Learning to Optimize

Ke Li **Jitendra Malik**

`{ke.li,malik}@eecs.berkeley.edu`

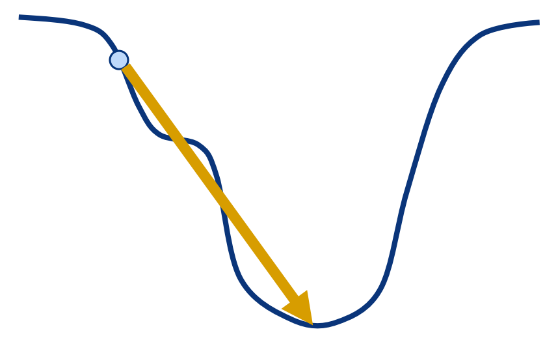
Introduction

- Optimization problems are ubiquitous in science and engineering.
- Devising a new optimization algorithm manually is challenging. Is there a better way?
- If the mantra of machine learning is to learn what is traditionally manually designed…

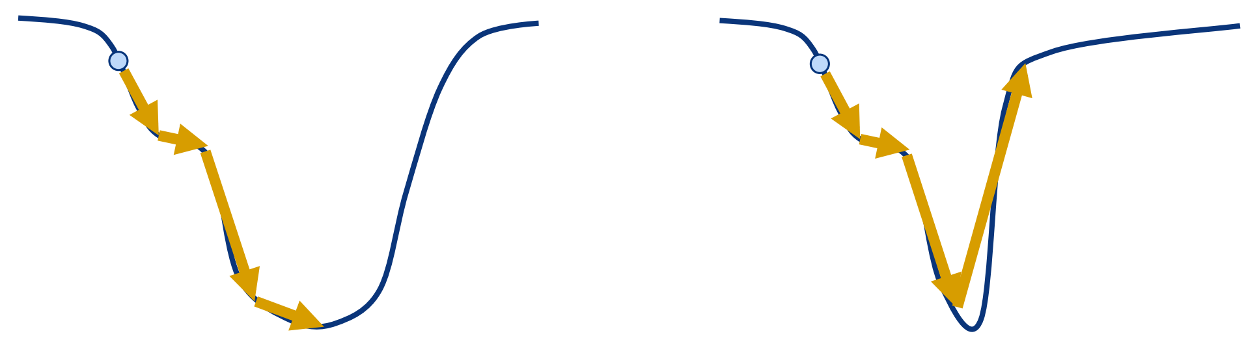
Why not *learn* the optimization algorithm itself?

Challenges

- This domain is prone to overfitting and underfitting.
- If we want to do well on a *single* objective function:
  - Consider an algorithm that memorizes the optimum.
  - This is the best optimizer since it gets to the optimum in one step.



- If we want to do well on *all* objective functions:
  - Given any optimizer, we can always construct an objective function that it performs poorly on.



- Goal: Do well on a class of objective functions with similar geometry, e.g.:
  - Logistic regression loss functions
  - Neural net classification loss functions

Setting

- Given: a set of training objective functions $f_1, \ldots, f_n \sim \mathcal{F}$, a distribution $\mathcal{D}$ for initializing the iterate and a meta-loss $\mathcal{L}(f, x^{(1)}, \ldots, x^{(T)})$ that measures the quality of the iterates $x^{(1)}, \ldots, x^{(T)}$.
- An optimization algorithm $\mathcal{A}$ takes an objective function f and an initial iterate $x^{(0)}$ as input and produces a sequence of iterates $x^{(1)}, \ldots, x^{(T)}$.
- Goal: learn $\mathcal{A}^*$ such that $\mathbb{E}_{f \sim \mathcal{F}, x^{(0)} \sim \mathcal{D}}\left[\mathcal{L}(f, \mathcal{A}^*(f, x^{(0)}))\right]$ is minimized.
- We choose $\mathcal{L}(f, x^{(1)}, \ldots, x^{(T)}) = \sum_{i=1}^{T} f(x^{(i)})$

Parameterizing Optimization Algorithms

Algorithm 1 General structure of optimization algorithms

```
Require: Objective function $f$
  $x^{(0)} \leftarrow$ random point in the domain of $f$
  for $i = 1, 2, \ldots$ do
    $\Delta x \leftarrow \phi(f, \{x^{(0)}, \ldots, x^{(i-1)}\})$
    if stopping condition is met then
      return $x^{(i-1)}$
    end if
    $x^{(i)} \leftarrow x^{(i-1)} + \Delta x$
  end for
```

Gradient Descent $\phi(\cdot) = -\gamma \nabla f(x^{(i-1)})$

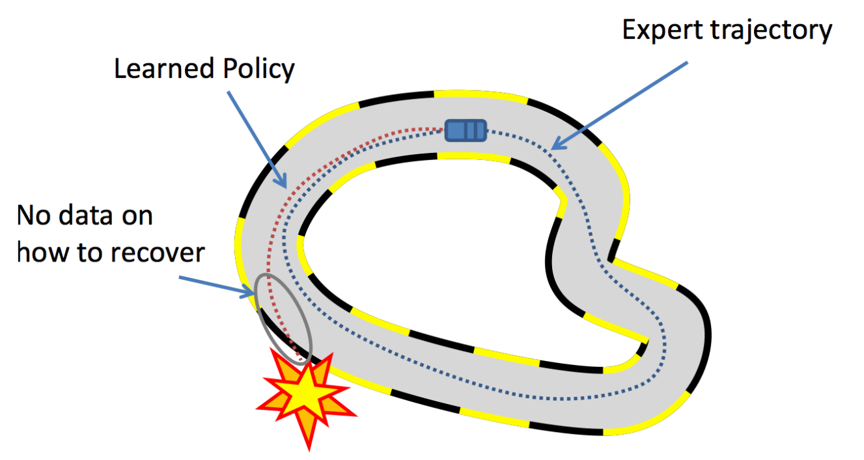
Momentum $\phi(\cdot) = -\gamma \left(\sum_{j=0}^{i-1} \alpha^{i-1-j} \nabla f(x^{(j)}) \right)$

Learned Algorithm $\phi(\cdot) =$ Neural Net

- Input: Recent history of iterates, gradients and objective values
- Output: Step vector Δx
- Searching over the space of optimization algorithms reduces to learning the parameters of the neural net.

Properties of the Learning Problem

- The prediction of the neural net at any point in time affects the inputs that it sees in the future.
- This violates the i.i.d. assumption in supervised learning.
- Compounding errors: A policy trained using supervised learning does not know how to recover from previous mistakes.



Credit: John Schulman

- A supervised learner that makes a mistake with probability ϵ incurs a cumulative error of $O(\epsilon T^2)$, rather than $O(\epsilon T)$. (Ross and Bagnell, 2010)

Reinforcement Learning

- The goal of RL is to find:

$$\pi^* = \arg\min_{\pi} \mathbb{E}_{s_0, a_0, s_1, \ldots, s_T}\left[\sum_{t=0}^{T} c(s_t)\right]$$

Time Horizon

Cost

where the expectation is taken w.r.t.

$$q\left(s_0, a_0, s_1, \ldots, s_T\right) = p_0\left(s_0\right) \prod_{t=0}^{T-1} \pi\left(a_t \middle| s_t, t\right) p\left(s_{t+1} \middle| s_t, a_t\right)$$

State

Action

Initial State Distribution

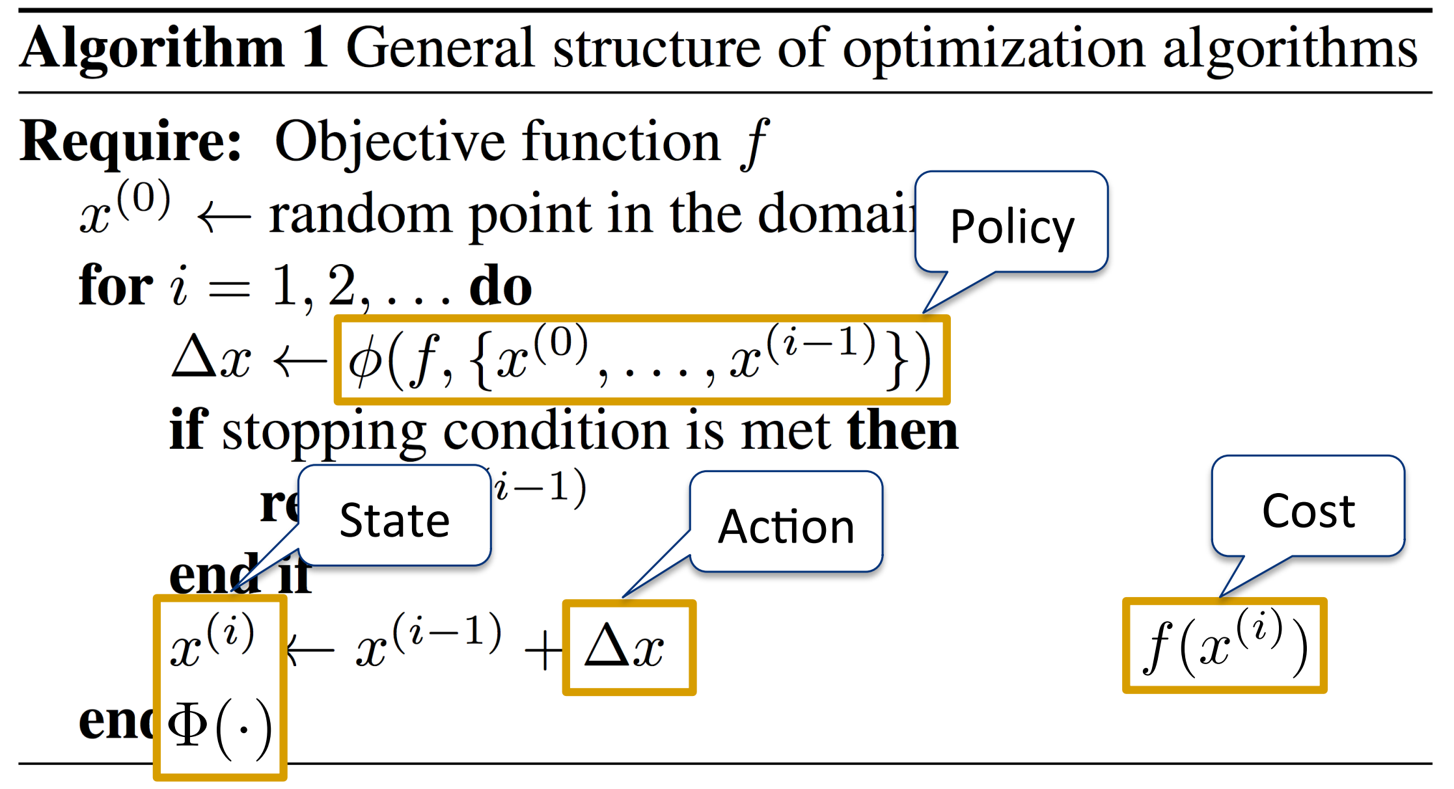
Policy

Dynamics

- The method we use is Guided Policy Search (Levine and Abbeel, 2014), which alternates between computing target trajectories and training the policy to replicate them. More precisely, it solves:

$$\min_{\theta, \eta} \mathbb{E}_{\psi}\left[\sum_{t=0}^{T} c(s_t)\right] \text{ s.t. } \psi\left(a_t \middle| s_t, t; \eta\right) = \pi\left(a_t \middle| s_t; \theta\right) \ \forall a_t, s_t, t$$

Formulation



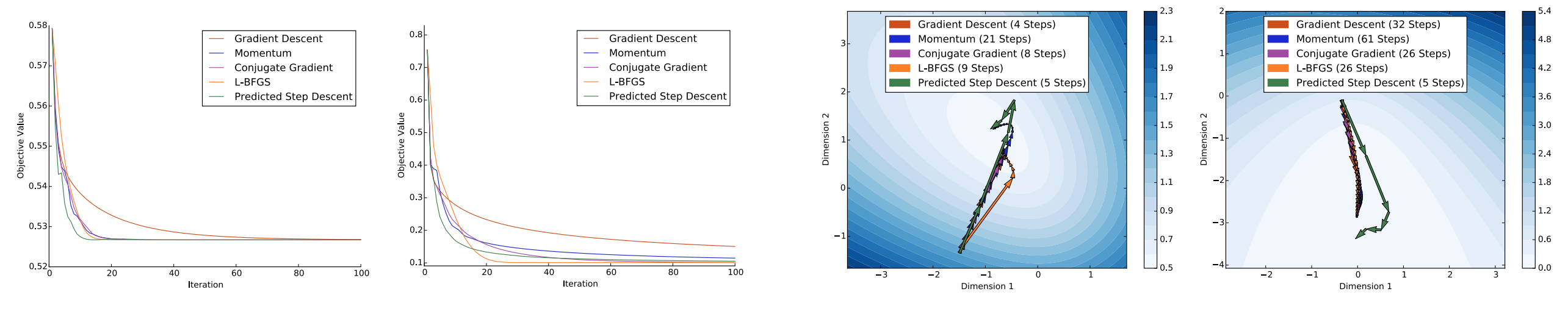
Algorithm 1 General structure of optimization algorithms

```
Require: Objective function $f$
  $x^{(0)} \leftarrow$ random point in the domai
  for $i = 1, 2, \ldots$ do
    $\Delta x \leftarrow \phi(f, \{x^{(0)}, \ldots, x^{(i-1)}\})$
    if stopping condition is met then
      re $^{i-1)}$
    end if
    $x^{(i)} \leftarrow x^{(i-1)} + \Delta x$          $f(x^{(i)})$
  end $\Phi(\cdot)$
```

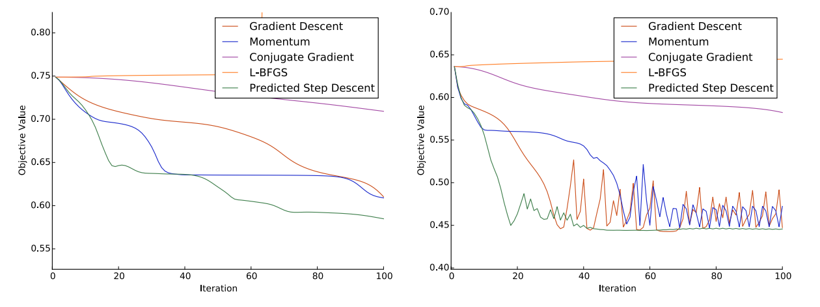
Experiments

- We trained optimizers for the following classes of low-dimensional optimization problems:
  - Logistic Regression (Convex)
  - Robust Linear Regression (Non-convex)
  - Small Neural Net Classifier (Non-convex)
- Trained on a set of random problems.
- Tested on a different set of random problems.

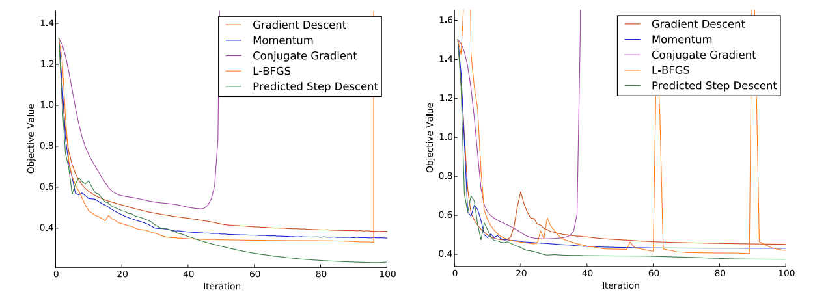
Logistic Regression:



Robust Linear Regression:



Small Neural Net:



arXiv:1606.01885

June 2016

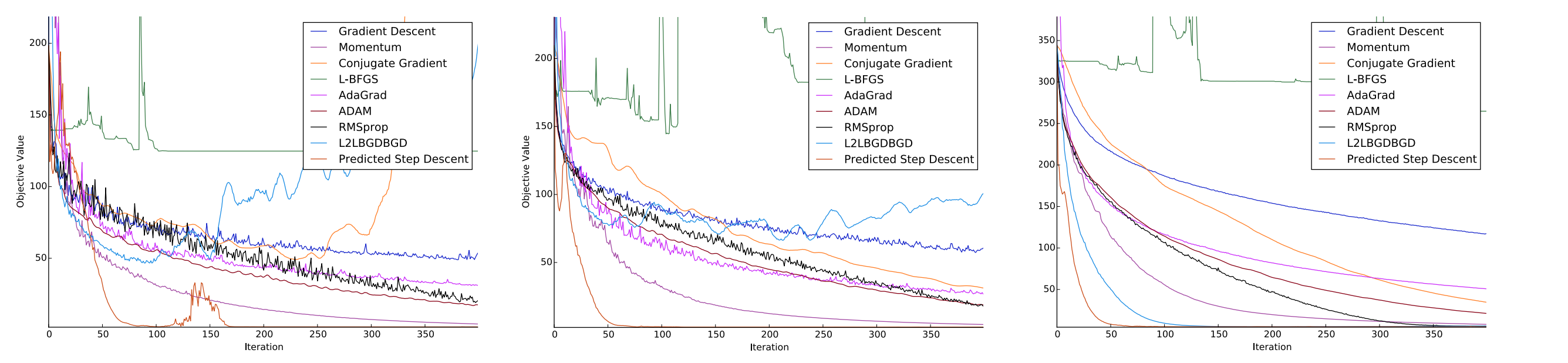
Future Work

Learning to Optimize Neural Nets

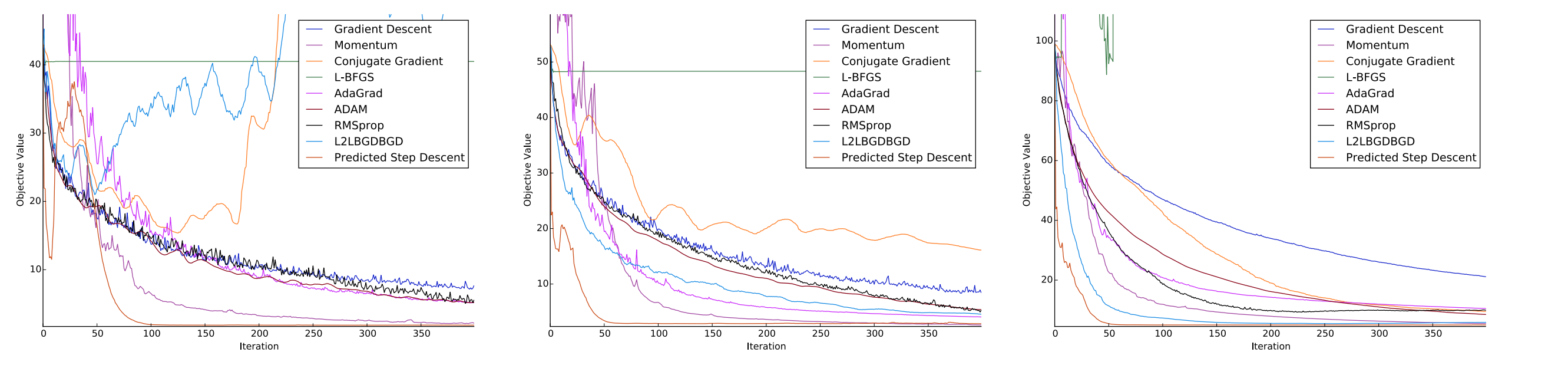
(https://arxiv.org/abs/1703.00441)

- Trained optimizer on the experience of training neural net on MNIST (a *single* objective function).
- Tested it on the problems of training a neural net on Toronto Faces, CIFAR-10 and CIFAR-100.

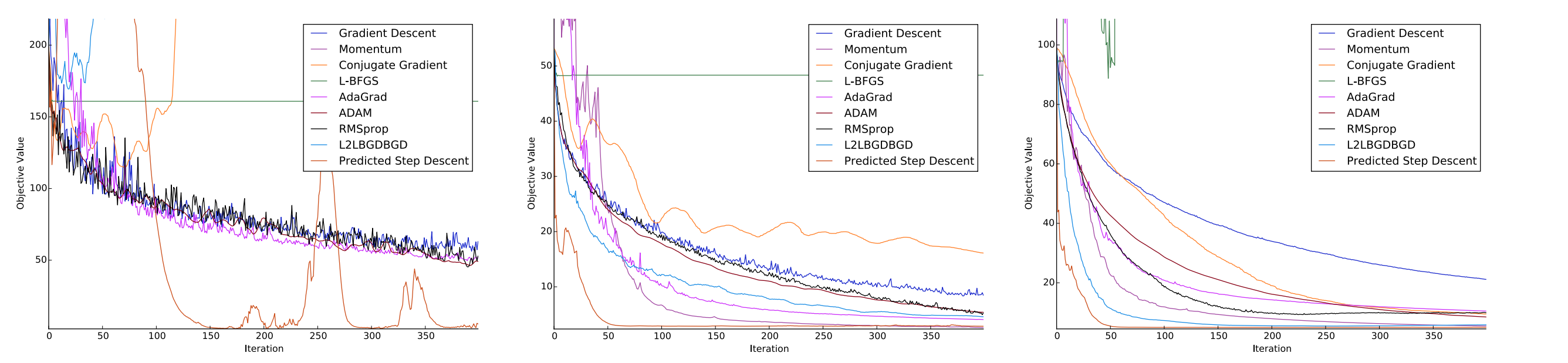
Wider Architecture:



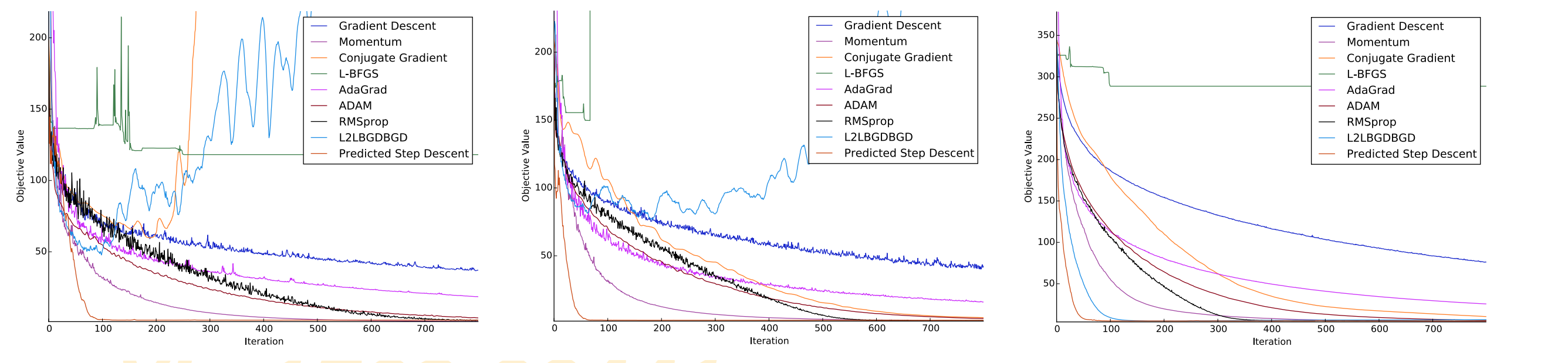
Noisier Gradients:



Wider Architecture and Noisier Gradients:



Wider Architecture and Longer Time Horizon:



arXiv:1703.00441

March 2017