by the fully differentiable NTM."

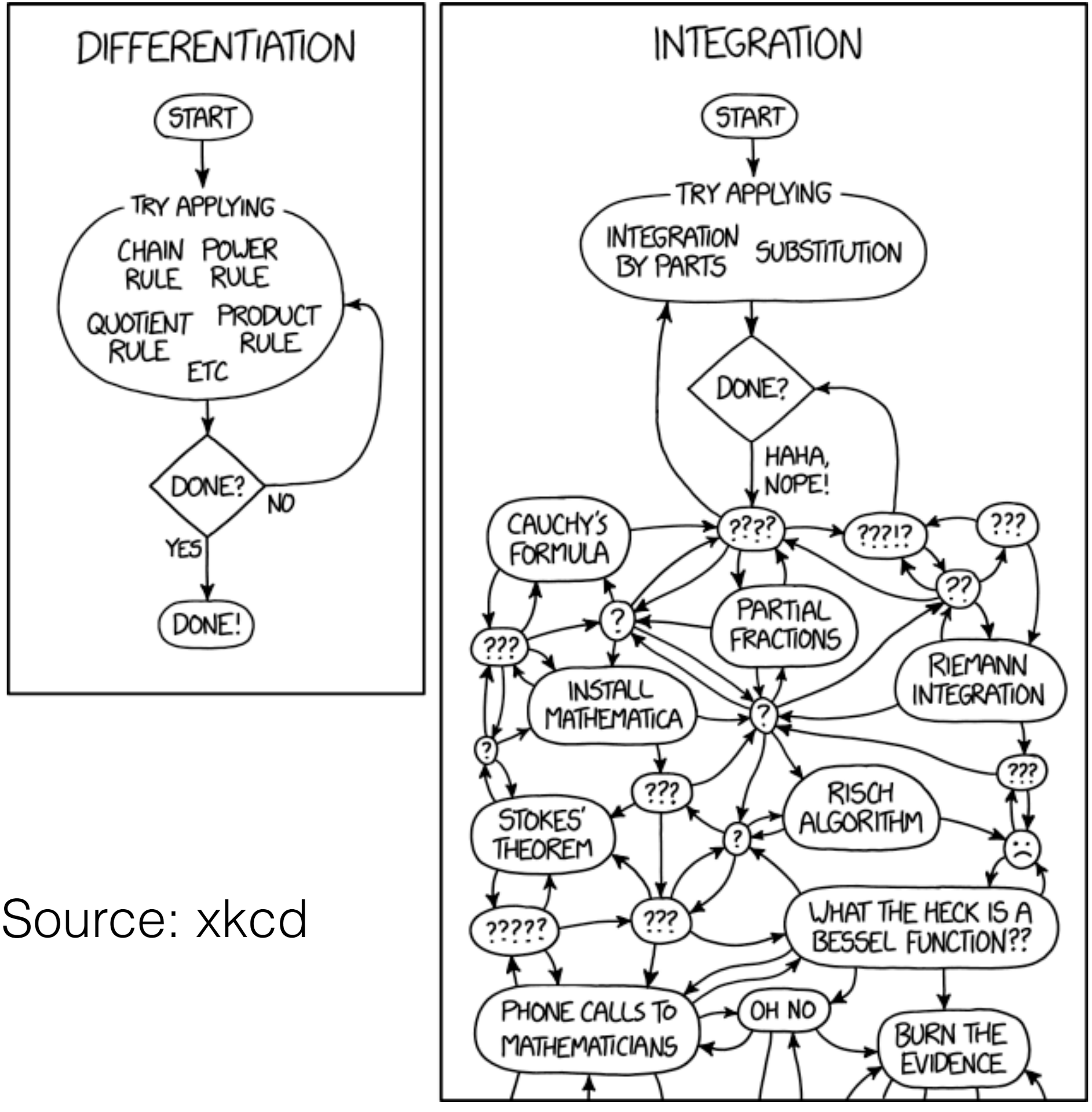
Reinforcement Learning Neural Turing Machines

Wojciech Zaremba, Ilya Sutskever, 2015

Why are the easy things easy?

- Gradients give more information the more parameters you have
- Backprop (reverse-mode AD) only takes about as long as the original function
- Local optima less of a problem than you think




DIFFERENTIATION
START
TRY APPLYING
CHAIN RULE
POWER RULE
QUOTIENT RULE
PRODUCT RULE
ETC
DONE?
NO
YES
DONE!
INTEGRATION
START
TRY APPLYING
INTEGRATION BY PARTS
SUBSTITUTION
DONE?
HAHA, NOPE!
CAUCHY'S FORMULA
????
???!?
???
??
?
???
PARTIAL FRACTIONS
INSTALL MATHEMATICA
RIEMANN INTEGRATION
RISCH ALGORITHM
STOKES' THEOREM
?????
WHAT THE HECK IS A BESSEL FUNCTION??
PHONE CALLS TO MATHEMATICIANS
OH NO
BURN THE EVIDENCE

Source: xkcd

Gradient descent

- Cauchy (1847)

Begin with some initial θ_0
$t \leftarrow 0$
while not converged:

- Pick a step size η_t
- $\theta_{t+1} \leftarrow \theta_t - \eta_t \nabla f(\theta_t)$
- $t \leftarrow t + 1$

Gradient-based optimization scales with dimension

With reverse-mode differentiation (backprop):

Expression	Time cost	Scalars returned
$f(\mathbf{x})$	1	1
$\nabla f(\mathbf{x})$	~ 2	D

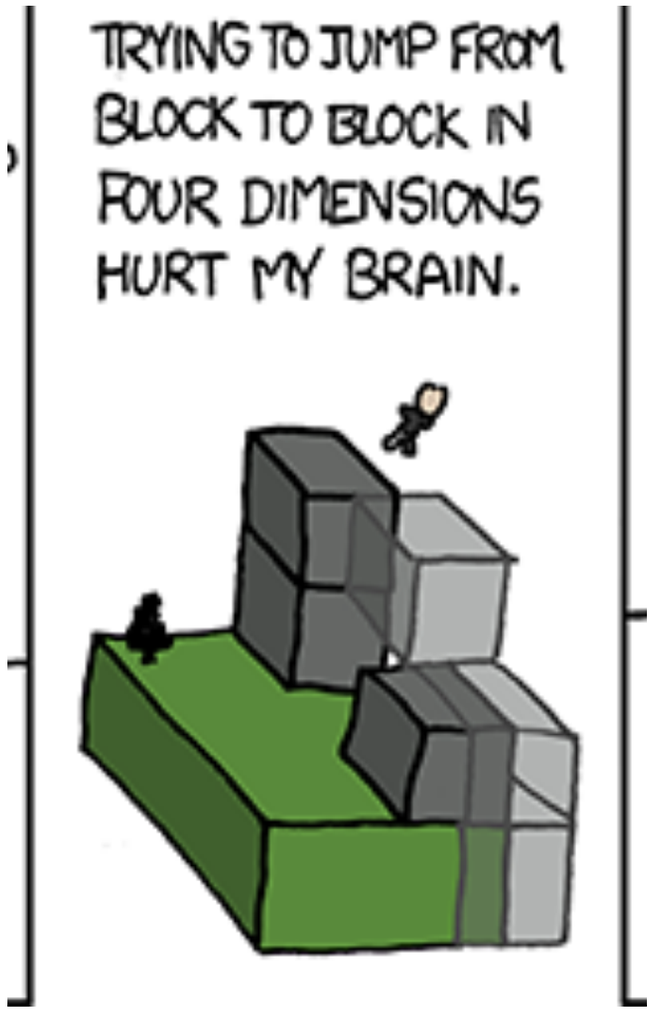
Why are the hard things hard?

- Discrete variables means we can't use backdrop to get gradients

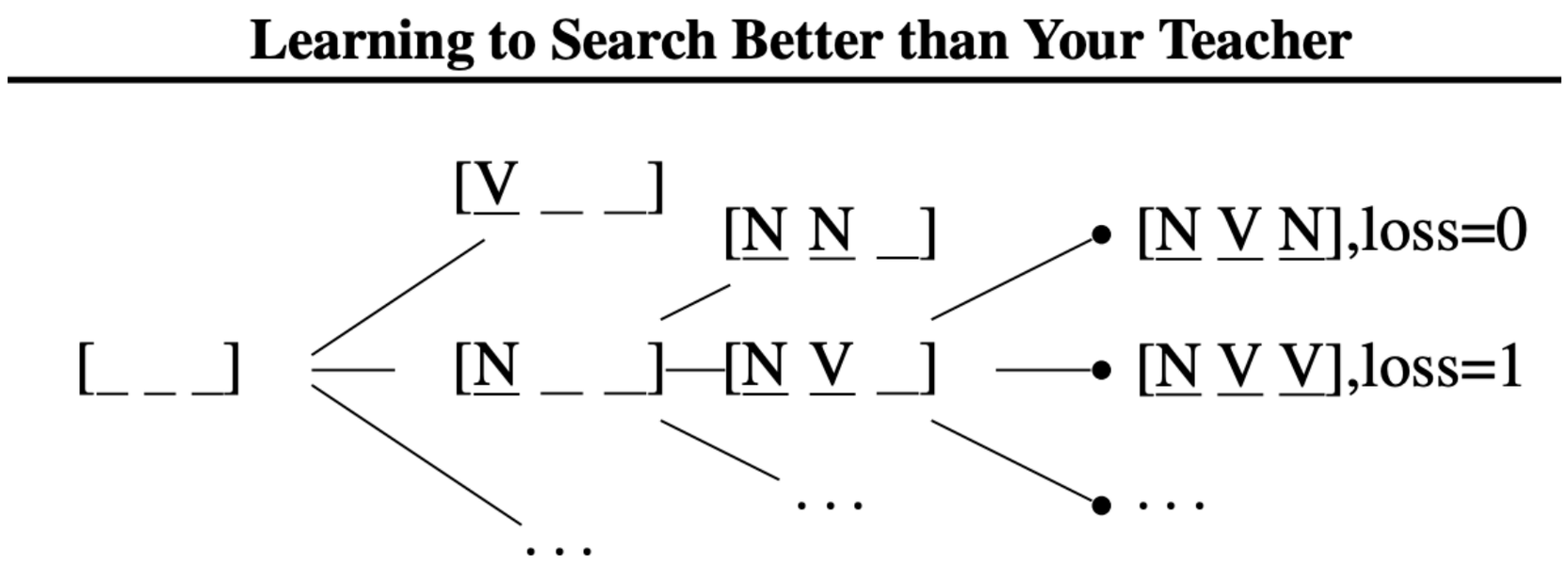
- No cheap gradients means that we don't know which direction to move to improve

- Not using our knowledge of the structure of the function being optimized

- Becomes as hard as optimizing a black-box function



Scope of applications:



- Any problem with a large search space, and a well-defined objective that can’t be evaluated on partial inputs.
    - e.g. SAT solving, proof search, writing code, neural architecture design

An illustration of the search space of a sequential tagging example that assigns a part-of-speech tag sequence to the sentence “John saw Mary.” Each state represents a partial labeling. The start state b = [ ] and the set of end states E = {[N V N], [N V V], . . .}. Each end state is associated with a loss. A policy chooses an action at each state in the search space to specify the next state.

Questions I want to understand better

- Current state of the art in
  - MCTS
  - SAT solving
  - program induction
  - planning
  - curriculum learning
  - adaptive search algorithms

Week 3: Monte Carlo Tree Search and applications

- Background, AlphaZero, thinking fast and slow
- Applications to:
    - planning chemical syntheses
    - robotic planning (sort of)
- Recent advances

Week 4: Learning to SAT Solve and Prove Theorems

- Learning neural nets to guess which assignments are satisfiable / if any a clause is satisfiable
    - Can convert to any NP-complete problem
    - Need higher-order logic to prove Reimann Hypothesis?
- Overview of theorem-proving environments, problems, datasets
- Overview of literature:
    - RL approaches
    - Continuous embedding approaches
    - Curriculum learning
- Less focus on relaxation-based approaches

What can we hope for?

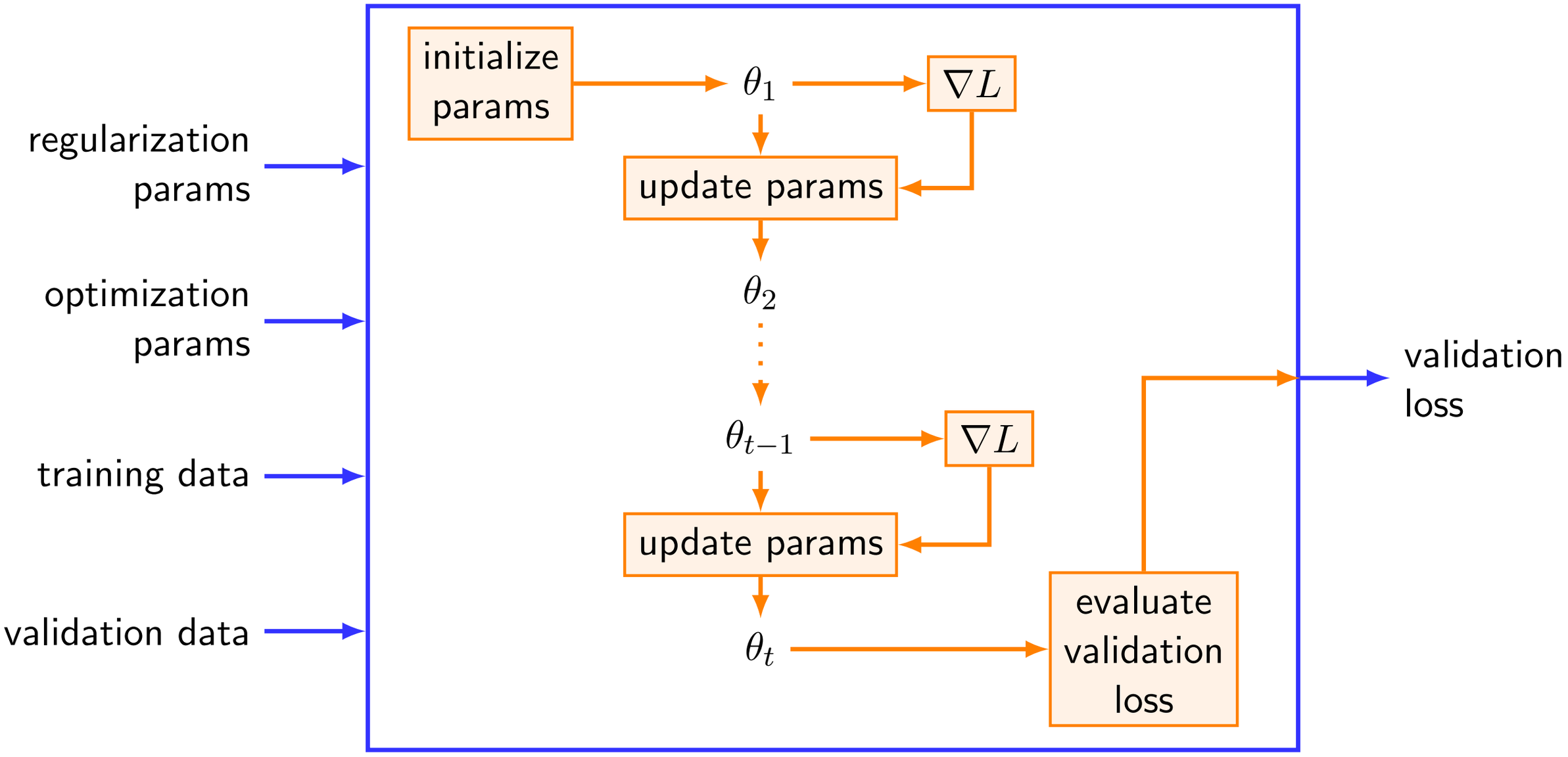
- Searching, inference, SAT are all NP-Hard
- What success looks like:
  - A set of different approaches with different pros and cons
  - Theoretical and practical understand of what methods to try and when
  - Ability to use any side information or re-use previous solutions

Week 5: Nested continuous optimization

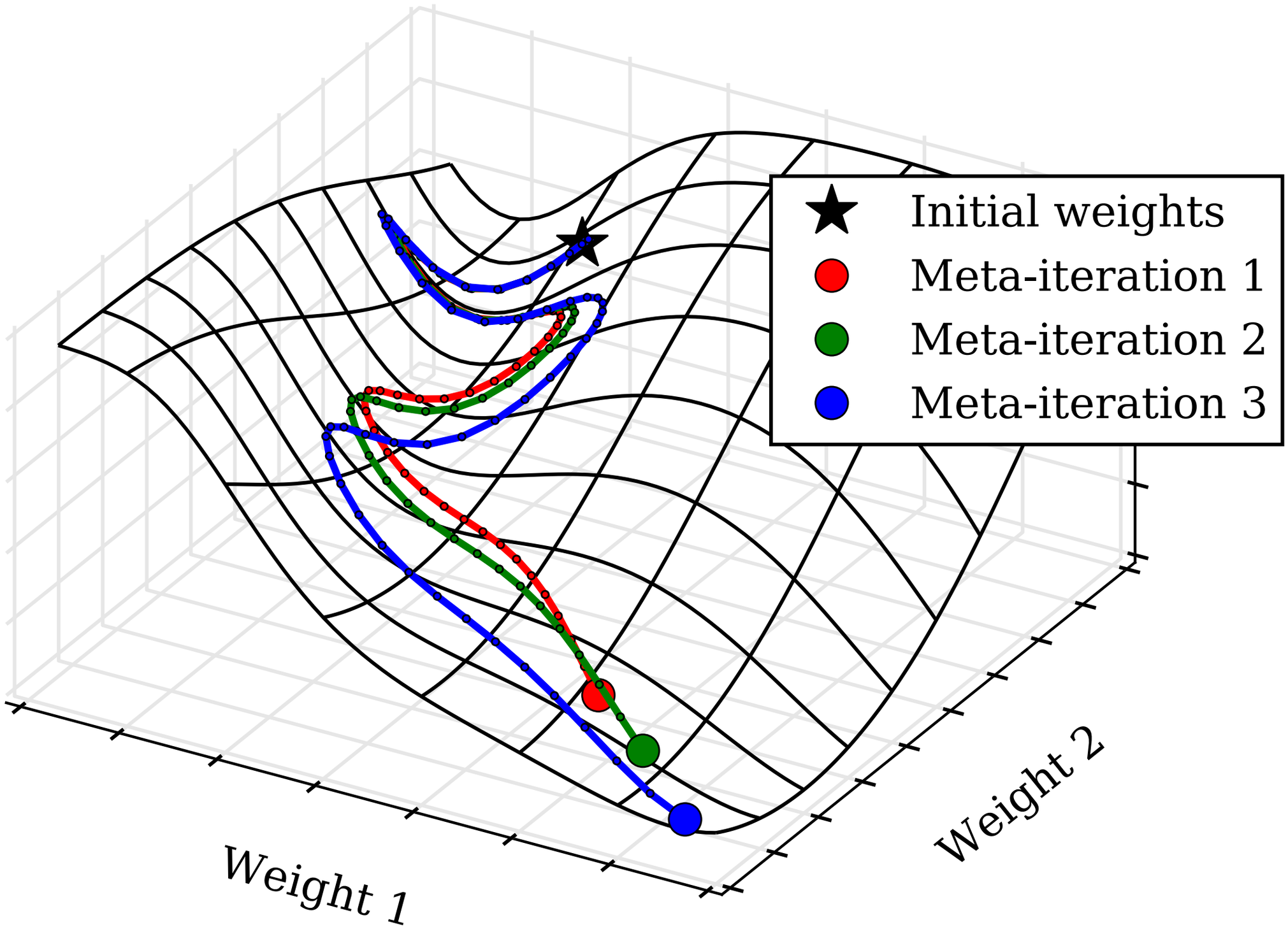
- Training GANs, hyperparameter optimization, solving games, meta-learning can all be cast as optimizing and optimization procedure.

- Three main approaches:
    - Backprop through optimization (MAML, sort of)
    - Learn a best-response function (SMASH, Hypernetworks)
    - Use implicit function theorem (iMAML, deep equilibirum models)
        - need inverse of Hessian of inner problem at optimum
- Game theory connections (Stackleberg Games)

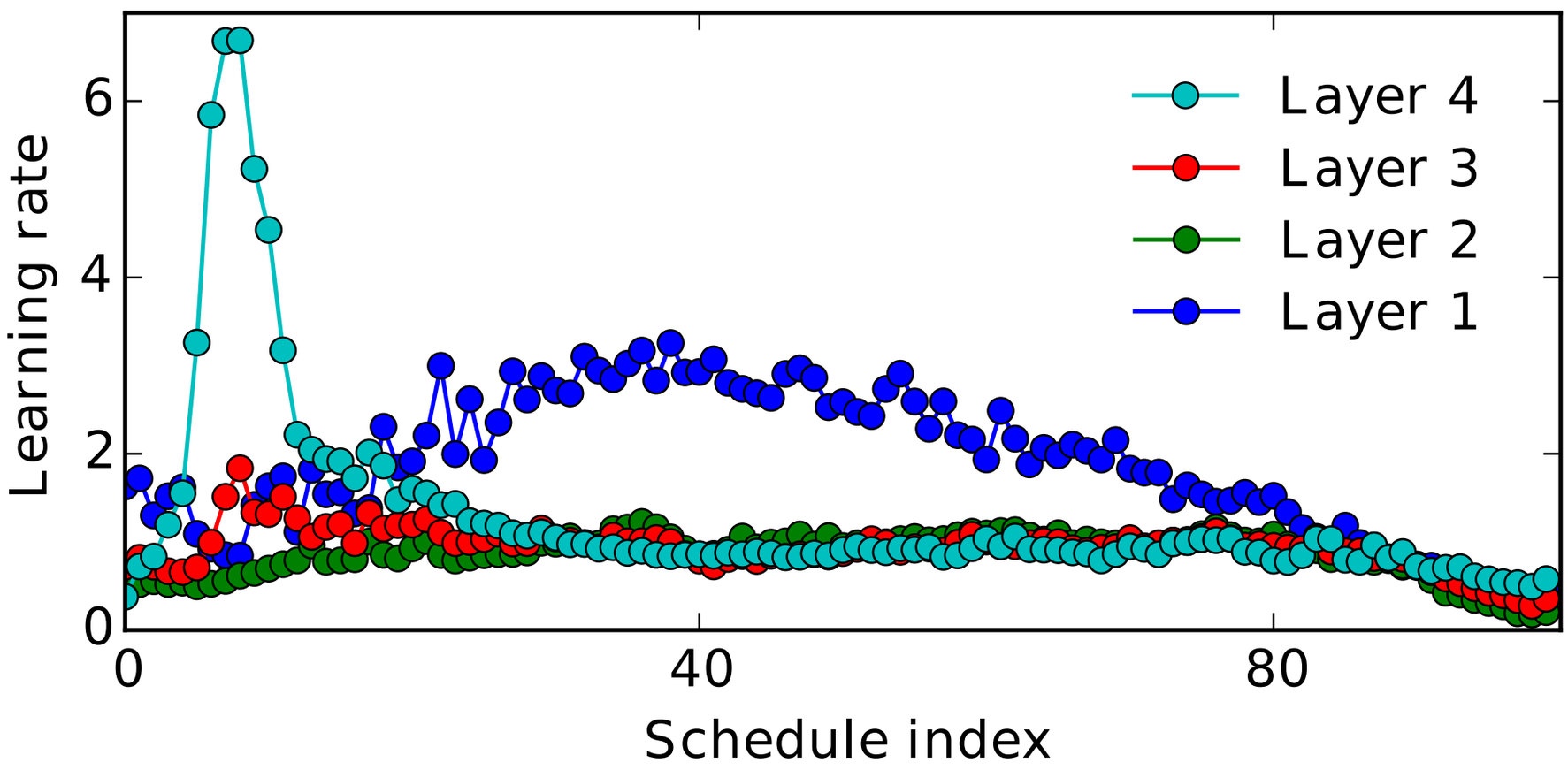
1. Snoek, Larochelle and Adams, Practical Bayesian Optimization of Machine Learning Algorithms, NIPS 2012
2. Golovin *et al.*, Google Vizier: A Service for Black-Box Optimization, SIGKDD 2017
3. Bengio, Gradient-Based Optimization of Hyperparameters, Neural Computation 2000
4. Domke, Generic Methods for Optimization-Based Modeling, AISTATS 2012

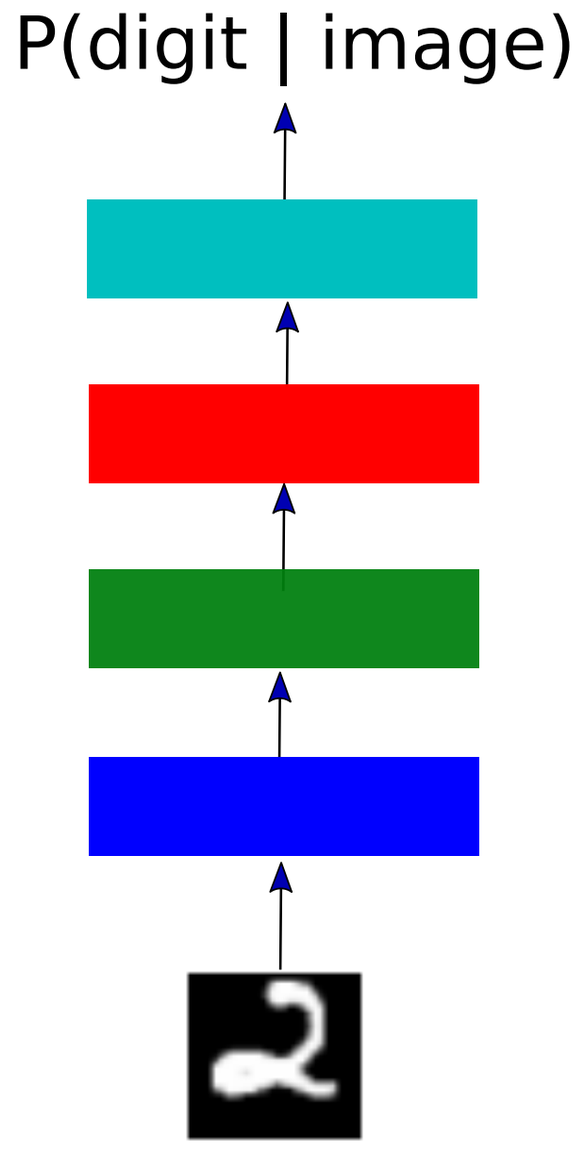


1. Snoek, Larochelle and Adams, Practical Bayesian Optimization of Machine Learning Algorithms, NIPS 2012
2. Golovin *et al.*, Google Vizier: A Service for Black-Box Optimization, SIGKDD 2017
3. Bengio, Gradient-Based Optimization of Hyperparameters, Neural Computation 2000
4. Domke, Generic Methods for Optimization-Based Modeling, AISTATS 2012


Initial weights
Meta-iteration 1
Meta-iteration 2
Meta-iteration 3
Weight 1
Weight 2

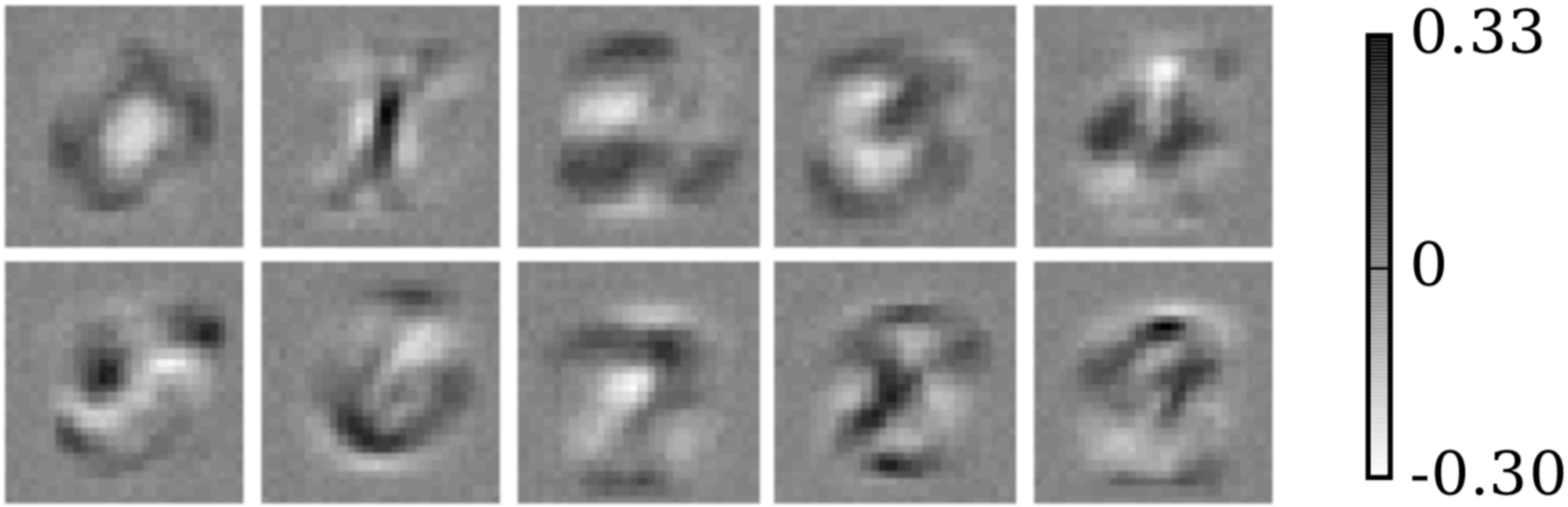
Optimized training schedules





Optimizing training data

- Training set of size 10 with fixed labels on MNIST
- Started from blank images



More project ideas

- Using the approximate implicit function theorem to speed up training of GANs. E.g. iMaml

Week 6: Active learning, POMDPs, Bayesian Optimization

- Distinction between exploration and exploitation dissolves under Bayesian decision theory: Planning over what you'll learn and do.

- Hardness results

- Approximation strategies:

  - One-step heuristics (expected improvement, entropy reduction)

  - Monte-Carlo planning

- Differentiable planning in continuous spaces

More project ideas

- Efficient nonmyopic search: “On the practical side we just did one-step lookahead with a simple approximation, a lot you could take from the approximate dynamic programming literature to make things work better in practice with roughly linear slowdowns I think.”

Week 7: Evolutionary approaches and Direct Optimization

- Genetic algorithm is a vague class of algorithms, very flexible
    - Fun to tweak, hard to get to work
- Recent connection of one type (Evolution Strategies) to standard gradient estimators, optimizing a surrogate function
- Direct optimization: A general strategy for estimating gradients through discrete optimization, involving a local discrete search

Aside: Evolution Strategies optimize a linear surrogate

$$\hat{w} = (X^\top X)^{-1} X^\top y$$

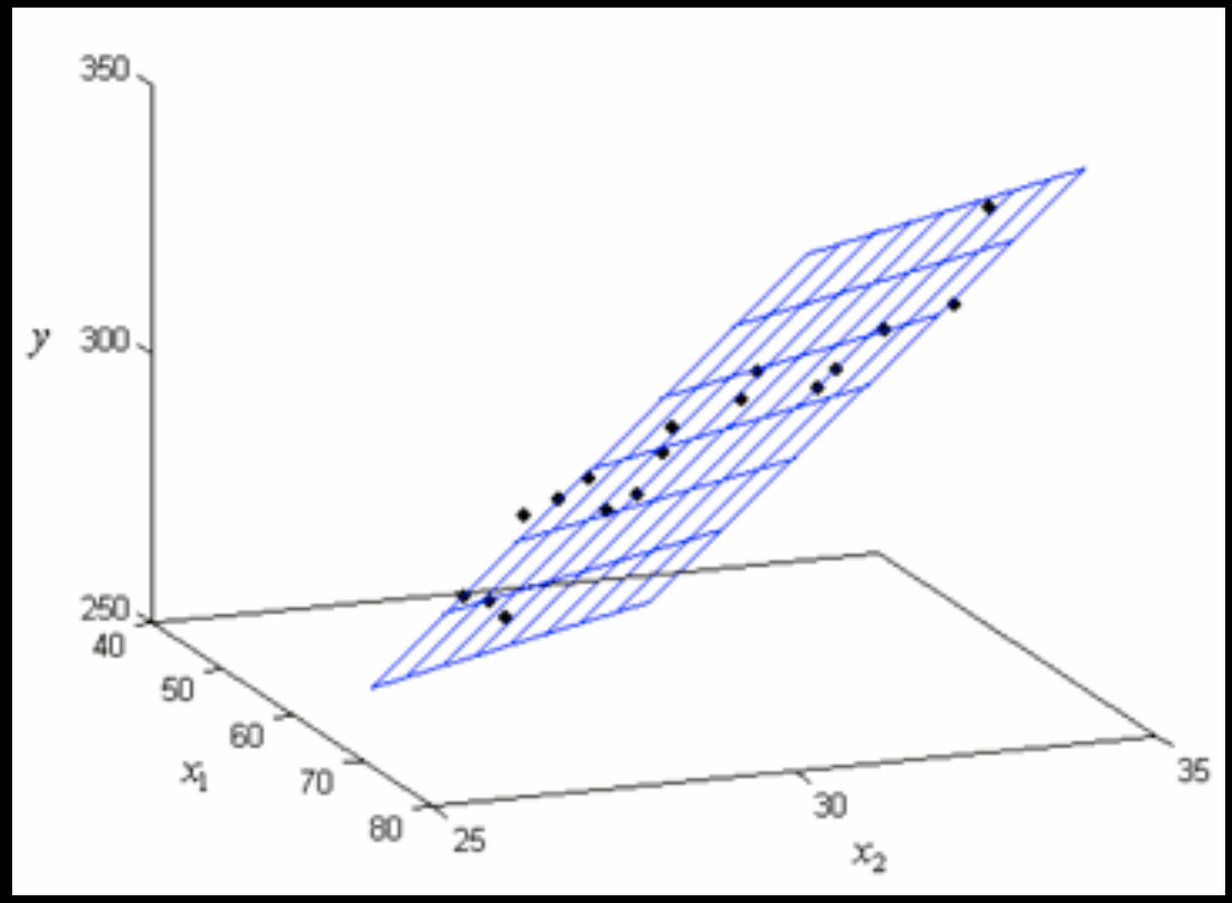
$$\approxeq \mathbb{E}\left[(X^\top X)\right]^{-1} X^\top y$$

$$= \left[I\sigma^2\right]^{-1} X^\top y$$

$$= \left[I\sigma^2\right]^{-1} (\epsilon\sigma)^\top y$$

$$= \sum_i \frac{\epsilon_i y_i}{\sigma}$$

$$= \sum_i \frac{\epsilon_i f(\epsilon_i \sigma)}{\sigma}$$

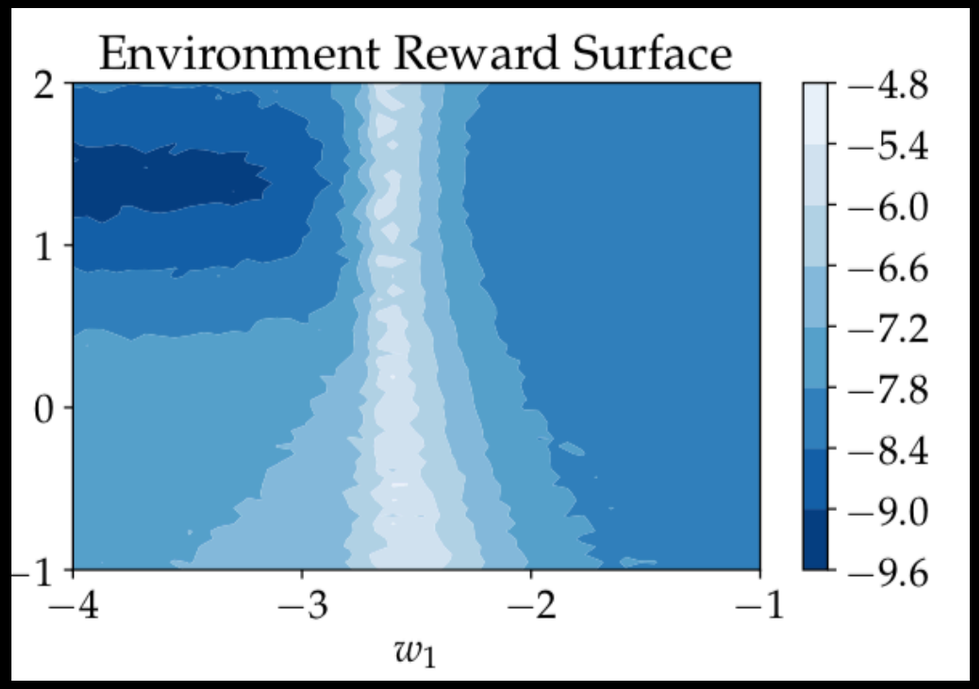


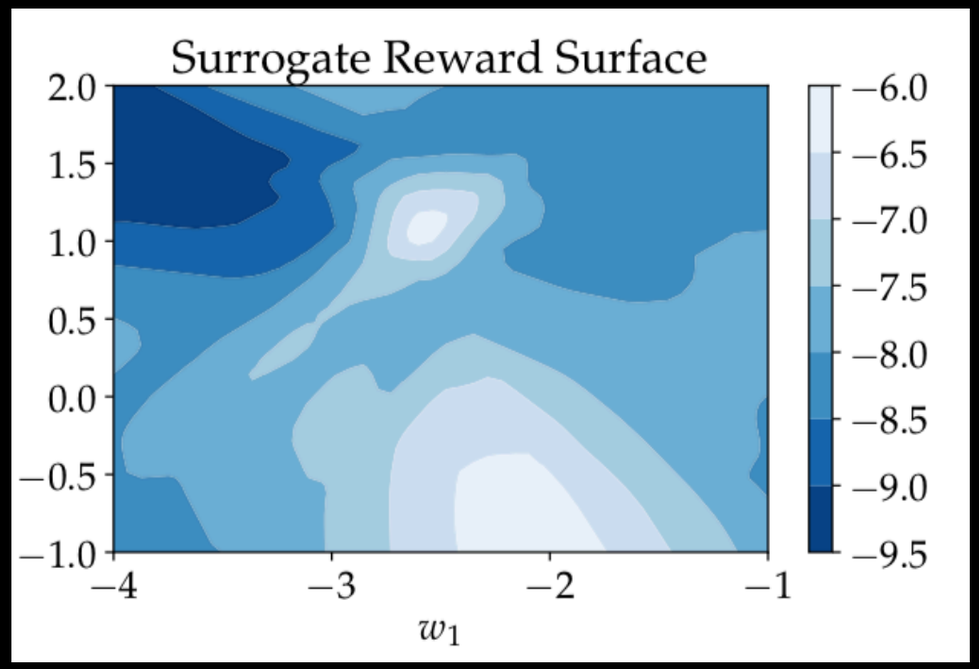
$$\epsilon \sim \mathcal{N}(0, I)$$

$$x = \epsilon\sigma$$

Aside: Evolution Strategies optimize a linear surrogate

- Throws away all observations each step
- Use a neural net surrogate, and experience replay
- Distributed ES algorithm works for any gradient-free optimization algorithm
- w/ students Geoff Roeder, Yuhuai (Tony) Wu, Jaiming Song





More project ideas

- Generalize evolution strategies to non-linear surrogate functions

Week 8: Learning to Program

- So hard, I'm putting it at the end. Advanced projects.

- Relaxations (Neural Turing Machines) don't scale. Naive RL approaches (trial and error) don't work

- Can look like proving theorems (Curry-Howard correspondence)

    - Fun connections to programming languages, dependent types

- Lots of potential for compositionality, curriculum learning

Week 9: Meta-reasoning

- Playing chess: Which piece to think about moving? Need to think about that.

- Proving theorems: Which lemma to try to prove first? Need to think about that.

- Bayes' rule is no help here.

- Few but excellent works:

  - Stuart Russel + Students (Meta-reasoning)

  - Andrew Critch + friends (Reasoning about your own future beliefs about mathematic statements)

Week 10: Asymptotically Optimal Algorithms

- Fun to think about, hard to implement.

- Goedel machine: Spend 50% of time searching for software updates of yourself that will provably improve expected performance. Run one whenever found.

  - How to approximate?

- AIXI: Use Bayesian decision theory on the most powerful computable prior: set of all computable programs.

  - How to approximate?

Questions

Break / Sign up for Learning to SAT solve + thm prove

Backpropagation Through

Backpropagation Through THE VOID

Backpropagation Through the Void

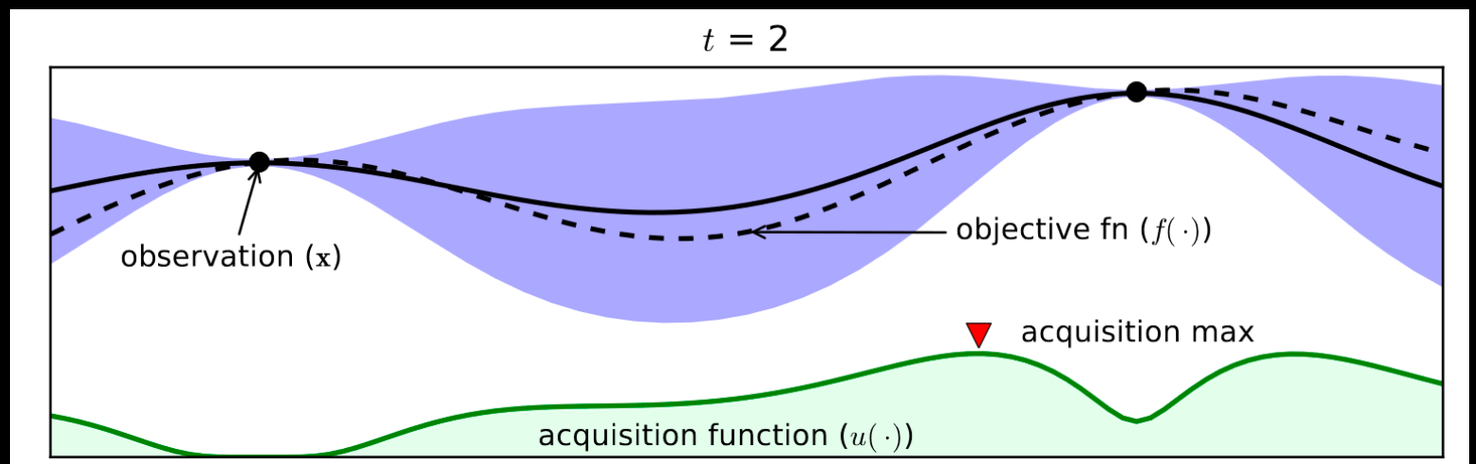
Will Grathwohl
Dami Choi
Yuhuai Wu
Geoff Roeder
David Duvenaud

Where do we see this guy?

$$\mathcal{L}(\theta) = \mathbb{E}_{p(b|\theta)}[f(b)]$$

- Variational Inference
- Hamiltonian Monte Carlo
- Policy Optimization
- Hard Attention

Bayesian optimization doesn't scale yet



Shahriari et al., 2016

- Bayesopt is usually expensive, relative to model evals
- Global surrogate models not good enough in high dim.
- Even for expensive black-box functions, gradient-based optimization is embarrassingly competitive
- Can we add some cheap model-based optimization to REINFORCE?

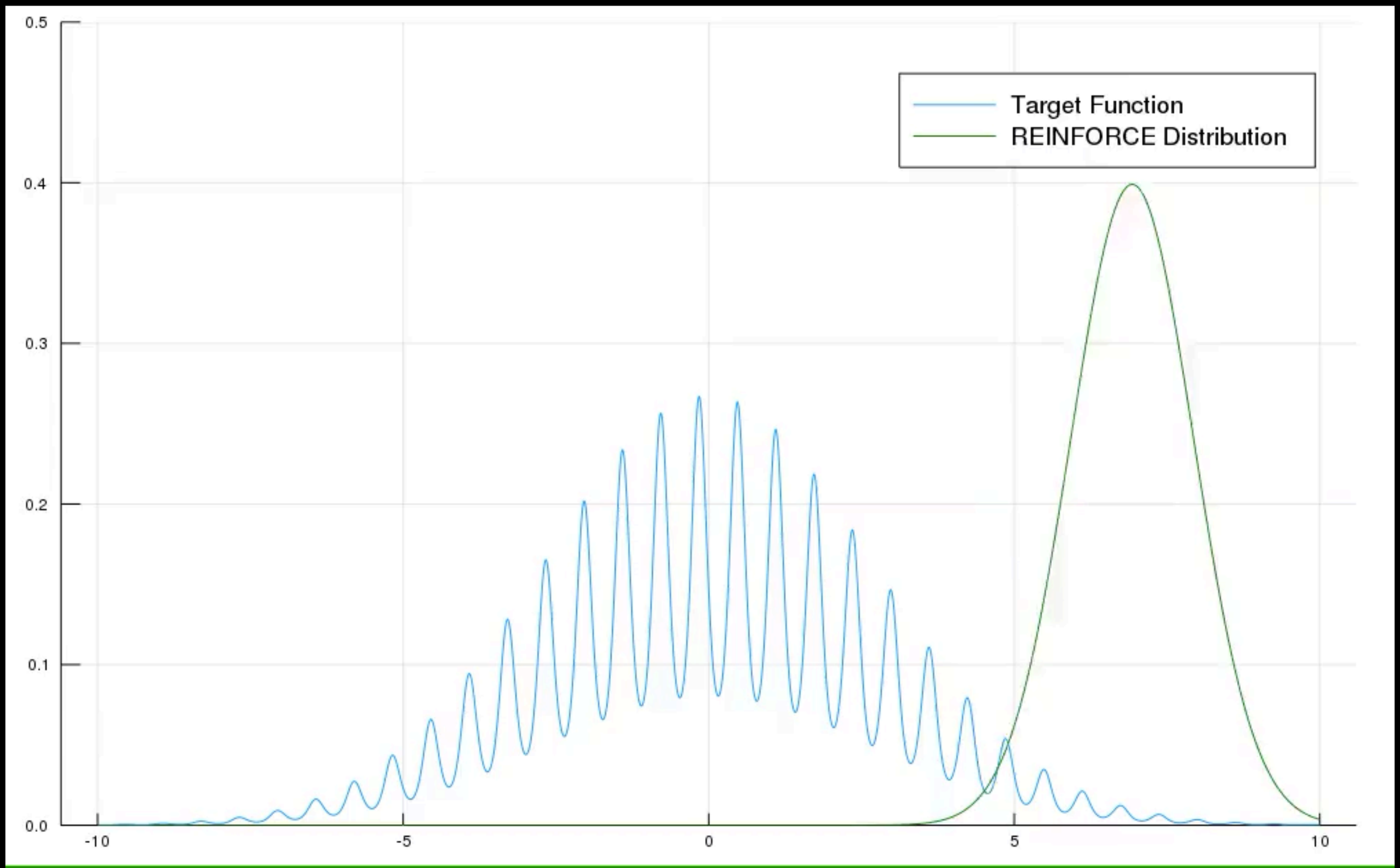
REINFORCE (Williams, 1992)

$$\hat{g}_{\text{REINFORCE}}[f] = f\,(b)\,\frac{\partial}{\partial\theta}\log p(b|\theta), \qquad b \sim p(b|\theta)$$

- Unbiased
- Works for any f, not differentiable or even unknown

- high variance

$$\hat{g}_{\text{REINFORCE}}[f] = f\,(b)\,\frac{\partial}{\partial\theta}\log p(b|\theta), \qquad b \sim p(b|\theta)$$



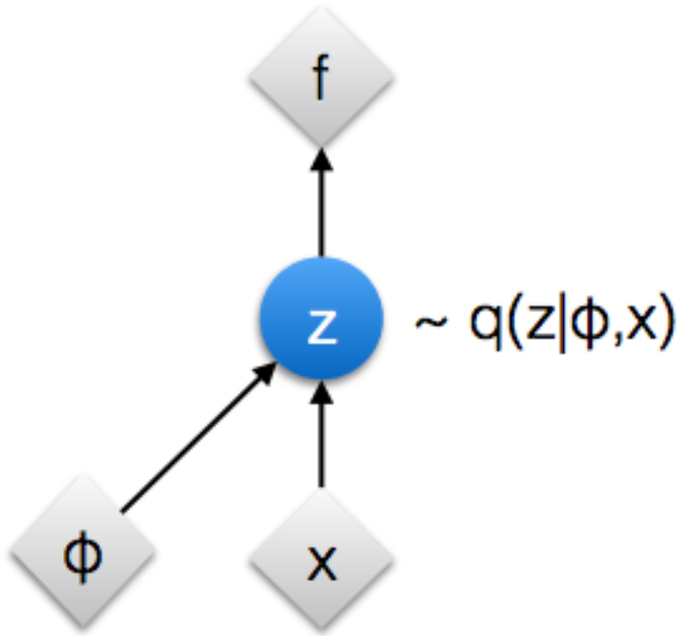
Reparameterization Trick:

$$\hat{g}_{\text{reparam}}[f] = \frac{\partial f}{\partial b}\frac{\partial b}{\partial \theta} \qquad b = T(\theta, \epsilon), \epsilon \sim p(\epsilon)$$

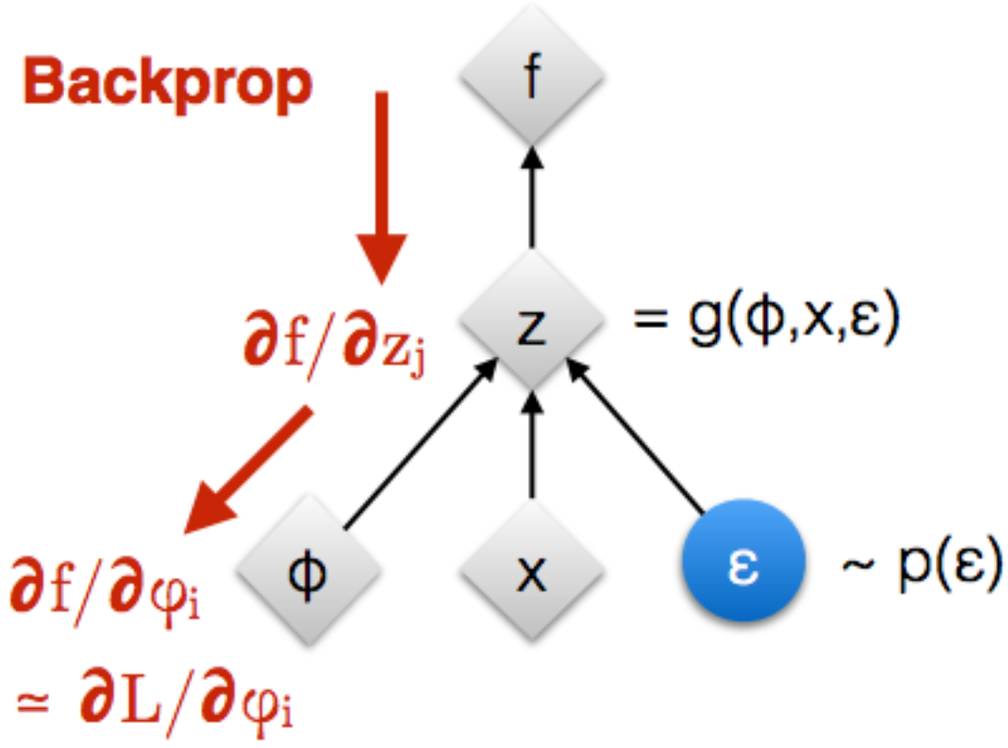
- Usually lower variance
- Unbiased
- Gold standard, allowed huge continuous models

- Requires $f(b)$ to be known and differentiable
- Requires $p(b|\theta)$ to be differentiable

Original form



Reparameterised form



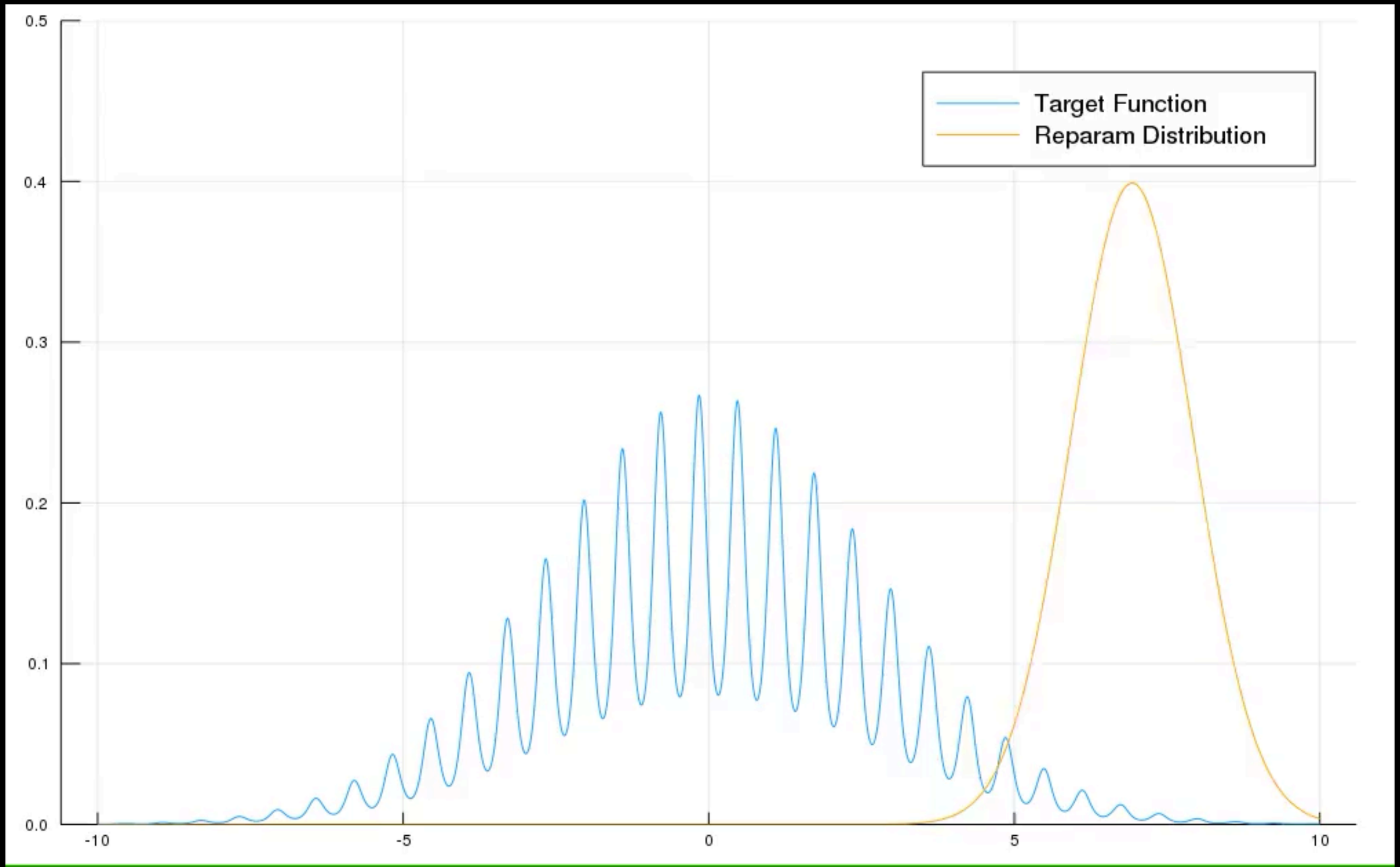
: Deterministic node

: Random node

[Kingma, 2013]
[Bengio, 2013]
[Kingma and Welling 2014]
[Rezende et al 2014]

Source: Kingma's NIPS 2015 workshop slides

$$\hat{g}_{\text{reparam}}[f] = \frac{\partial f}{\partial b}\frac{\partial b}{\partial \theta} \qquad b = T(\theta, \epsilon), \epsilon \sim p(\epsilon)$$



Concrete/Gumbel-softmax

$$\hat{g}_{\text{concrete}}[f] = \frac{\partial f}{\partial \sigma(z/t)} \frac{\partial \sigma(z/t)}{\partial \theta} \qquad z = T(\theta, \epsilon), \epsilon \sim p(\epsilon)$$

- Tune variance vs bias
- Works well in practice for discrete models

- Biased
- $f(b)$ must be known and differentiable
- $p(z|\theta)$ must be differentiable
- Uses undefined behavior of $f(b)$

Control Variates

- Allow us to reduce variance of a Monte Carlo estimator

$$\hat{g}_{\text{new}}(b) = \hat{g}(b) - c(b) + \mathbb{E}_{p(b)}[c(b)]$$

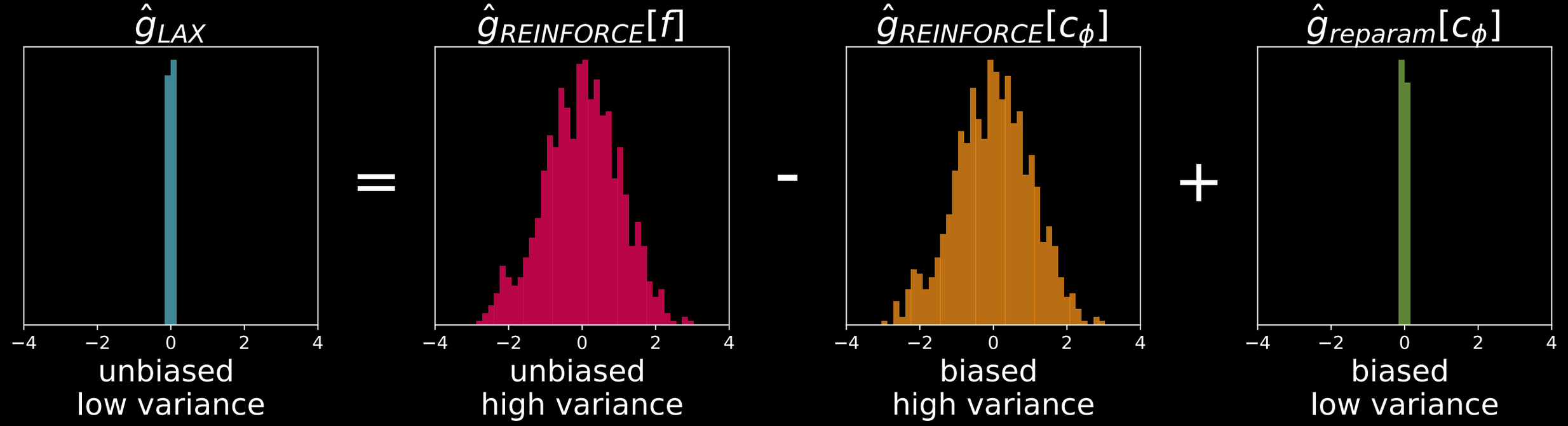
- Variance is reduced if $\text{corr}(g, c) > 0$
- Need to adapt g as problem changes during optimization

Our Approach

$$\hat{g}_{\text{LAX}} = g_{\text{REINFORCE}}[f] - g_{\text{REINFORCE}}[c_\phi] + g_{\text{reparam}}[c_\phi]$$
$$= [f(b) - c_\phi(b)] \frac{\partial}{\partial \theta} \log p(b|\theta) + \frac{\partial}{\partial \theta} c_\phi(b)$$

Our Approach

$$\hat{g}_{\text{LAX}} = g_{\text{REINFORCE}}[f] - g_{\text{REINFORCE}}[c_\phi] + g_{\text{reparam}}[c_\phi]$$



Optimizing the Control Variate

$$\frac{\partial}{\partial \phi} \text{Variance}(\hat{g}) = \mathbb{E}\left[\frac{\partial}{\partial \phi} \hat{g}^2\right]$$

- For any unbiased estimator we can get Monte Carlo estimates for the gradient of the variance of $\hat{g}$
- Use to optimize c_ϕ
- Got trick from Ruiz et al. and REBAR paper

A self-tuning gradient estimator

- Jointly optimize original problem and surrogate together with stochastic optimization
- Requires higher-order derivatives

Algorithm 1 LAX: Optimizing parameters and a gradient control variate simultaneously.

Require: $f(\cdot)$, $\log p(b|\theta)$, reparameterized sampler $b = T(\theta, \epsilon)$, neural network $c_\phi(\cdot)$, step sizes α_1, α_2

while not converged **do**

$\epsilon \sim p(\epsilon)$ ▷ Sample noise

$b \leftarrow T(\epsilon, \theta)$ ▷ Compute input

$\hat{g}_\theta \leftarrow [f(b) - c_\phi(b)] \nabla_\theta \log p(b|\theta) + \nabla_\theta c_\phi(b)$ ▷ Estimate gradient of objective

$\hat{g}_\phi \leftarrow \partial \hat{g}_\theta^2 / \partial \phi$ ▷ Estimate gradient of variance of gradient

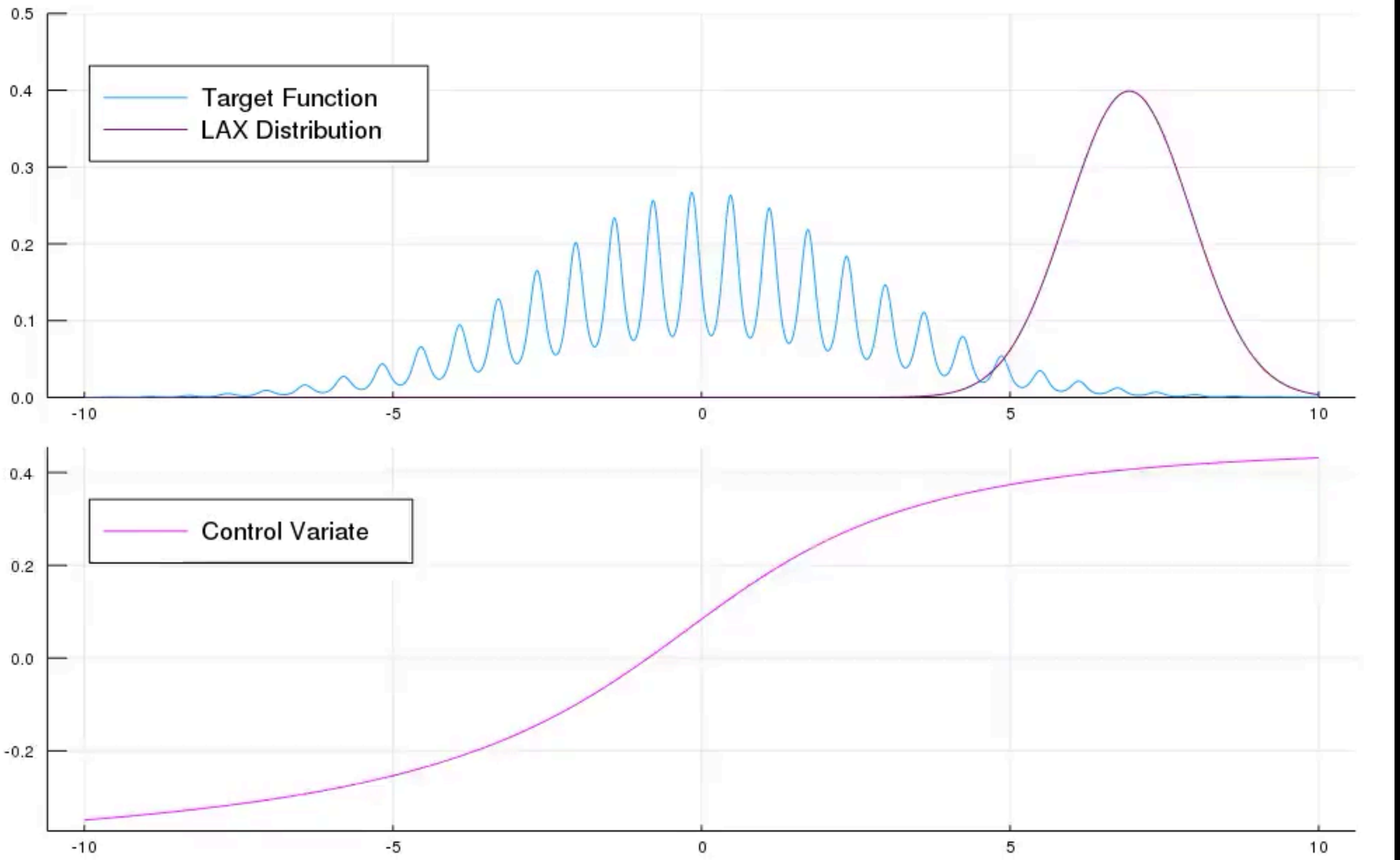
$\theta \leftarrow \theta - \alpha_1 \hat{g}_\theta$ ▷ Update parameters

$\phi \leftarrow \phi - \alpha_2 \hat{g}_\phi$ ▷ Update control variate

end while

return θ

$$\hat{g}_{\text{LAX}} = [f(b) - c_\phi(b)] \frac{\partial}{\partial \theta} \log p(b|\theta) + \frac{\partial}{\partial \theta} c_\phi(b)$$



What about discrete variables?

Extension to discrete $p(b|\theta)$

$$\hat{g}_{\text{DLAX}} = f(b)\frac{\partial}{\partial\theta}\log p(b|\theta) - c_\phi(z)\frac{\partial}{\partial\theta}\log p(z|\theta) + \frac{\partial}{\partial\theta}c_\phi(z)$$

$$b = H(z), z \sim p(z|\theta)$$

$$H(z) = b \sim p(b|\theta)$$

- Unbiased for all c_ϕ

Extension to discrete $p(b|\theta)$

$$\hat{g}_{\text{RELAX}} = [f(b) - c_\phi(\tilde{z})] \frac{\partial}{\partial \theta} \log p(b|\theta) + \frac{\partial}{\partial \theta} c_\phi(z) - \frac{\partial}{\partial \theta} c_\phi(\tilde{z})$$

$$b = H(z), z \sim p(z|\theta), \tilde{z} \sim p(z|b, \theta)$$

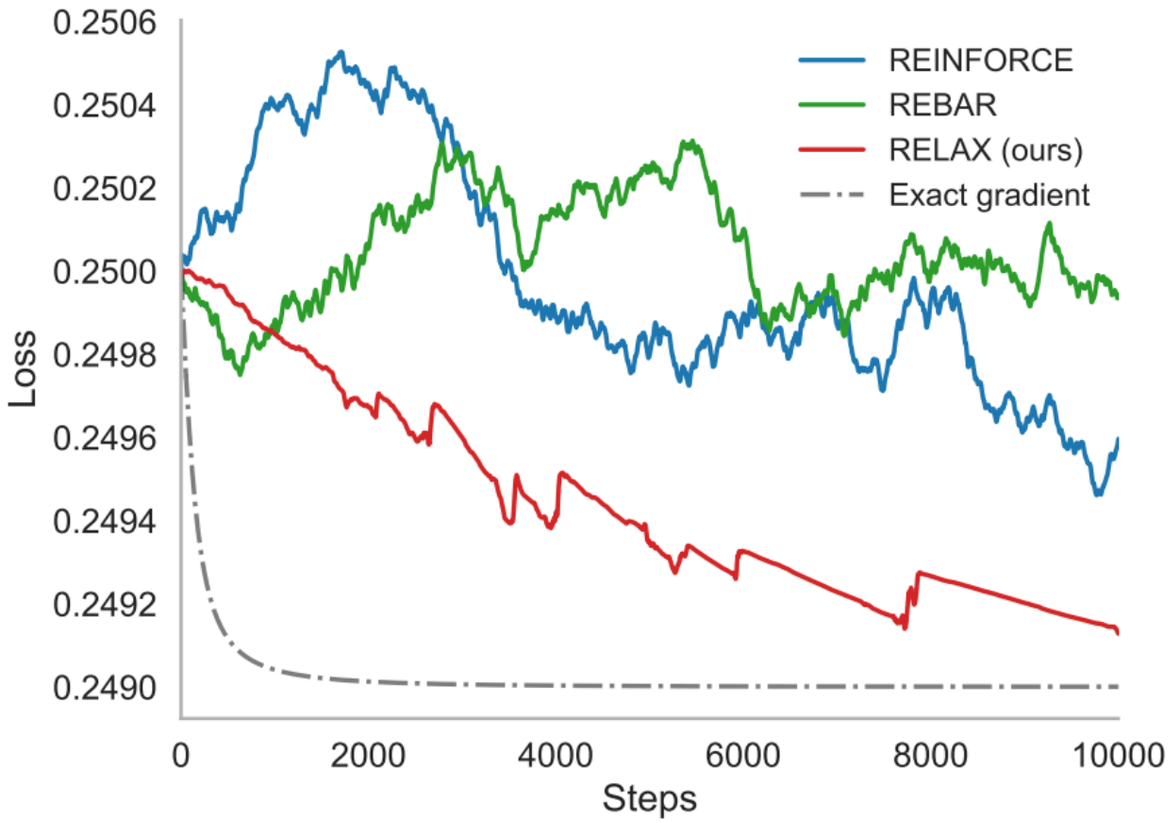
- Main trick introduced in REBAR (Tucker et al. 2017).
- We just noticed it works for any c()
- REBAR is special case of RELAX where c is concrete relaxation
- We use autodiff to tune entire surrogate, not just temperature

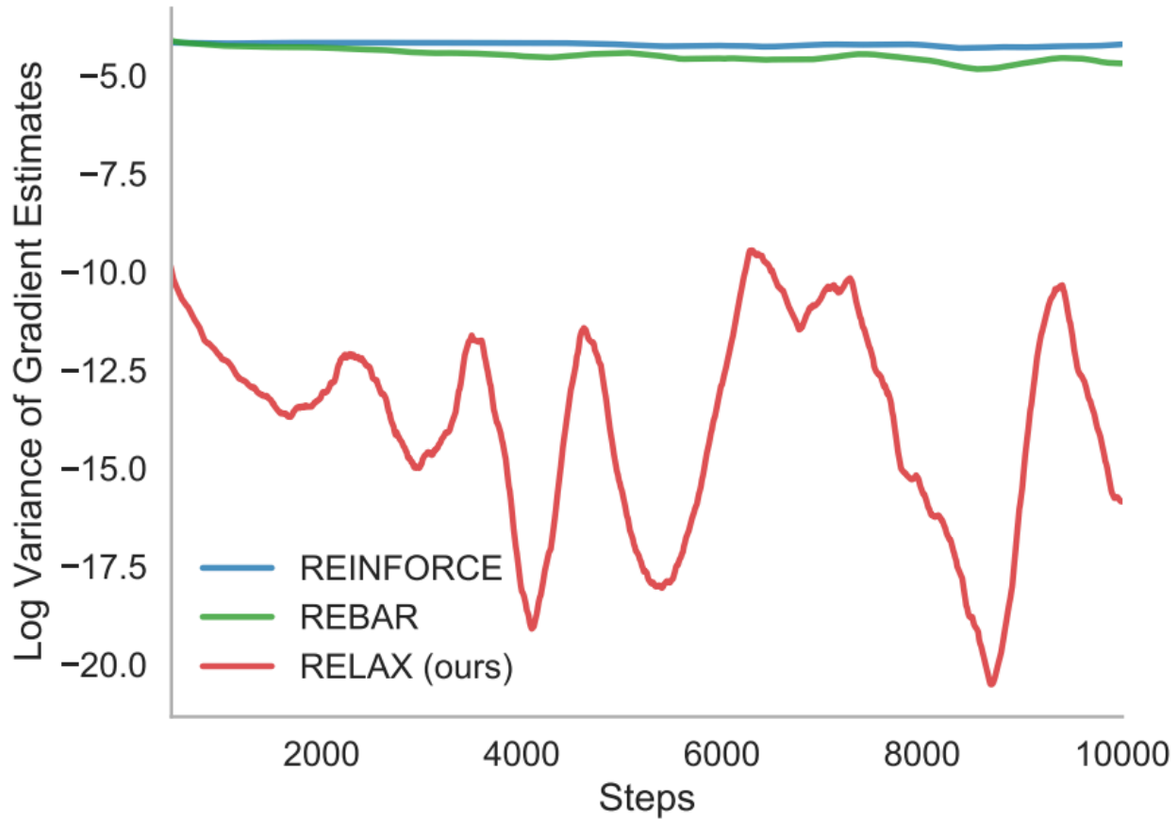
Toy Example

$$\mathbb{E}_{p(b|\theta)}[(t-b)^2]$$

- Used to validate REBAR (used t = .45)
- We use t = .499
- REBAR, REINFORCE extremely slow in this case
- Can RELAX improve?

Toy Example

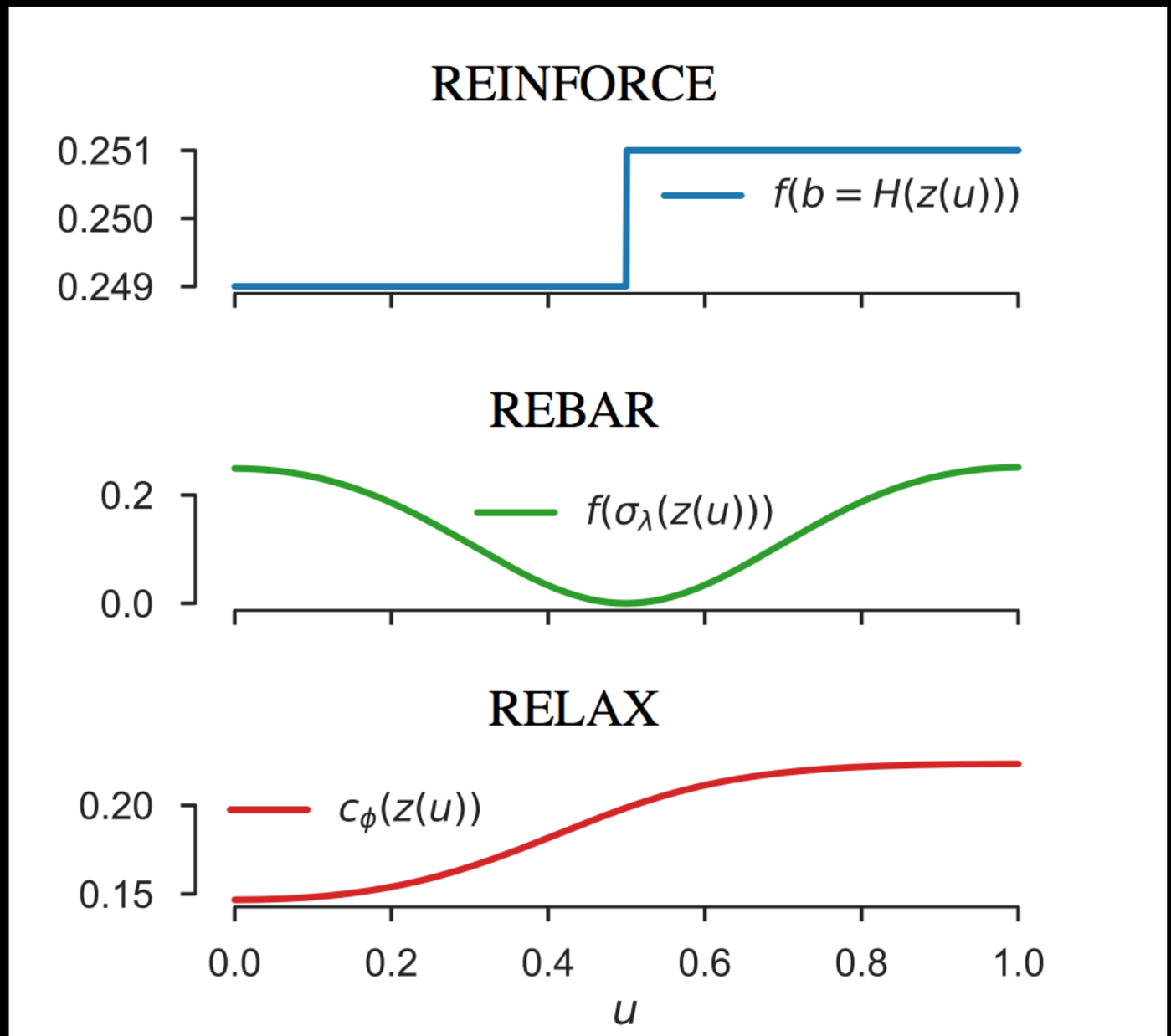




- massively reduced variance
- Surrogate needs time to catch up

Analyzing the Surrogate

- REBAR's fixed surrogate only adapts temperature param.
- RELAX surrogate balances REINFORCE variance and reparameterization variance
- Optimal surrogate is always smooth



Define functions, not computation graphs

```
def relax(params, est_params, noise_u, noise_v, func_vals):
    samples = bernoulli_sample(params, noise_u)
    log_temperature, nn_params = est_params

    def surrogate(relaxed_samples):
        return nn_predict(nn_params, relaxed_samples)

    def surrogate_cond(params):
        cond_noise = conditional_noise(params, samples, noise_v)  # z tilde
        return concrete(params, log_temperature, cond_noise, surrogate)

    grad_surrogate = elementwise_grad(concrete)(params, log_temperature, noise_u, surrogate)
    surrogate_cond, grad_surrogate_cond = value_and_grad(surrogate_cond)(params)
    return reinforce(params, noise_u, func_vals - surrogate_cond) + \
           grad_surrogate - grad_surrogate_cond

def relax_all(params, est_params, noise_u, noise_v, f):
    # Returns objective, gradients, and gradients of variance of gradients.
    func_vals = f(bernoulli_sample(params, noise_u))
    var_vjp, grads = make_vjp(relax, argnum=1)(params, est_params, noise_u, noise_v, func_vals)
    d_var_d_est = var_vjp(2 * grads / grads.shape[0])
    return func_vals, grads, d_var_d_est
```

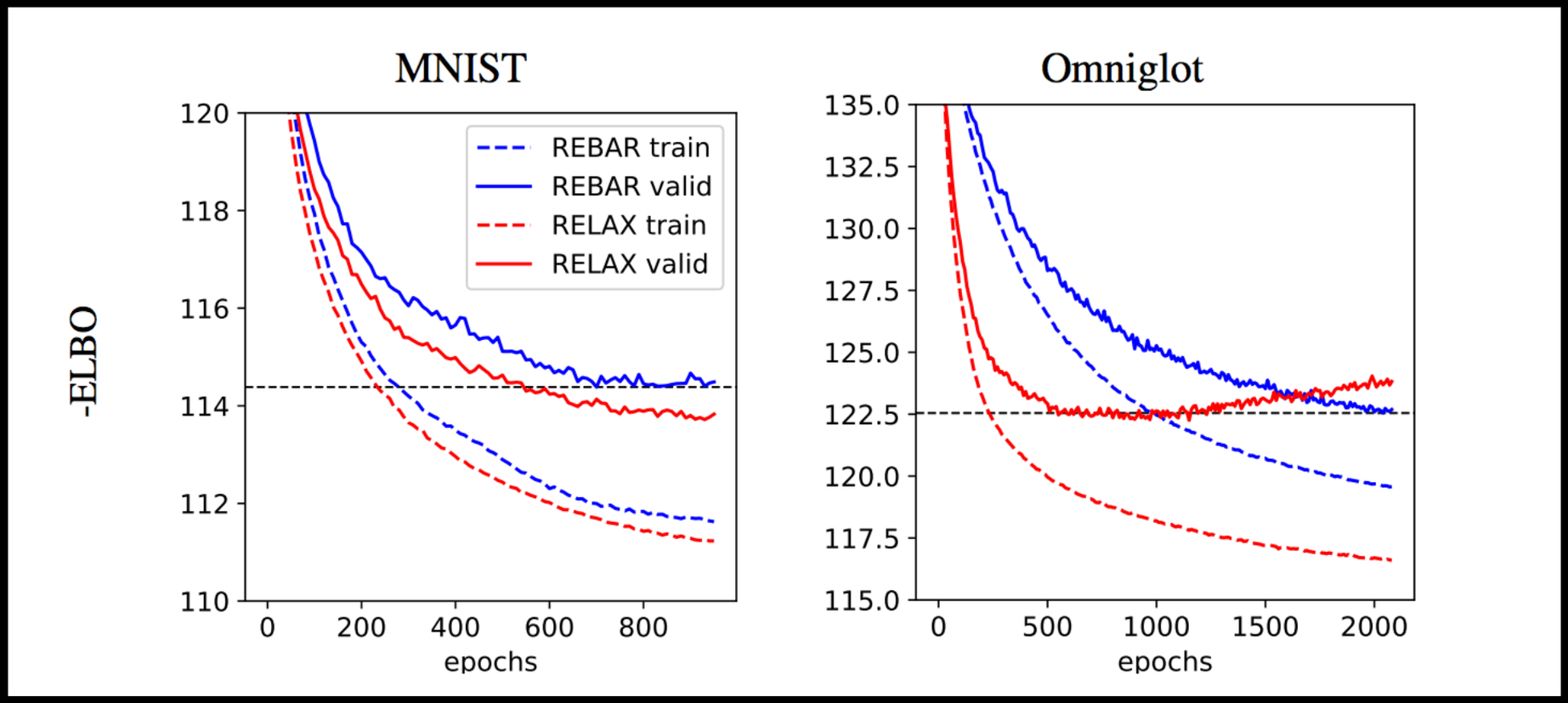
Discrete VAEs

$$\log p(x) \geq \mathcal{L}(\theta) = \mathbb{E}_{q(b|x)}[\log p(x|b) + \log p(b) - \log q(b|x)]$$

- Latent state is 200 Bernoulli variables
- Can’t use reparameterization trick
- Can still use our knowledge of structure of model, combining REBAR and RELAX:

$$c_\phi(z) = f(\sigma_\lambda(z)) + r_\rho(z)$$

Bernoulli VAE Results



Dataset	Model	Concrete	NVIL	MuProp	REBAR	RELAX
MNIST	Nonlinear	−102.2	−101.5	-101.1	-81.01	**-78.13**
	linear one-layer	-111.3	−112.5	−111.7	-111.6	**-111.20**
	linear two-layer	-99.62	−99.6	−99.07	-98.22	**-98.00**
Omniglot	Nonlinear	−110.4	−109.58	-108.72	-56.76	**-56.12**
	linear one-layer	-117.23	−117.44	−117.09	-116.63	**-116.57**
	linear two-layer	-109.95	−109.98	−109.55	-108.71	**-108.54**

Table 1: Highest training ELBO for discrete variational autoencoders.

Rederiving Actor-Critic

$$\hat{g}_{\mathrm{AC}} = \sum_{t=1}^{T} \frac{\partial \log \pi(a_t|s_t, \theta)}{\partial \theta} \left[\sum_{t'=t}^{T} r_{t'} - c_\phi(s_t) \right]$$

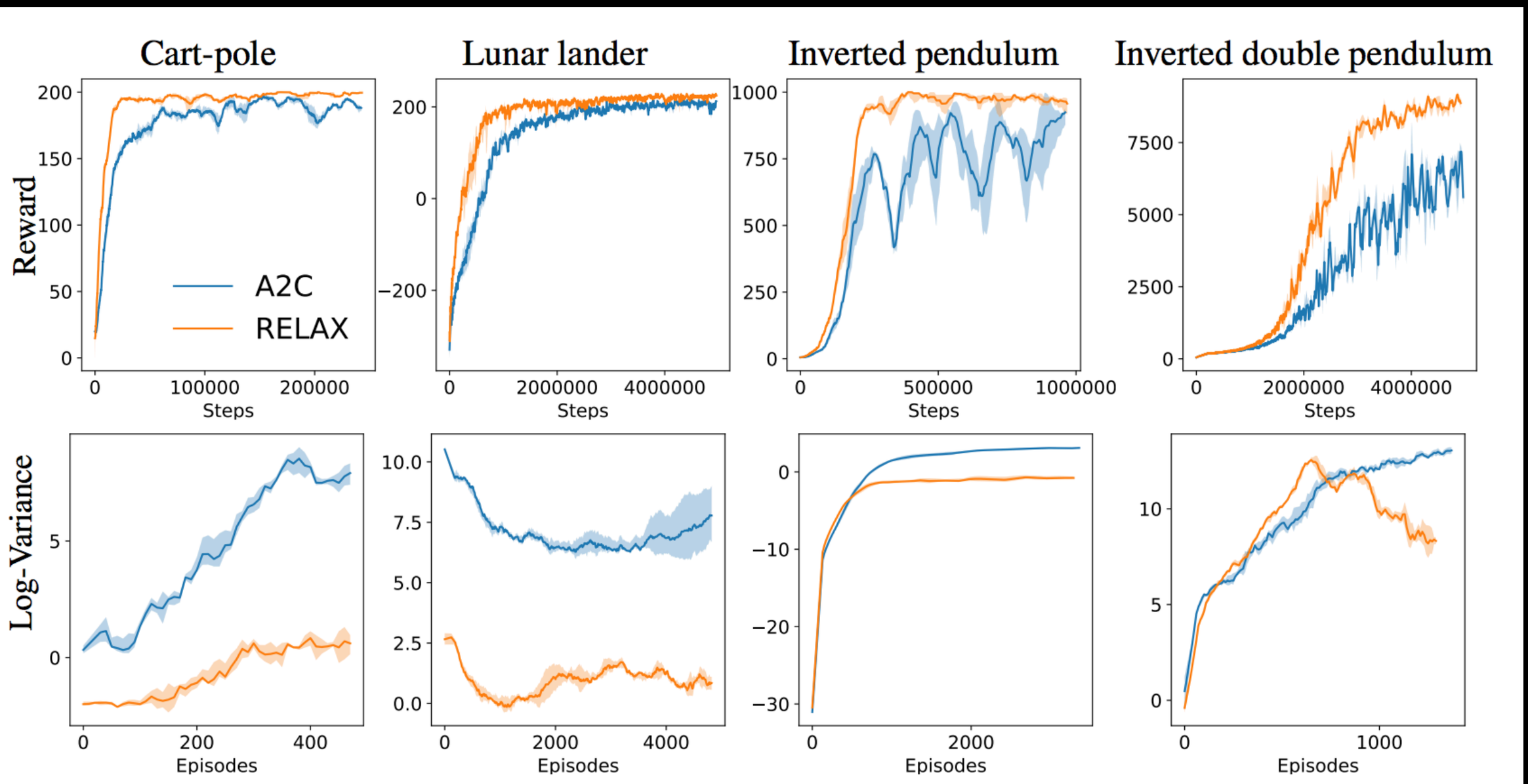
- c_ϕ is an estimate of the value function
- This is exactly the REINFORCE estimator using an estimate of the value function as a control variate
- Why not use action in control variate?
- Dependence on action would add bias

LAX for RL

$$\hat{g}_{\text{LAX}} = \sum_{t=1}^{T} \frac{\partial \log \pi(a_t|s_t,\theta)}{\partial \theta} \left[\sum_{t'=t}^{T} r_{t'} - c_\phi(s_t, a_t) \right] + \frac{\partial}{\partial \theta} c_\phi(s_t, a_t)$$

- Action-dependence in control variate
- unbiased for policy, and unbiased for baseline
- Standard baseline optimization methods minimize squared error from reward or value function. We directly minimize variance.

Model-Free RL “Results”



- Faster convergence, but real story is unbiased critic updates.

- Excellent criticism of experimental setup in “The Mirage of Action-Dependent Baselines in Reinforcement Learning” (Tucker et al. 2018). Better experiments would examine high-dimensional action regime.

RELAX Properties

- Pros:
  - unbiased
  - low variance (after tuning)
  - usable when $f(b)$ is unknown, or not differentiable
  - useable when $p(b|\theta)$ is discrete
- Cons:
  - need to define surrogate
  - when progress is made, need to wait for surrogate to adapt
  - Higher-order derivatives still awkward in TF and PyTorch

Searching through the void?

- RELAX only works well on categorical variables.
- Can’t re-use noise variables between decision branches without making true function jagged + hard to relax