Learning to Optimize

Ke Li
Jitendra Malik

Introduction

- Machine learning operates on a data-driven philosophy that favours automatic pattern discovery over manual design.
- Yet, the algorithms that power machine learning are still manually designed.
- Can we learn these algorithms instead?

Berkeley
UNIVERSITY OF CALIFORNIA

Learning to Learn

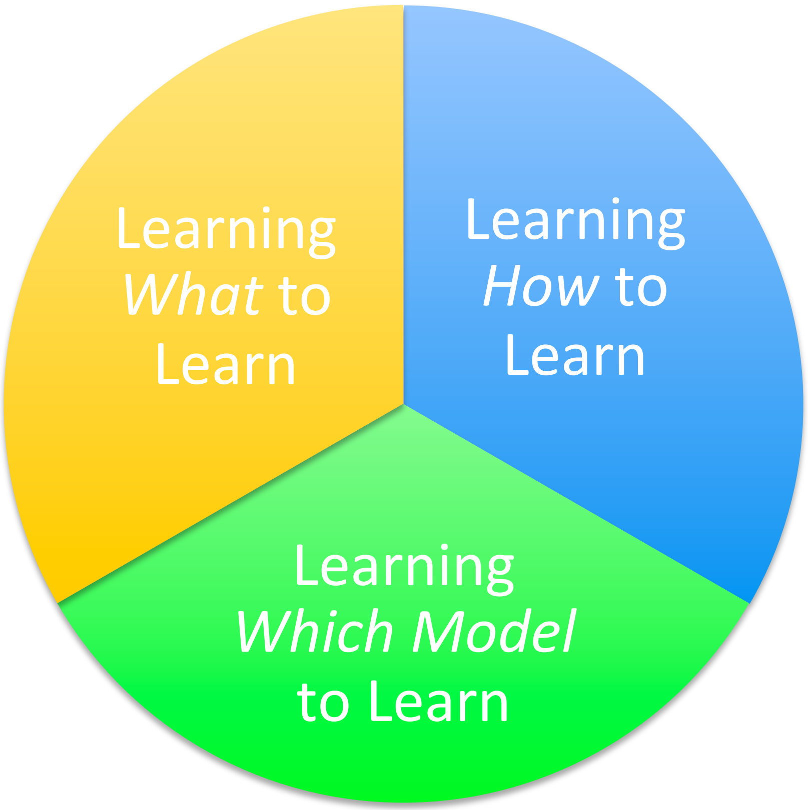
- Inspired by metacognition (Aristotle, 350 BC), which refers to the ability of humans to reason about their own process of reasoning.
- Goal: learn some general knowledge about the learning outcome or process that is useful across many tasks.
  - Unlike ordinary learning, generalization is not across instances, but across tasks.
- Terms:
  - Base-learning: instance-level learning
  - Meta-learning: task-level learning

Fundamental Challenge

- Key Problem: how do we parameterize the space of all possible learning methods such that it is both:
  1) expressive, and
  2) efficiently searchable?
- Two Extremes:
  – Enumerate a small set of methods: not expressive.
  – Search over all general-purpose programs: takes exponential time.

Learning to Learn

Different methods differ in the type of meta-knowledge they learn.

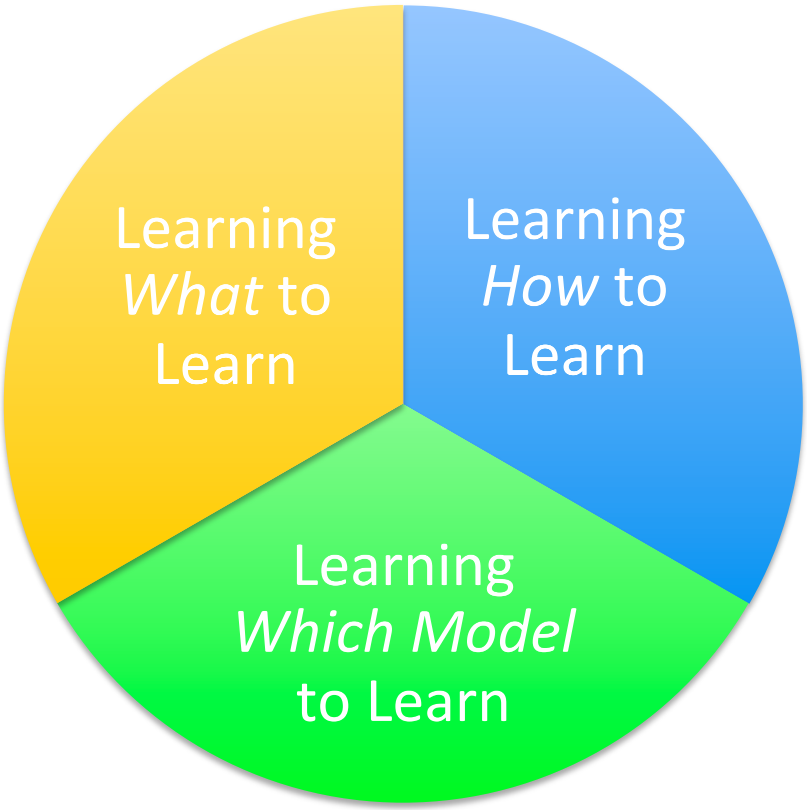


Berkeley
UNIVERSITY OF CALIFORNIA

Learning to Learn

Different methods differ in the type of meta-knowledge they learn.

Learn parameter values of the base-model that are useful across tasks.



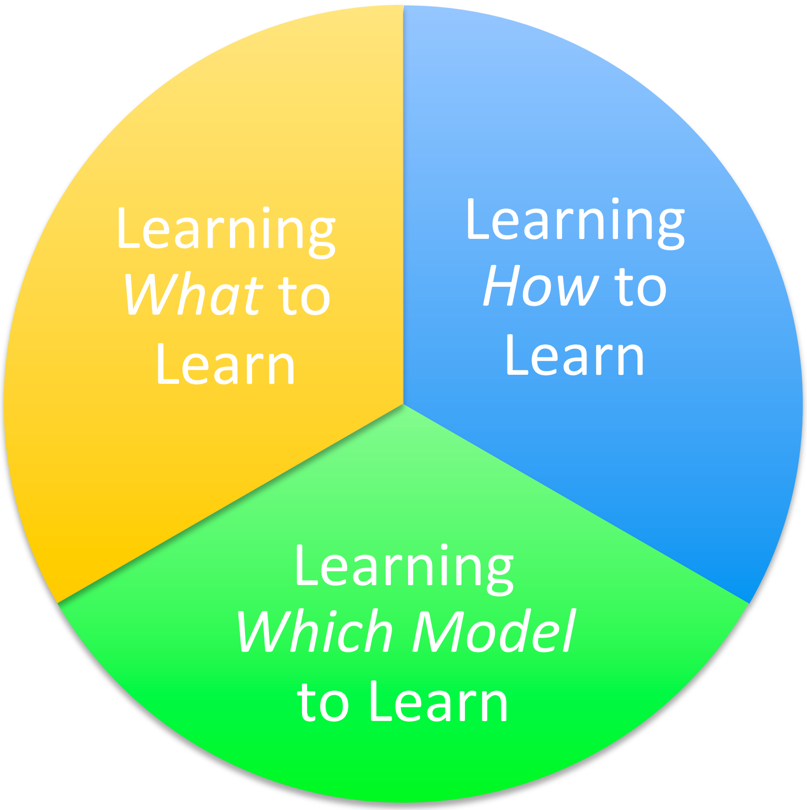
Berkeley
UNIVERSITY OF CALIFORNIA

Learning to Learn

Different methods differ in the type of meta-knowledge they learn.

Learn parameter values of the base-model that are useful across tasks.

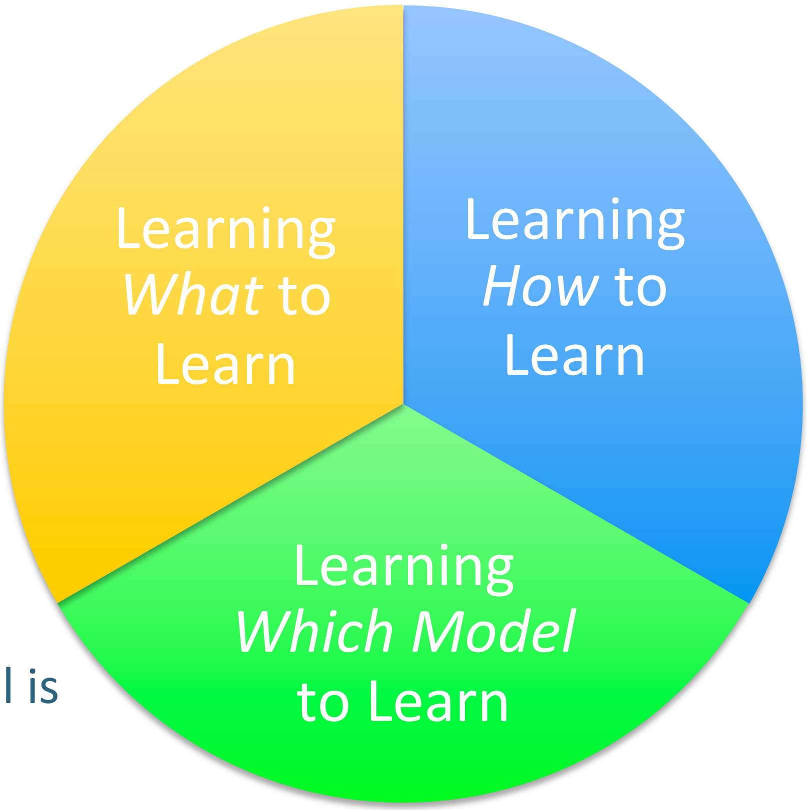
- Transfer Learning
- Multi-Task Learning
- Few-Shot Learning



Learning to Learn

Different methods differ in the type of meta-knowledge they learn.

Learn parameter values of the base-model that are useful across tasks.
- Transfer Learning
- Multi-Task Learning
- Few-Shot Learning



Learn which base-model is best suited for a task.

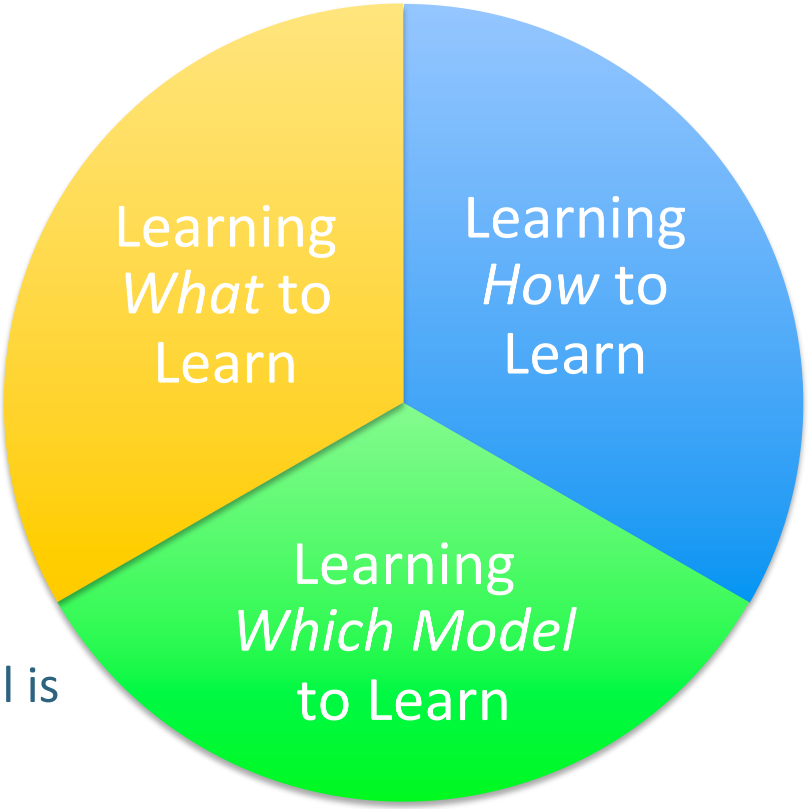
Berkeley UNIVERSITY OF CALIFORNIA

Learning to Learn

Different methods differ in the type of meta-knowledge they learn.

Learn parameter values of the base-model that are useful across tasks.

- Transfer Learning
- Multi-Task Learning
- Few-Shot Learning



Learn which base-model is best suited for a task.

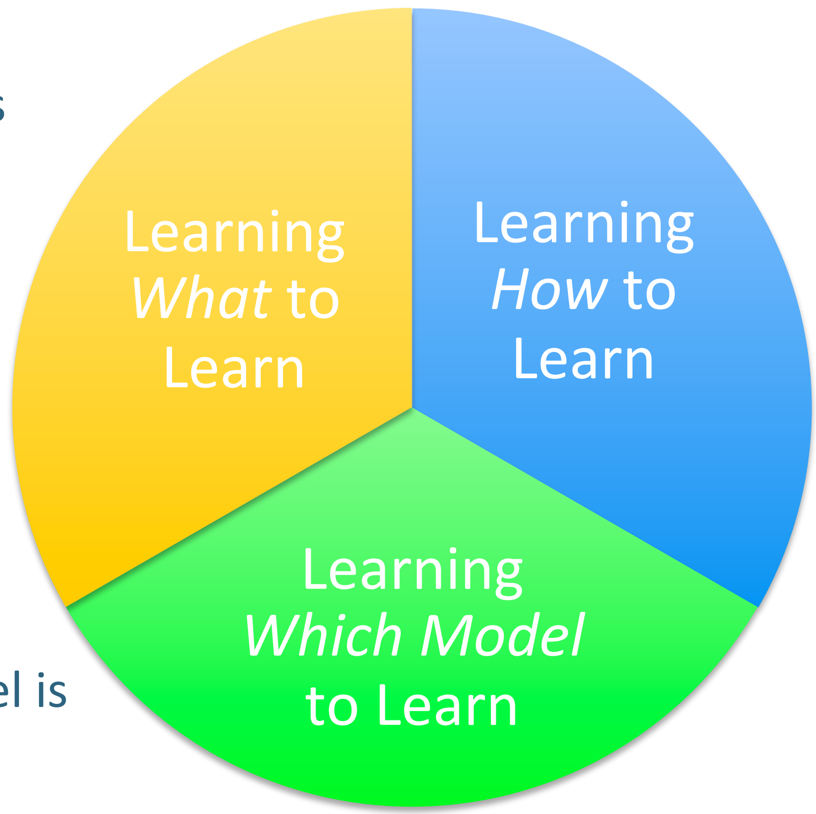
- Hyperparameter Optimization

Learning to Learn

Different methods differ in the type of meta-knowledge they learn.

Learn parameter values of the base-model that are useful across tasks.

- Transfer Learning
- Multi-Task Learning
- Few-Shot Learning



Our Contribution: Learn how to train the base-model.

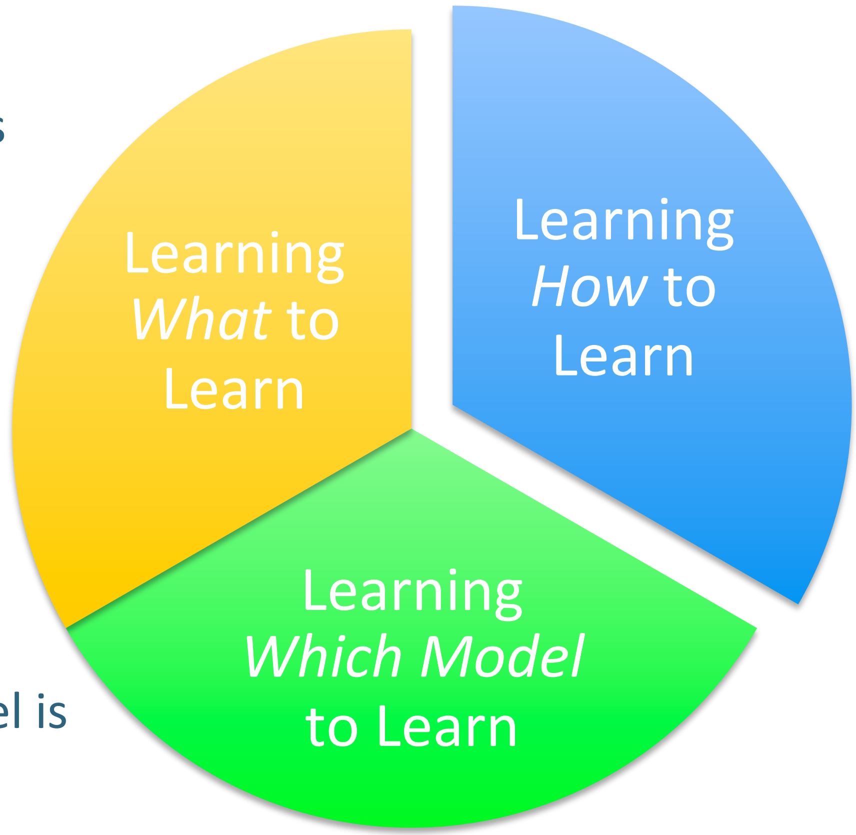
Learn which base-model is best suited for a task.

- Hyperparameter Optimization

Learning to Learn

Different methods differ in the type of meta-knowledge they learn.

Learn parameter values of the base-model that are useful across tasks.
- Transfer Learning
- Multi-Task Learning
- Few-Shot Learning



Our Contribution:
Learn how to train the base-model.
(Focus of this talk)

Learn which base-model is best suited for a task.
- Hyperparameter Optimization

Berkeley
UNIVERSITY OF CALIFORNIA

Learning How to Learn

Setting

- Most learning algorithms optimize some objective function.
  - Learning how to learn reduces to learning an optimization algorithm.
- We train an optimization algorithm on a set of objective functions.
- The learner searches the space of possible optimization algorithms and outputs an optimization algorithm that performs well on the set of objective functions.

Berkeley
UNIVERSITY OF CALIFORNIA

Learning to Optimize

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**
    $\Delta x \leftarrow \phi(\{x^{(j)}, f(x^{(j)}), \nabla f(x^{(j)})\}_{j=0}^{i-1})$
    if stopping condition is met **then**
      return $x^{(i-1)}$
    end if
    $x^{(i)} \leftarrow x^{(i-1)} + \Delta x$
  end for

Learning to Optimize

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**

$\Delta x \leftarrow \phi(\{x^{(j)}, f(x^{(j)}), \nabla f(x^{(j)})\}_{j=0}^{i-1})$

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

Berkeley
UNIVERSITY OF CALIFORNIA

Learning to Optimize

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**

$\Delta x \leftarrow \phi(\{x^{(j)}, f(x^{(j)}), \nabla f(x^{(j)})\}_{j=0}^{i-1})$

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

Berkeley UNIVERSITY OF CALIFORNIA

Learning to Optimize

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**
  $\Delta x \leftarrow \phi(\{x^{(j)}, f(x^{(j)}), \nabla f(x^{(j)})\}_{j=0}^{i-1})$
  if stopping condition is met **then**
    return $x^{(i-1)}$
  end if
  $x^{(i)} \leftarrow x^{(i-1)} + \Delta x$
end for

How do we learn $\phi(\cdot)$?

Berkeley UNIVERSITY OF CALIFORNIA

Learning to Optimize

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**

  $\Delta x \leftarrow \phi(\{x^{(j)}, f(x^{(j)}), \nabla f(x^{(j)})\}_{j=0}^{i-1})$

  if stopping condition is met **then**

    return $x^{(i-1)}$

  end if

  $x^{(i)} \leftarrow x^{(i-1)} + \Delta x$

end for

How do we learn $\phi(\cdot)$? We use reinforcement learning.

Background on Reinforcement Learning

- Set of states: $\mathcal{S} \subseteq \mathbb{R}^D$
- Set of actions: $\mathcal{A} \subseteq \mathbb{R}^d$
- Probability density over initial states: $p_i\left(s_0\right)$
- State transition probability density: $p\left(s_{t+1} \mid s_t, a_t\right)$
- Cost function: $c : \mathcal{S} \to \mathbb{R}$
- Time horizon: T
- Typically, the reinforcement learning algorithm does not know what $p\left(s_{t+1} \mid s_t, a_t\right)$ is.

Background on Reinforcement Learning

- Set of states: $\mathcal{S} \subseteq \mathbb{R}^D$
- Set of actions: $\mathcal{A} \subseteq \mathbb{R}^d$
- Probability density over initial states: $p_i\left(s_0\right)$
- State transition probability density: $p\left(s_{t+1} \mid s_t, a_t\right)$
- Cost function: $c : \mathcal{S} \to \mathbb{R}$
- Time horizon: T
- Typically, the reinforcement learning algorithm does not know what $p\left(s_{t+1} \mid s_t, a_t\right)$ is.

This is crucial.

Berkeley
UNIVERSITY OF CALIFORNIA

Background on Reinforcement Learning

- Policy: $\pi\left(a_t \middle| s_t, t\right)$
  - When it is independent of t, it is known as stationary.
- The goal is to find:

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

where the expectation is taken w.r.t.

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

Berkeley
UNIVERSITY OF CALIFORNIA

Reduction to Reinforcement Learning

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**

$\Delta x \leftarrow \phi(\Phi(\{x^{(j)}, f(x^{(j)}), \nabla f(x^{(j)})\}_{j=0}^{i-1}))$

if stopping condition is met **then**

return $x^{(i-1)}$

end if

$x^{(i)} \leftarrow x^{(i-1)} + \Delta x$

end for

Berkeley UNIVERSITY OF CALIFORNIA

Reduction to Reinforcement Learning

Algorithm 1 General structure of optimization algorithms

Require: Objective function f

$x^{(0)} \leftarrow$ random point in the dor

for $i = 1, 2, \ldots$ **do**

$\Delta x \leftarrow \phi(\Phi(\{x^{(j)}, f(x^{(j)}), \nabla f(x^{(j)})\}_{j=0}^{i-1}))$

if stopping condition is met **then**

ret $1)$

end if

$x^{(i)} \leftarrow x^{(i-1)} + \Delta x$

en

Policy

State: $x^{(i)}$, $\Phi(\cdot)$

Action: Δx

Cost: $f(x^{(i)})$

Reduction to Reinforcement Learning

- Under this formulation, the state transition probability density $p\left(s_{t+1} \middle| s_t, a_t\right)$ captures how the gradient and objective value are likely to change for any given step vector.
  - In other words, it encodes the distribution of local geometries of the objective functions of interest.
- The geometry of an unseen objective function is unknown.
  - This is OK, since reinforcement learning does not assume knowledge of $p\left(s_{t+1} \middle| s_t, a_t\right)$.

Why Reinforcement Learning?

Berkeley
UNIVERSITY OF CALIFORNIA

Simultaneous Discovery

- A similar idea was also proposed independently by Andrychowicz et al. soon after our paper appeared:

Berkeley UNIVERSITY OF CALIFORNIA

Learning to Optimize

Ke Li **Jitendra Malik**
Department of Electrical Engineering and Computer Sciences
University of California, Berkeley

Google DeepMind

Learning to learn by gradient descent by gradient descent

Marcin Andrychowicz **Misha Denil** **Sergio Gomez**
Matthew W. Hoffman **David Pfau** **Tom Schaul**
Nando de Freitas

Simultaneous Discovery

- A similar idea was also proposed independently by Andrychowicz et al. soon after our paper appeared:

Berkeley UNIVERSITY OF CALIFORNIA

Learning to Optimize

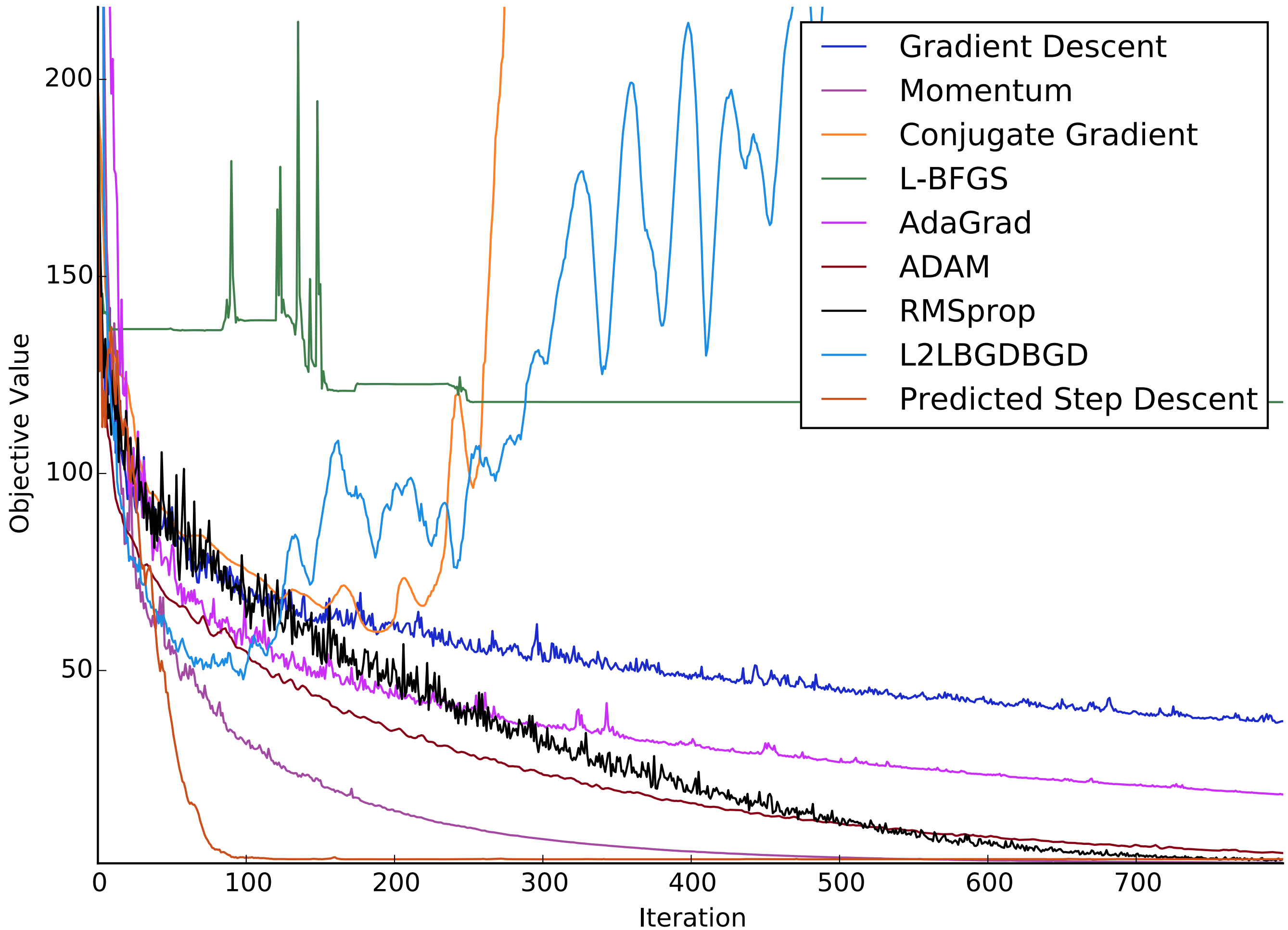
Uses Reinforcement Learning

Google DeepMind

Learning to learn by gradient descent by gradient descent

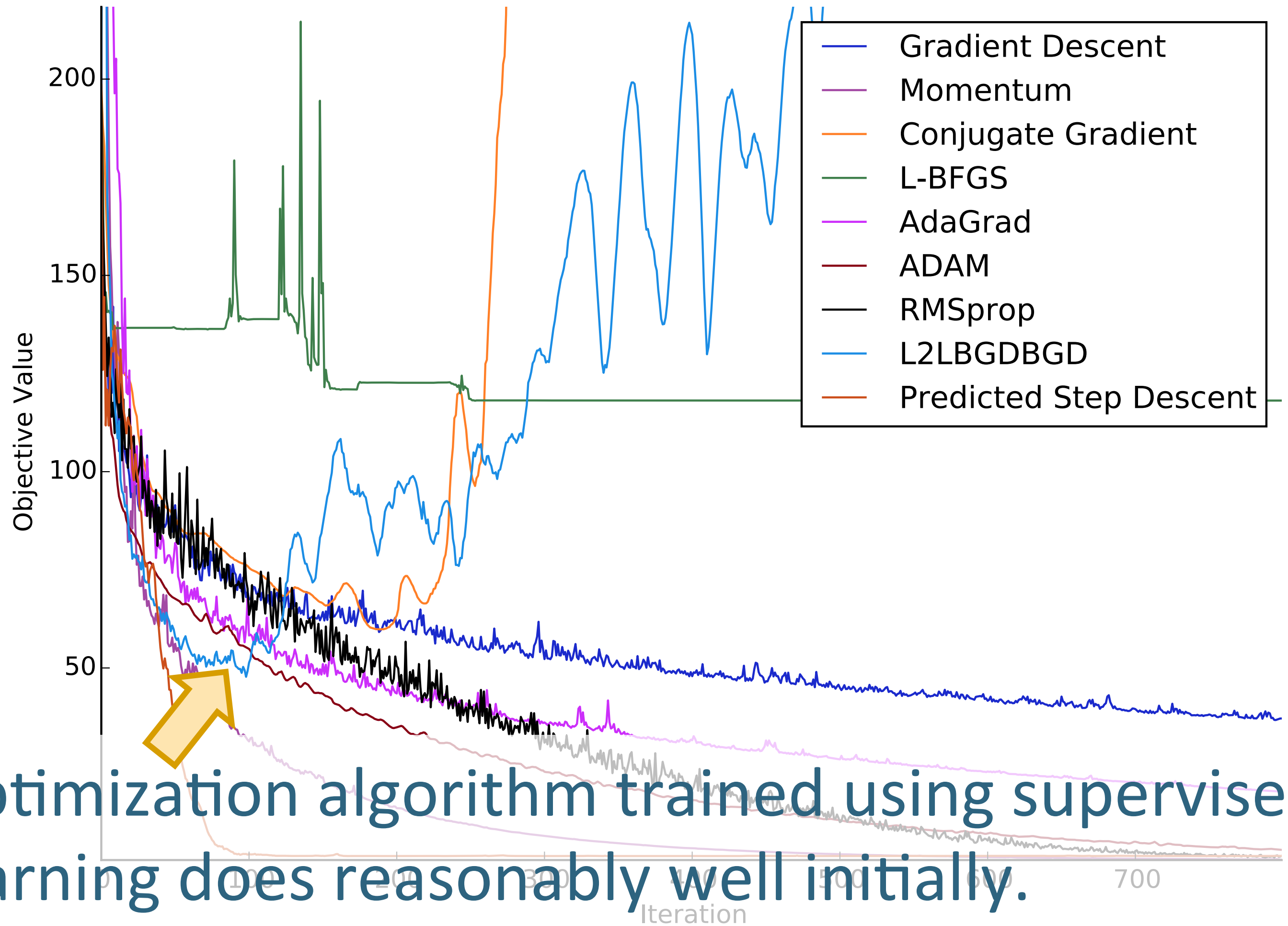
Uses Supervised Learning

Problem is Harder Than It Looks



Berkeley
UNIVERSITY OF CALIFORNIA

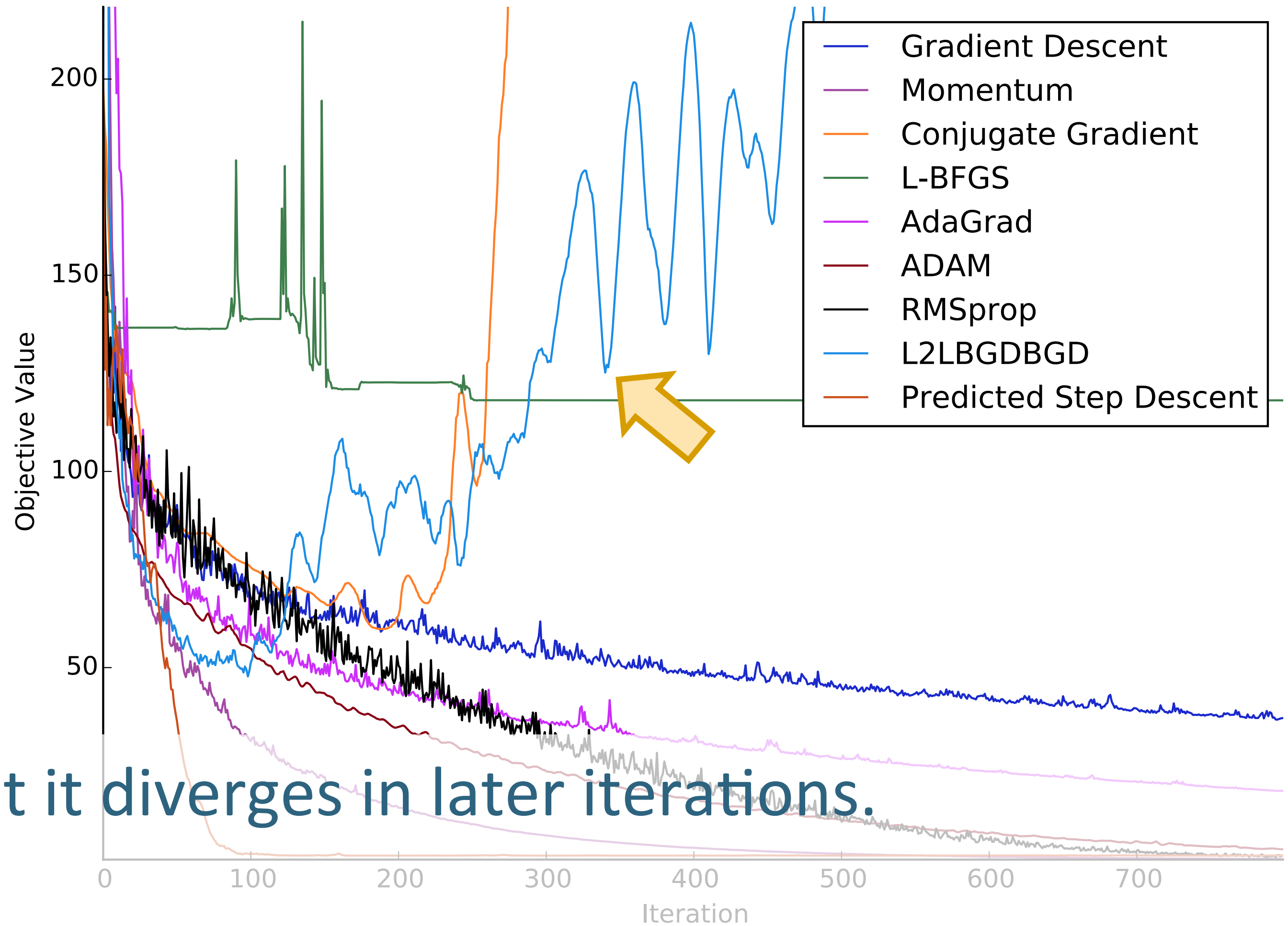
Problem is Harder Than It Looks



- Optimization algorithm trained using supervised learning does reasonably well initially.

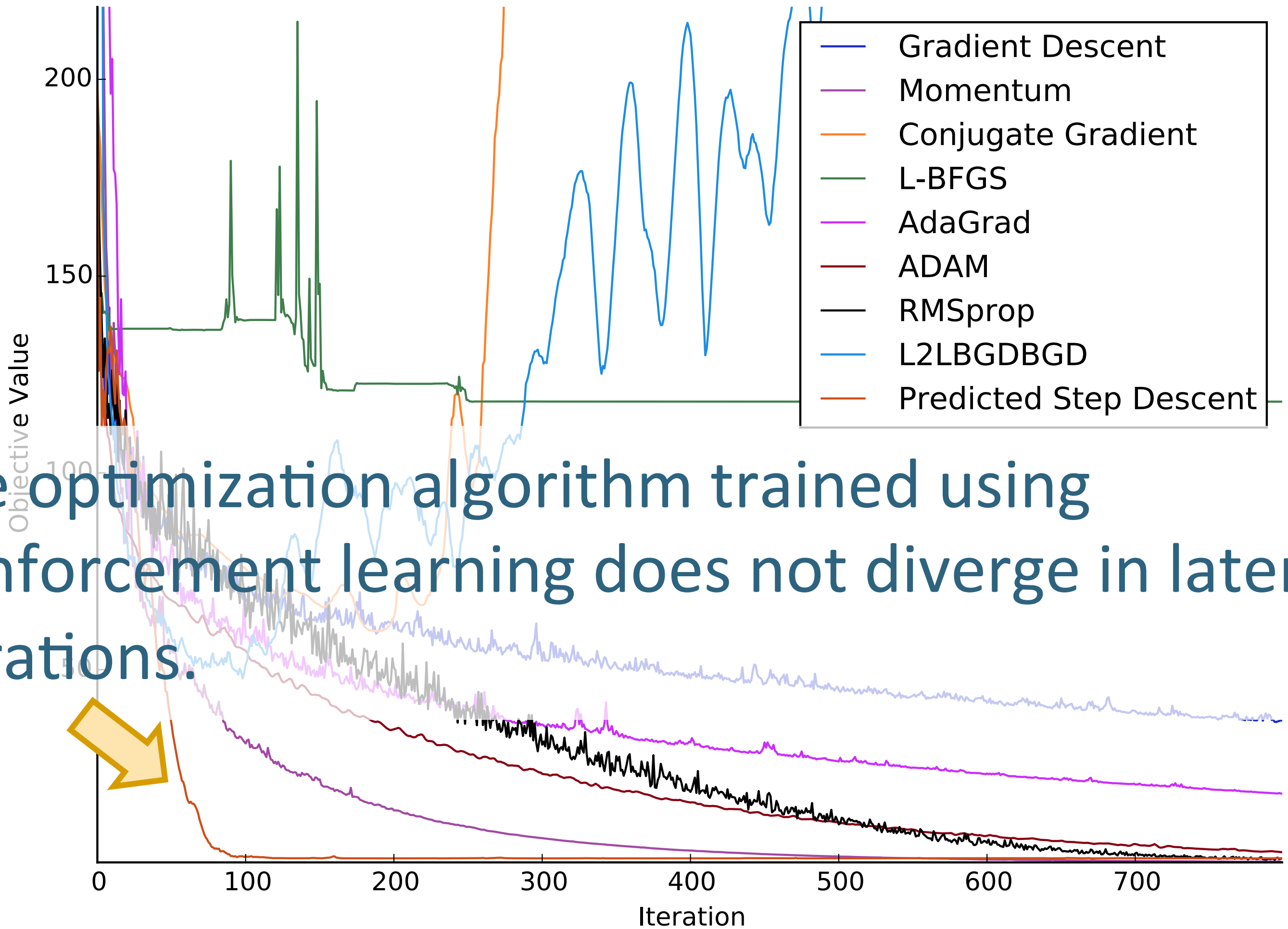
Berkeley
UNIVERSITY OF CALIFORNIA

Problem is Harder Than It Looks



- But it diverges in later iterations.

Problem is Harder Than It Looks



- The optimization algorithm trained using reinforcement learning does not diverge in later iterations.

Berkeley UNIVERSITY OF CALIFORNIA

What Generalization Means

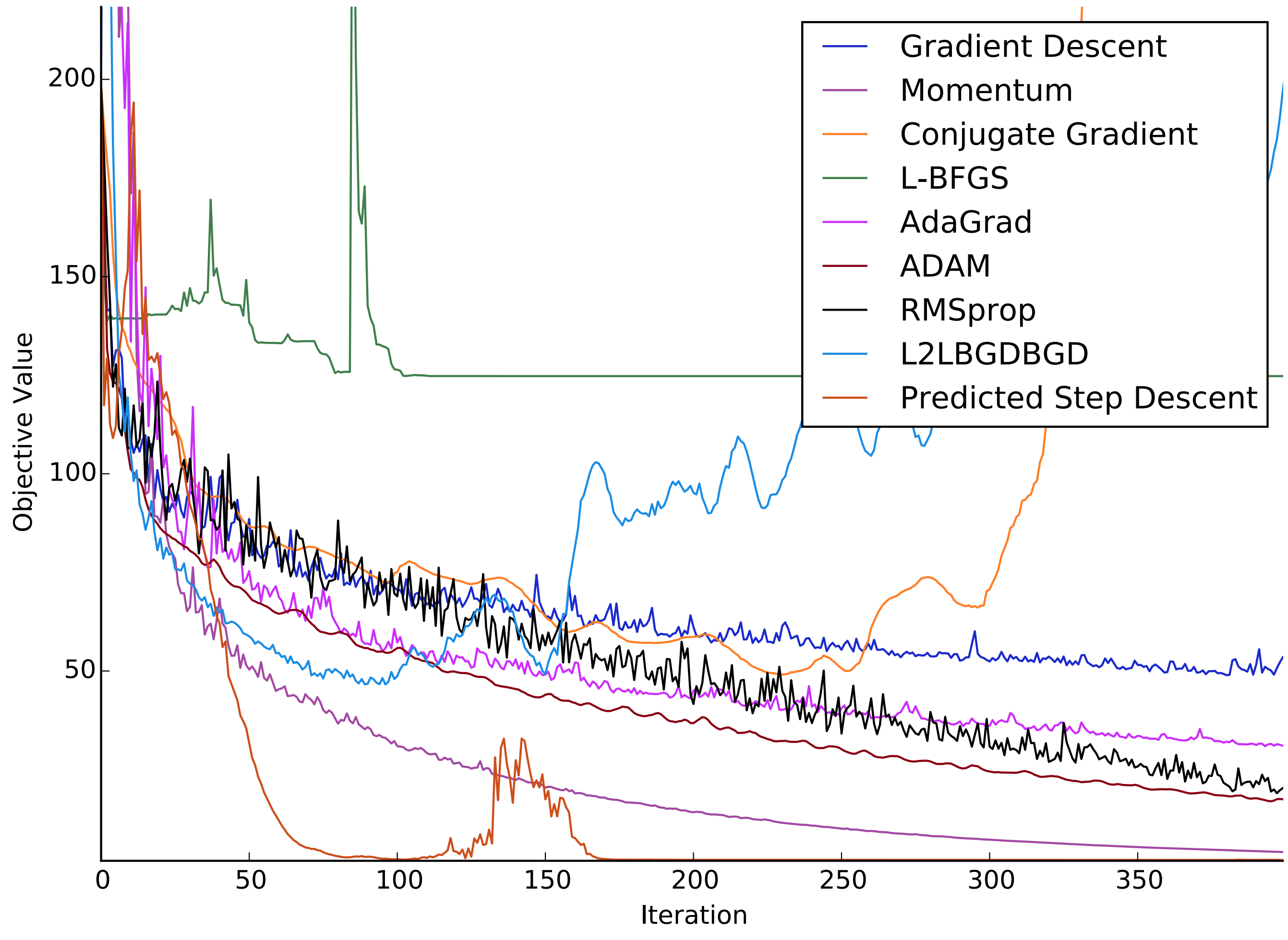
- Each example is an objective function.
  - In the learning-to-learn setting, it is the loss function for training a base-model on a task.
  - Objective functions can differ in two ways:
    - The base-model
    - The task
- Generalization is across objective functions.
  - In the learning-to-learn setting, it is across base-models and/or tasks.
- We should train the optimization algorithm on some base-models and tasks, and test it on different base-models and tasks.

Experimental Setting

- The training set consists of one objective function: the cross-entropy loss function for training a neural net on MNIST.
- The test set consists of the loss functions for training neural nets with different architectures on different datasets, i.e.: the Toronto Faces Dataset (TFD), CIFAR-10 and CIFAR-100.
- In other words, the optimization algorithm is:
  – *Meta-trained* on the problem of training a neural net on MNIST.
  – *Meta-tested* on the problems of training neural nets on TFD, CIFAR-10 and CIFAR-100.

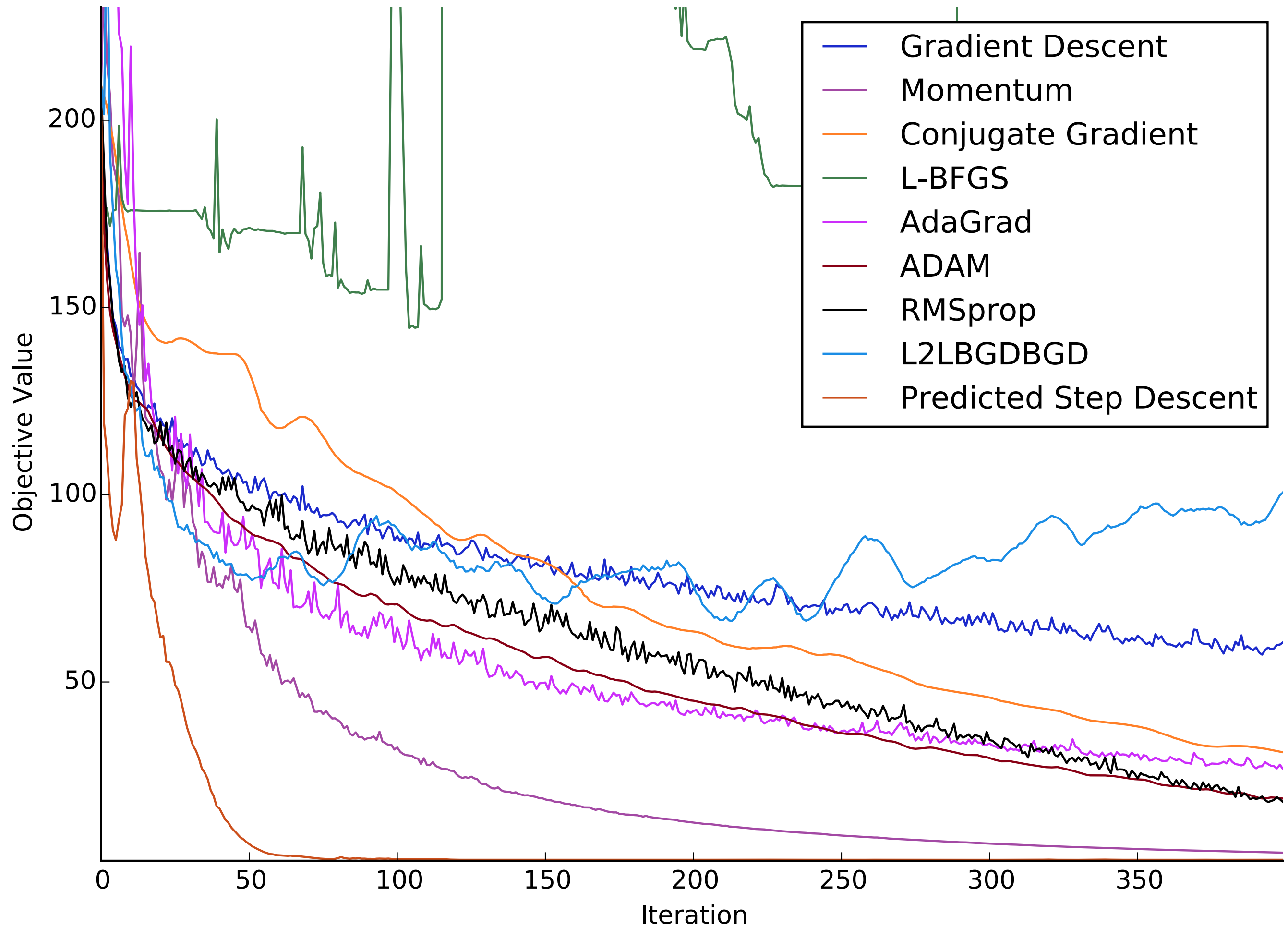
Berkeley UNIVERSITY OF CALIFORNIA

Larger Architecture (TFD)



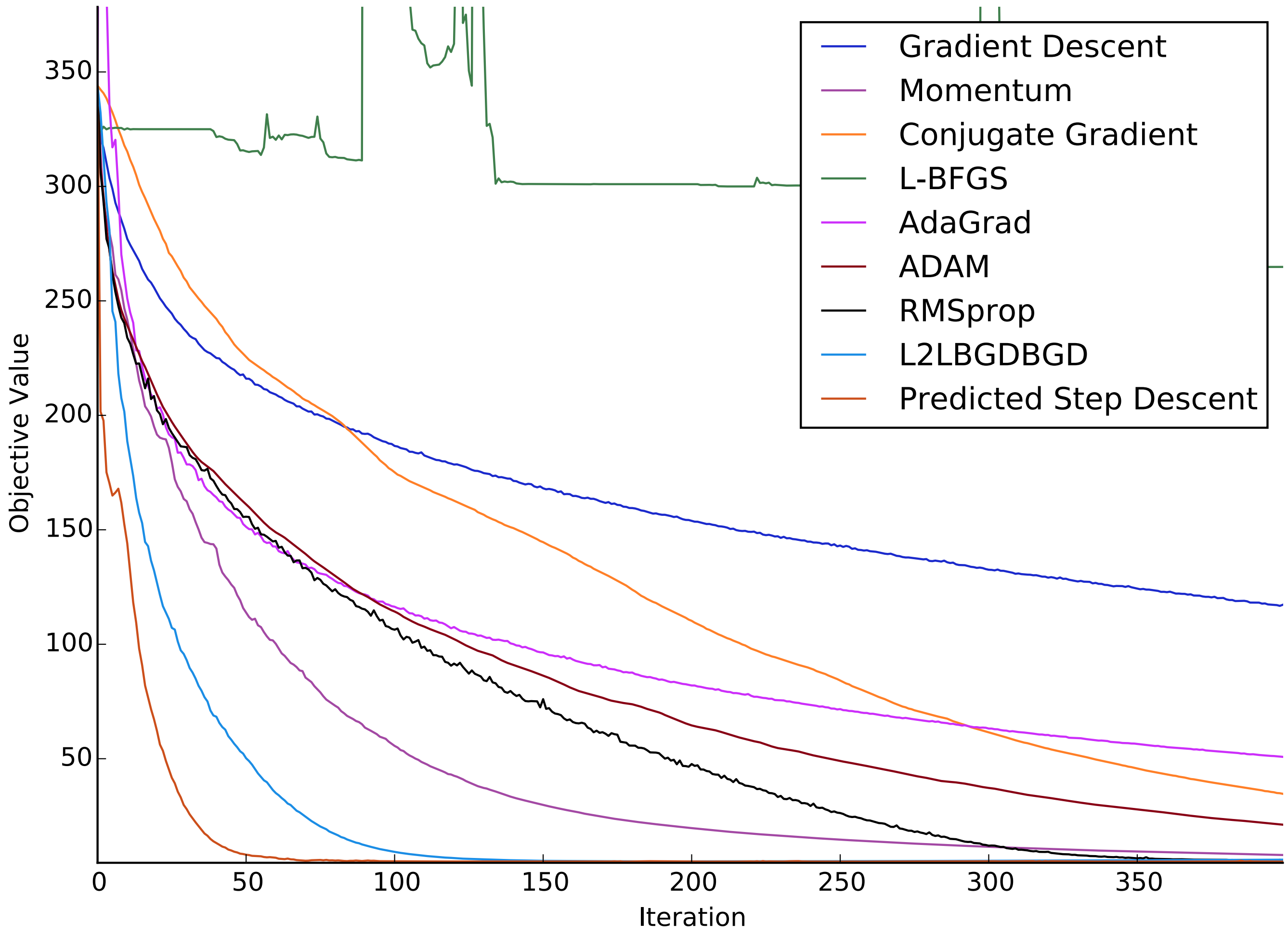
Berkeley UNIVERSITY OF CALIFORNIA

Larger Architecture (CIFAR-10)

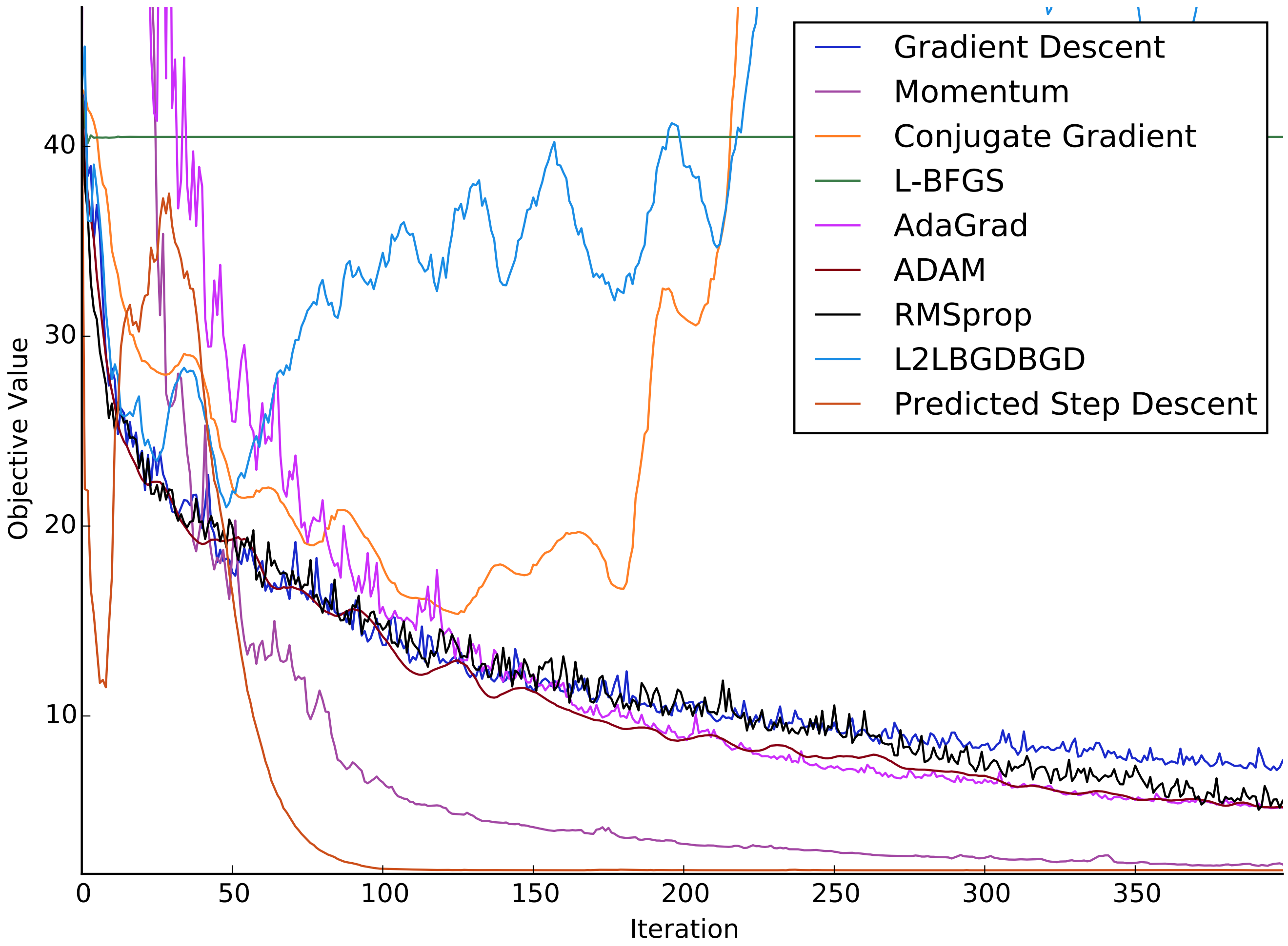


Berkeley UNIVERSITY OF CALIFORNIA

Larger Architecture (CIFAR-100)

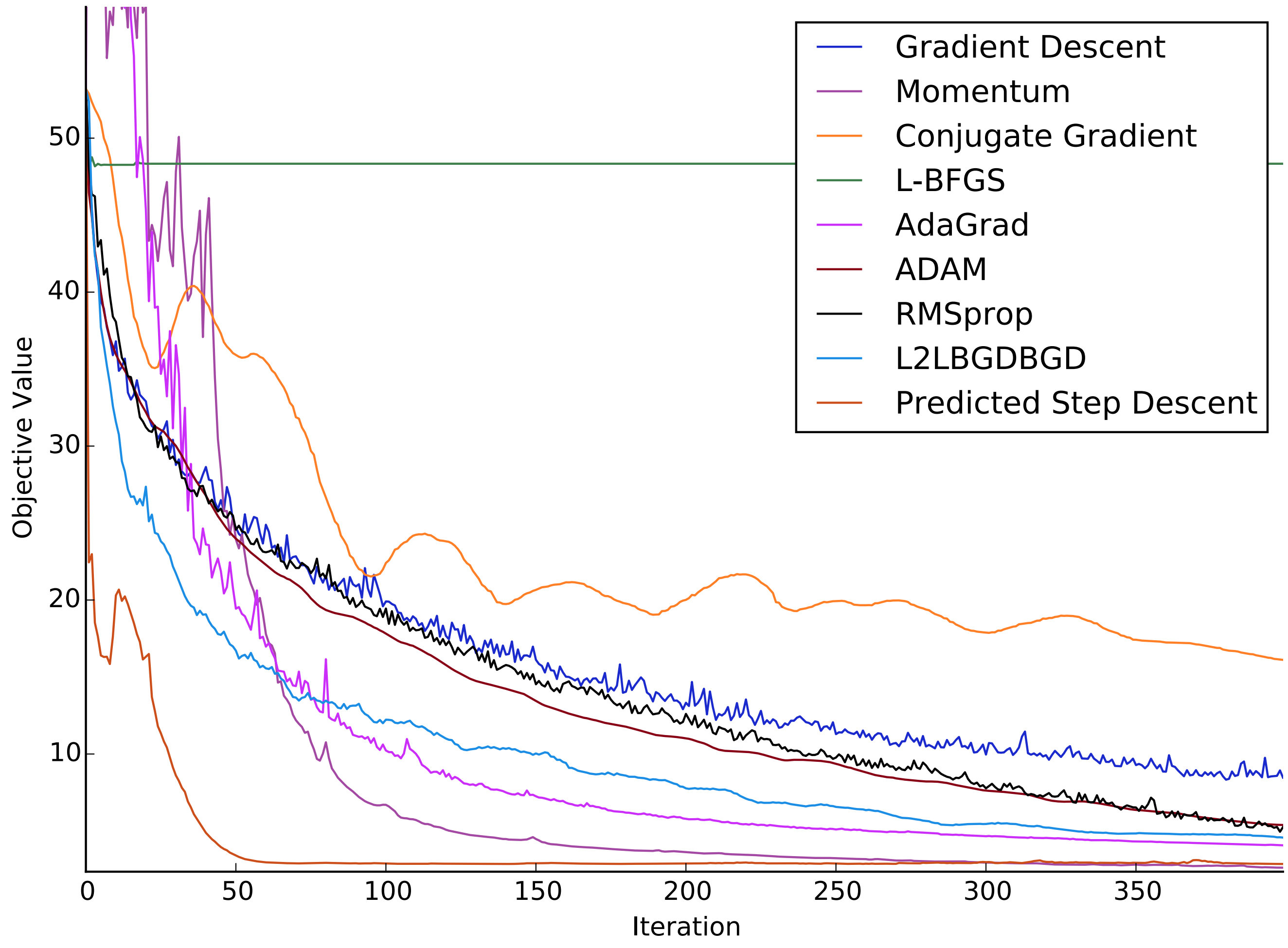


Noisier Gradients (TFD)



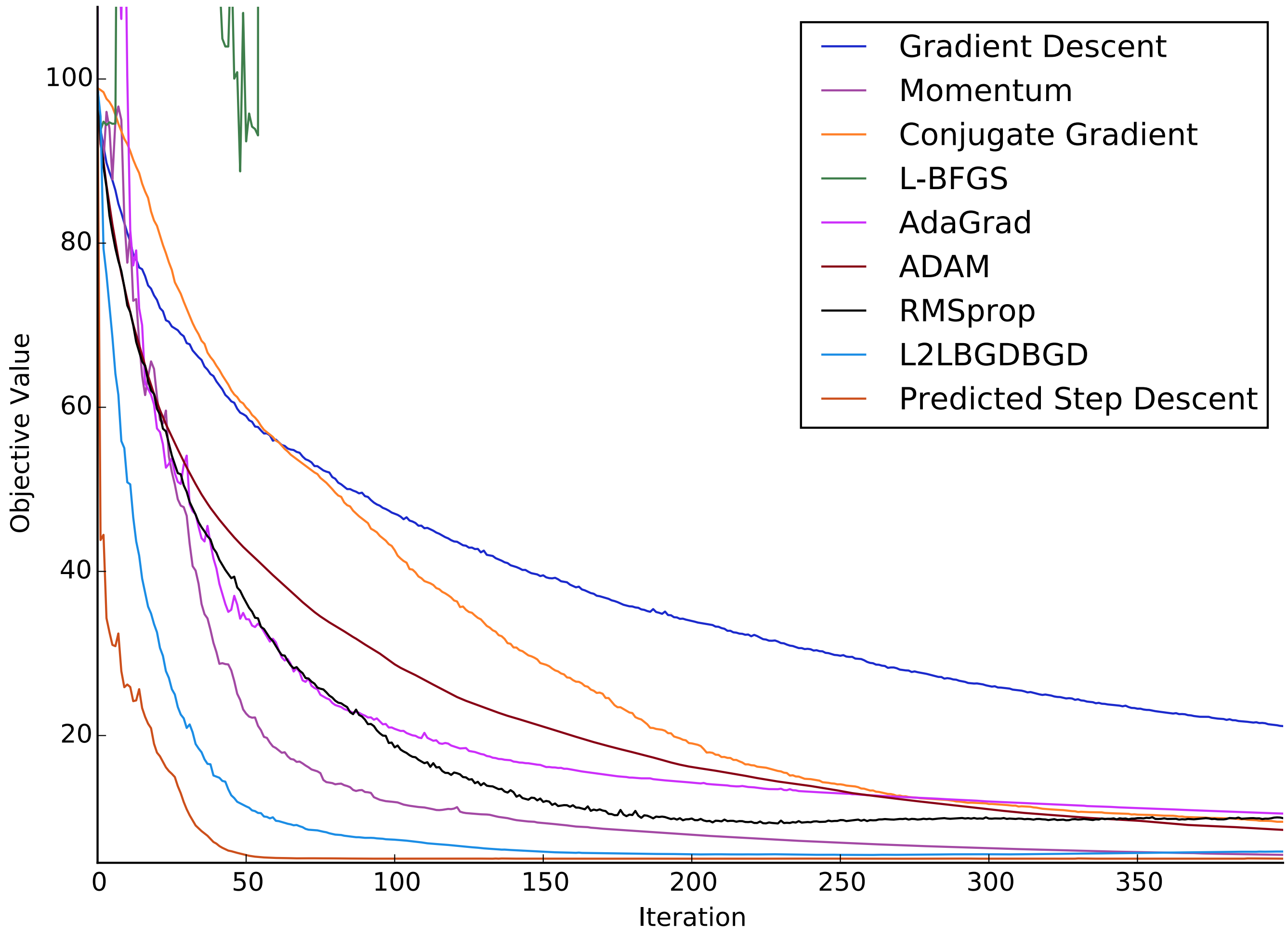
Berkeley
UNIVERSITY OF CALIFORNIA

Noisier Gradients (CIFAR-10)



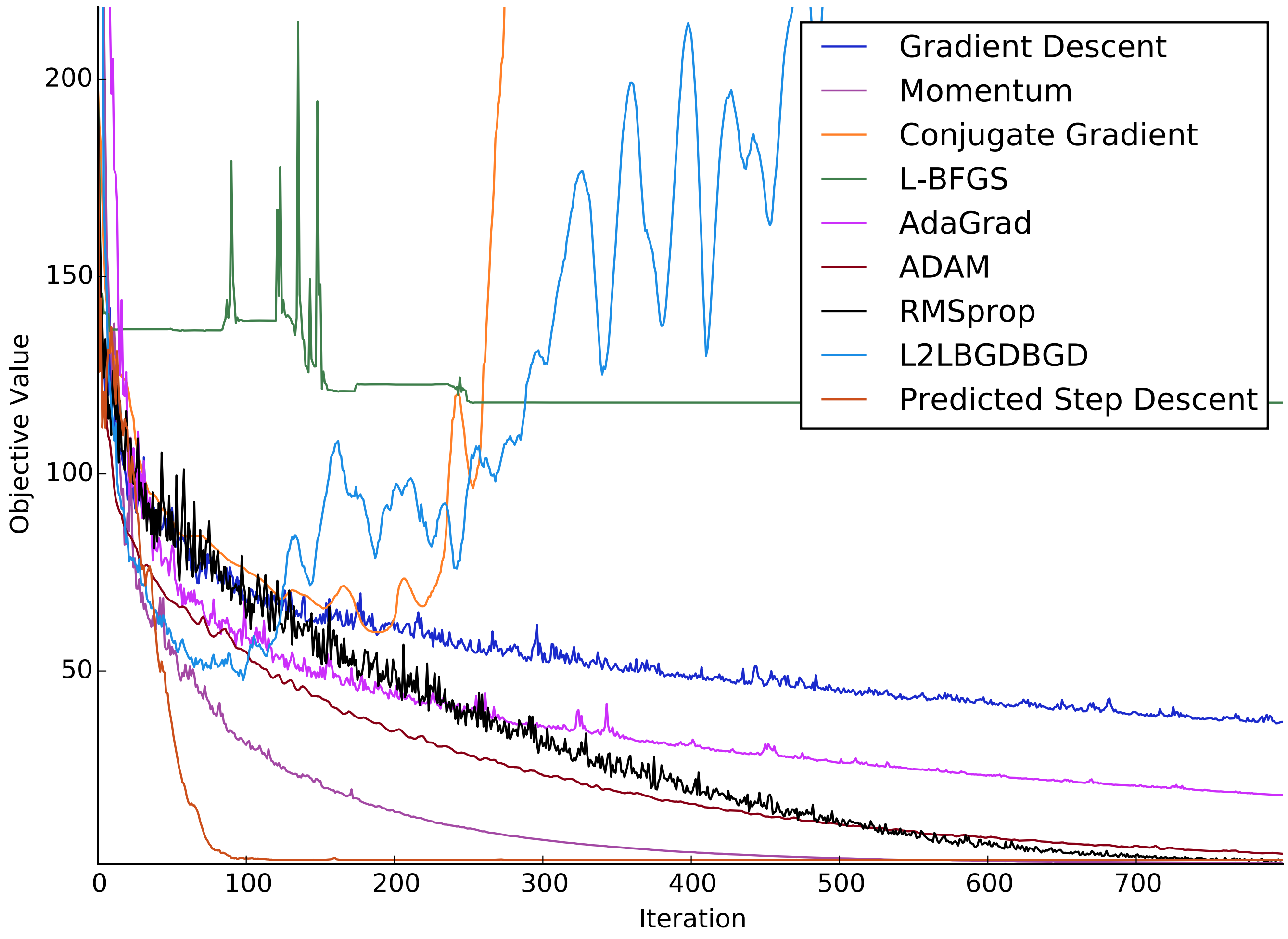
Berkeley
UNIVERSITY OF CALIFORNIA

Noisier Gradients (CIFAR-100)

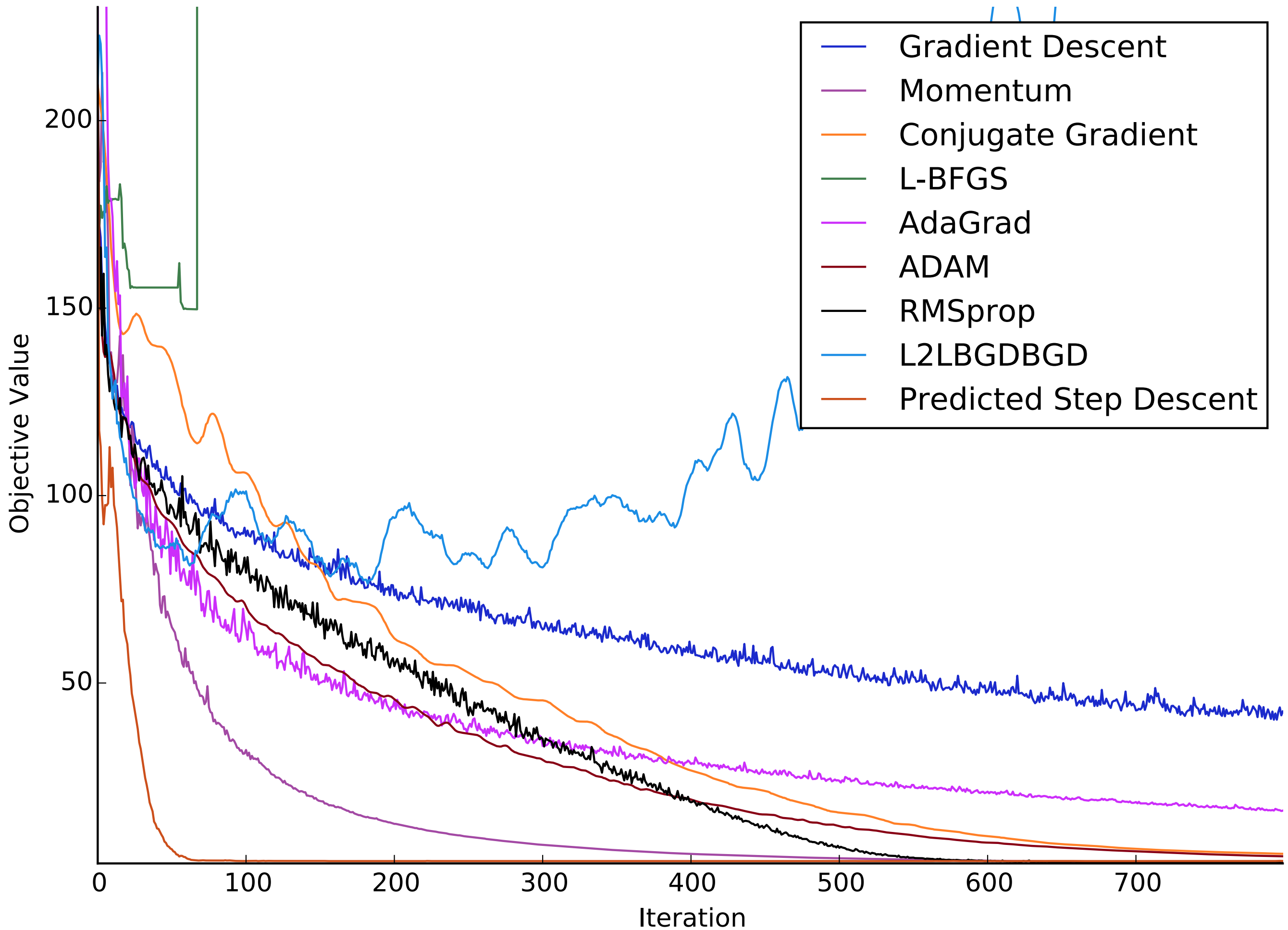


Berkeley
UNIVERSITY OF CALIFORNIA

Longer Time Horizon (TFD)

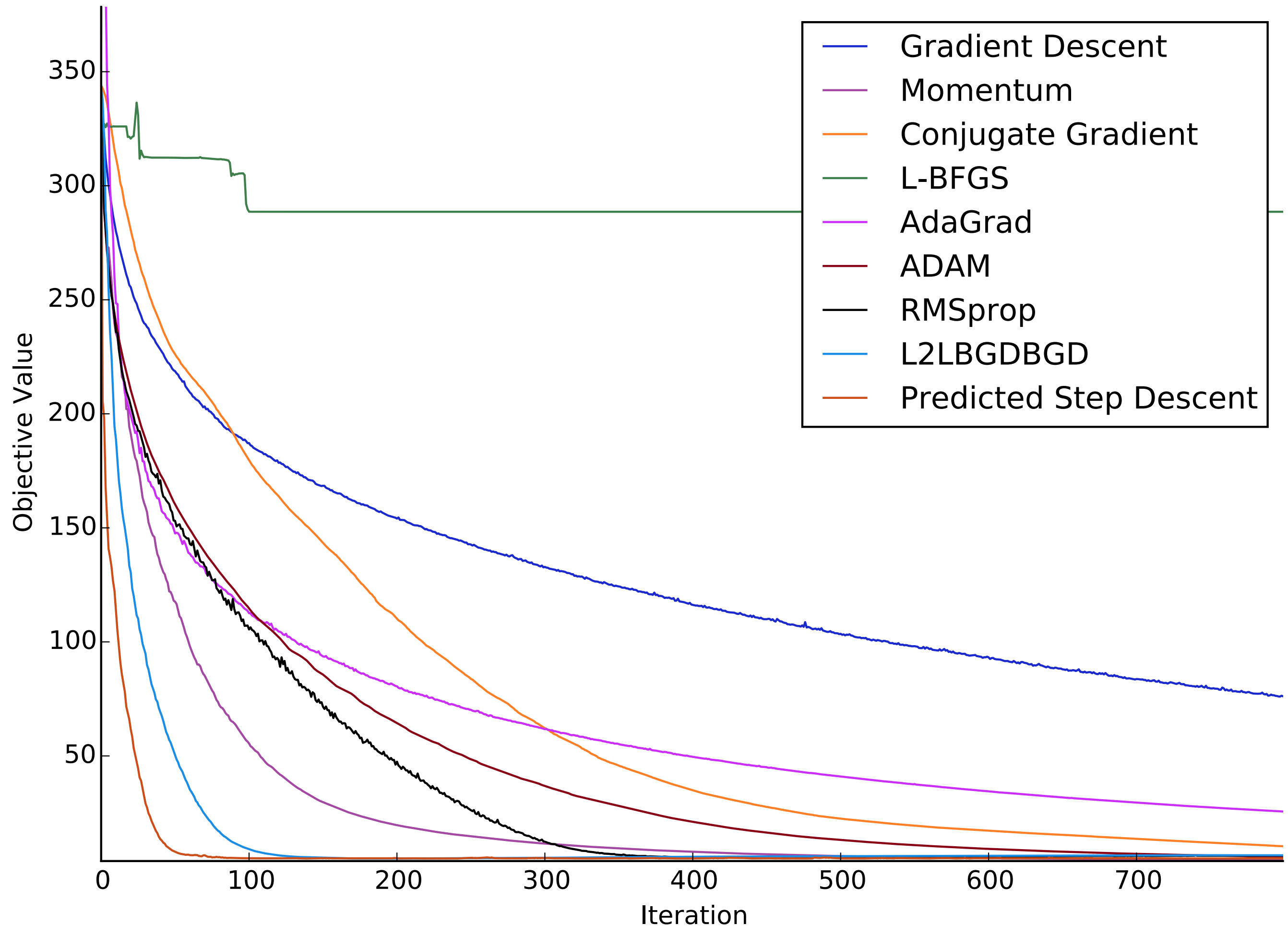


Longer Time Horizon (CIFAR-10)



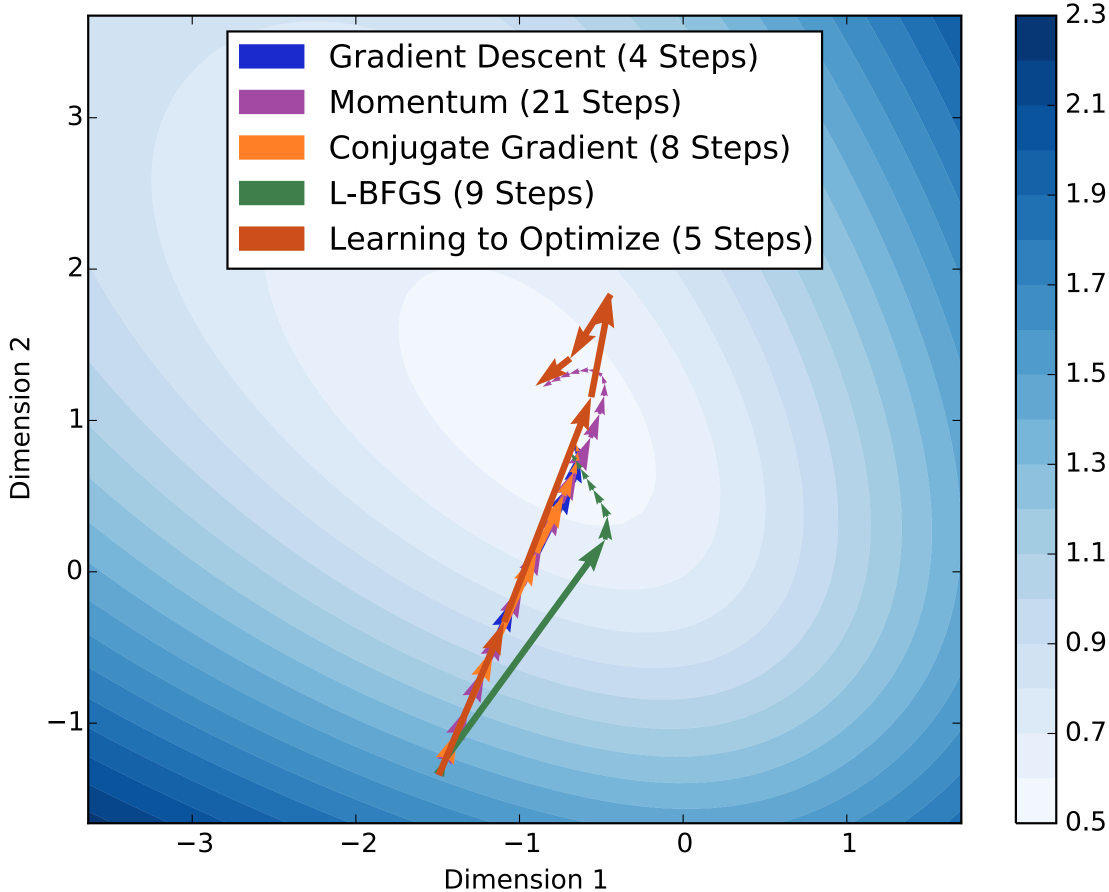
Berkeley
UNIVERSITY OF CALIFORNIA

Longer Time Horizon (CIFAR-100)



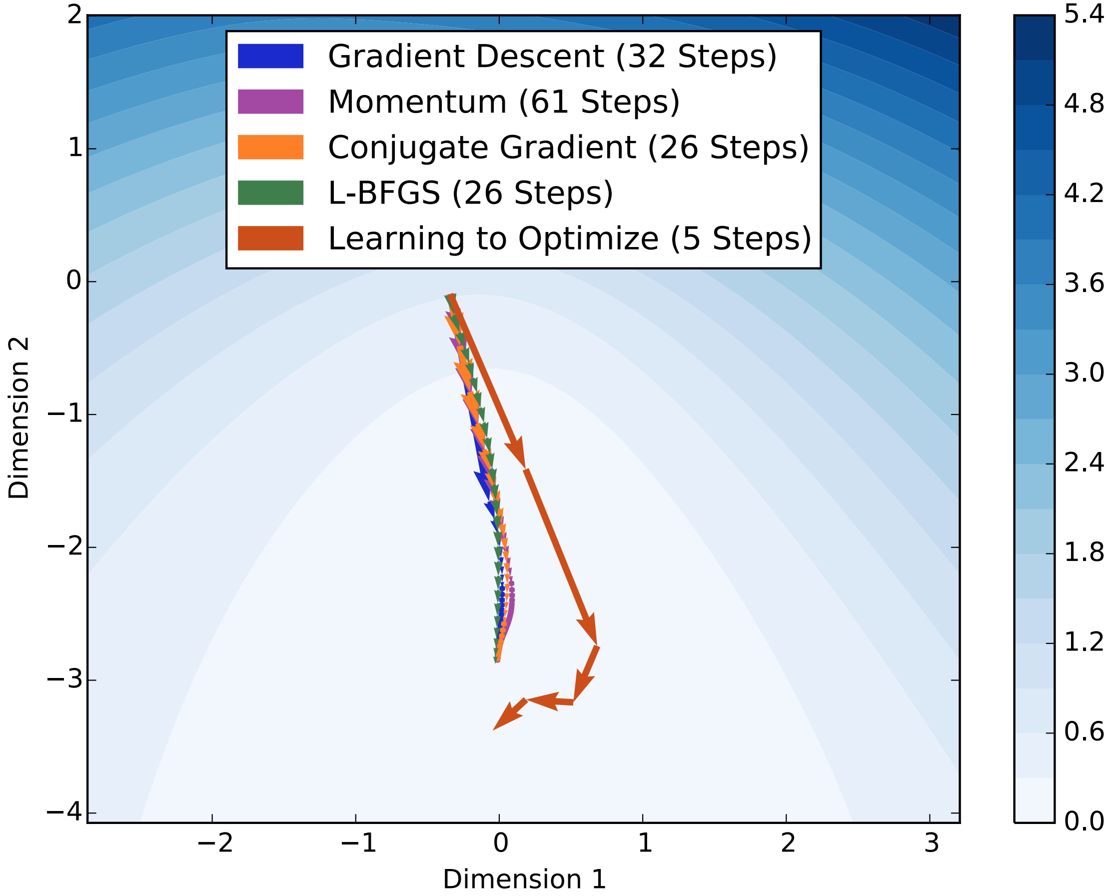
Berkeley
UNIVERSITY OF CALIFORNIA

2D Logistic Regression



Berkeley UNIVERSITY OF CALIFORNIA

2D Logistic Regression

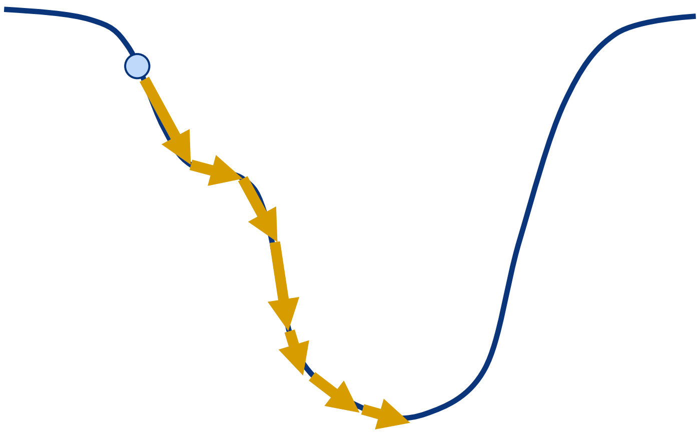


Berkeley UNIVERSITY OF CALIFORNIA

Generalization

Importance of Generalization

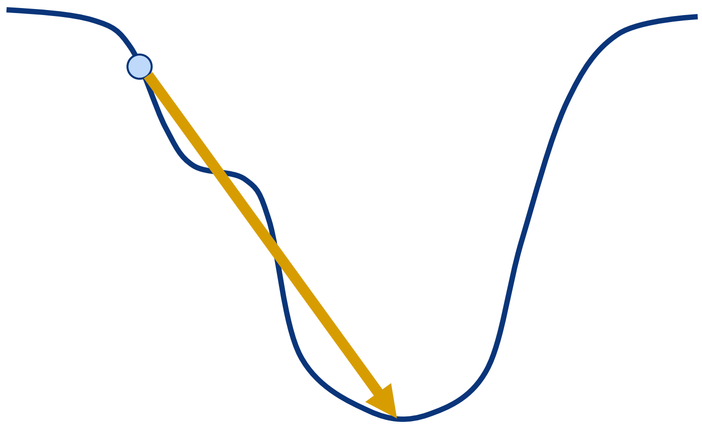
- Suppose we evaluate the performance of the optimizer on the training set.
- To learn an optimizer, we can simply run a traditional optimizer and memorize the solution.
- This is the best optimizer, since it gets to the optimum in one step.



Berkeley UNIVERSITY OF CALIFORNIA

Importance of Generalization

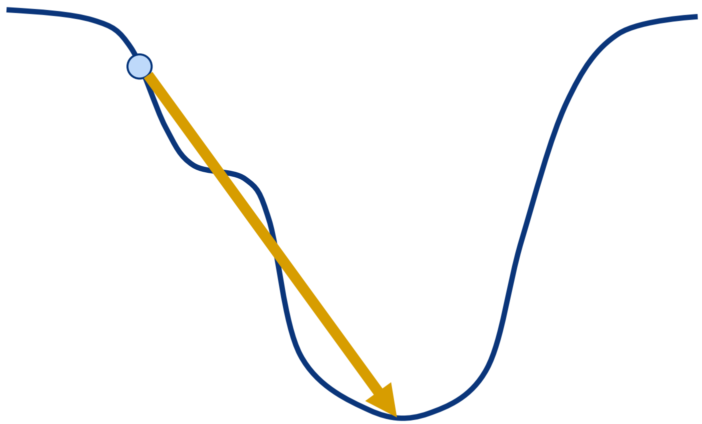
- Suppose we evaluate the performance of the optimizer on the training set.
- To learn an optimizer, we can simply run a traditional optimizer and memorize the solution.
- This is the best optimizer, since it gets to the optimum in one step.



Berkeley
UNIVERSITY OF CALIFORNIA

Importance of Generalization

- Suppose we evaluate the performance of the optimizer on the training set.
- To learn an optimizer, we can simply run a traditional optimizer and memorize the solution.
- This is the best optimizer



It would be pointless to learn the optimizer if we didn't care about generalization.

Extent of Generalization

- Generalization to *similar* base-models on *similar* tasks
  - Learned optimizer could memorize parts of the optimal parameters that are common across tasks and base-models.
    - E.g.: Weights of the lower layers in neural nets
  - Essentially the same as learning *what* to learn.

Extent of Generalization

- Stronger notion: Generalization to *similar* base-models on *dissimilar* tasks
  - The optimal parameters for dissimilar tasks are likely completely different.
  - An optimizer that memorizes any part of the optimal parameters will fail.
  - An optimizer that works in this setting must have learned not what the optimum is, but how to find it.

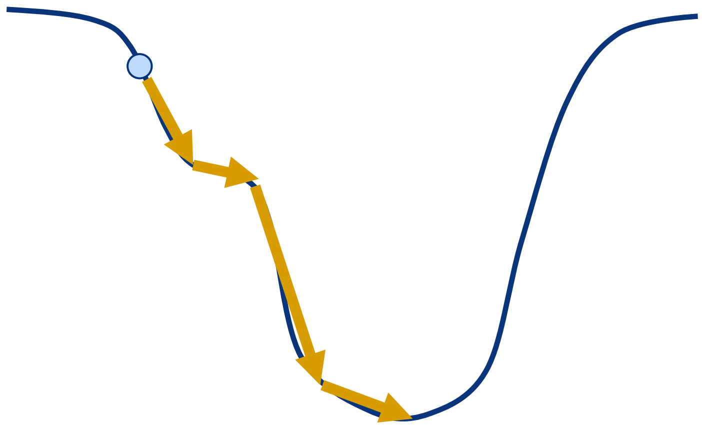
Extent of Generalization

- Even stronger notion: Generalization to *dissimilar* base-models on *dissimilar* tasks
  - The objective functions at test time could be arbitrarily different from the objective functions seen during training.
  - This is impossible – there is no optimizer that works well on all possible objective functions.

Berkeley
UNIVERSITY OF CALIFORNIA

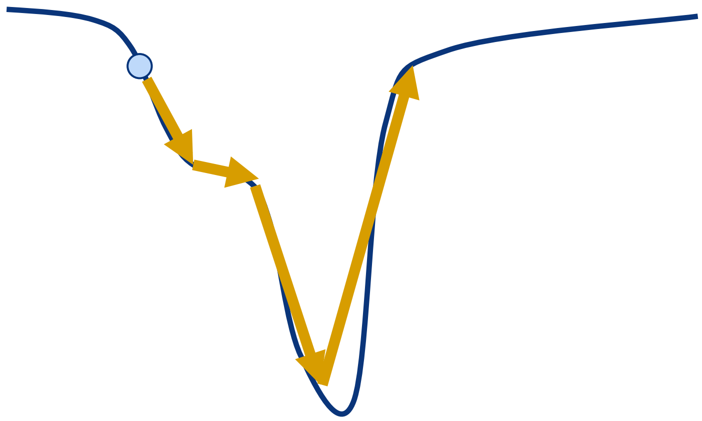
Extent of Generalization

- Given any optimizer, we can always find an objective function that it performs poorly on.
- We can simply change the objective function so that the final objective value is large.



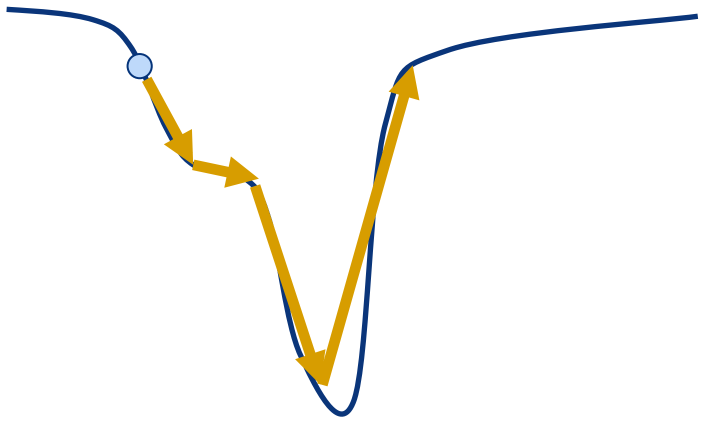
Extent of Generalization

- Given any optimizer, we can always find an objective function that it performs poorly on.
- We can simply change the objective function so that the final objective value is large.



Extent of Generalization

- Given any optimizer, we can always find an objective function that it performs poorly on.
- We can simply change the objective function so that the final objective value is large.



It is not possible for the learned optimizer to generalize to all possible objective functions.

Berkeley UNIVERSITY OF CALIFORNIA

Problem of Supervised Learning

- Supervised learning requires one of the following:
  – Observations at each time step are i.i.d., or
  – The dependence of the future observation on the current observation is known.
- In our setting, neither is true:
  – The step the optimizer takes affects future gradients.
  – How the current step affects the next gradient, i.e. the local geometry, is not known at test time.

Berkeley
UNIVERSITY OF CALIFORNIA

Problem of Supervised Learning

- When backpropagating through time, supervised learning essentially assumes the local geometry of an unseen objective function is the same as the local geometry of *one* of the training objective functions at *all* time steps.
  - In other words, it assumes $p\left(s_{t+1} \left| s_t, a_t \right.\right)$ is known and models it using the Hessians of the training objective functions.
  - This is incorrect, since the Hessians of an unseen objective function will be different.
- Hence, supervised learning overfits to the geometries of training objective functions.

Problem of Supervised Learning

- When an optimizer trained with supervised learning is applied to an unseen objective function:
  - It takes a step,

    → sees an unexpected gradient at the next iteration,

    → takes a step that is slightly off,

    → finds out the next gradient is even more unexpected,

    → takes another step that is more off,

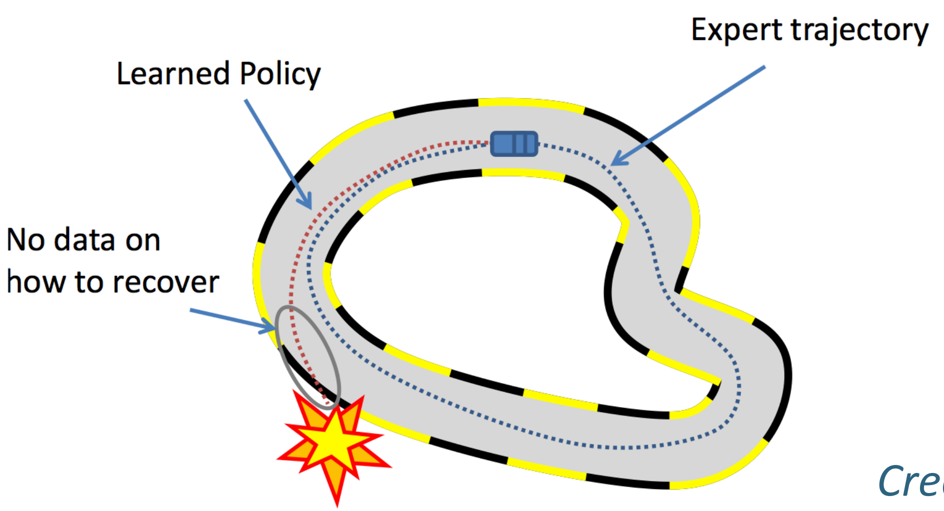
    …

    → eventually diverges.

Berkeley UNIVERSITY OF CALIFORNIA

Problem of Supervised Learning

- This is known as the problem of compounding errors.
  - Supervised learning leads to a cumulative error that grows quadratically in the time horizon, rather than linearly. (Ross & Bagnell, 2010)



Credit: John Schulman

Berkeley UNIVERSITY OF CALIFORNIA

Why RL Solves This Problem

- Reinforcement learning algorithm does not assume knowledge of $p\left(s_{t+1} \mid s_t, a_t\right)$, which characterizes the geometries of training objective functions.
  - So, conditions at meta-training and meta-test times match.
  - The learned policy must account for the uncertainty in $p\left(s_{t+1} \mid s_t, a_t\right)$, and must know how to recover from mistakes.

Reinforcement Learning Method

Guided Policy Search

- An (approximate) policy search algorithm for continuous state and action spaces. (Levine et al., 2015)
- Maintains two policies, ψ and π .
  - ψ lies in a time-varying linear policy class.
    - Optimal policy can be found in closed form.
  - π lies in a stationary non-linear policy class.
- Alternates between solving for ψ and π .

Berkeley
UNIVERSITY OF CALIFORNIA

ADMM

- Alternating direction method of multipliers (Boyd et al., 2011) solves the following problem:

$$\min_{\theta \in \Theta, \eta \in H} f(\theta) + g(\eta) \text{ s.t. } A\theta + B\eta = c$$

where f and g are convex functions, and Θ and H are closed convex sets.

- It alternates between the following updates:

$$\theta^{(t+1)} \leftarrow \arg\min_{\theta \in \Theta} f(\theta) + \langle \lambda^{(t)}, A\theta + B\eta^{(t)} - c \rangle + \frac{\rho}{2} \left\| A\theta + B\eta^{(t)} - c \right\|_2^2$$

$$\eta^{(t+1)} \leftarrow \arg\min_{\eta \in H} g(\eta) + \langle \lambda^{(t)}, A\theta^{(t+1)} + B\eta - c \rangle + \frac{\rho}{2} \left\| A\theta^{(t+1)} + B\eta - c \right\|_2^2$$

$$\lambda^{(t+1)} \leftarrow \lambda^{(t)} + \rho \left(A\theta^{(t+1)} + B\eta^{(t+1)} - c \right)$$

Bregman ADMM

- Bregman ADMM (Wang & Banerjee, 2014) generalizes ADMM and uses Bregman divergence as penalty. It solves:

$$\min_{\theta\in\Theta,\eta\in H} f(\theta) + g(\eta) \text{ s.t. } A\theta + B\eta = c$$

  where f and g are convex functions, and Θ and H are closed convex sets.
- It alternates between the following updates:

$$\theta^{(t+1)} \leftarrow \arg\min_{\theta\in\Theta} f(\theta) + \langle \lambda^{(t)}, A\theta + B\eta^{(t)} - c\rangle + \boxed{\rho B_\phi(c - A\theta, B\eta^{(t)})}$$

$$\eta^{(t+1)} \leftarrow \arg\min_{\eta\in H} g(\eta) + \langle \lambda^{(t)}, A\theta^{(t+1)} + B\eta - c\rangle + \boxed{\rho B_\phi(B\eta, c - A\theta^{(t+1)})}$$

$$\lambda^{(t+1)} \leftarrow \lambda^{(t)} + \rho\left(A\theta^{(t+1)} + B\eta^{(t+1)} - c\right)$$

Reinforcement Learning Problem

- Recall the reinforcement learning problem:

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

where the expectation is taken w.r.t.

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

State | Action | Initial State Distribution | Policy | Policy Parameters | State Transition Distribution

Berkeley
UNIVERSITY OF CALIFORNIA

Reinforcement Learning Problem

- Recall the reinforcement learning problem:

$$\min_{\theta} \mathbb{E}_{\theta} \left[\sum_{t=0}^{T} c(s_t) \right]$$

where the expectation is taken w.r.t.

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

State

Action

Initial State Distribution

Policy

Policy Parameters

State Transition Distribution

Guided Policy Search

- Guided Policy Search performs dual decomposition:

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

- It relaxes the problem by only enforcing equality on the first moments*:

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

*The Bregman divergence penalty is applied on the original distributions.

Guided Policy Search

- To solve the problem, it uses Bregman ADMM, which alternates between the following updates:

$$\eta \leftarrow \arg\min_{\eta} \sum_{t=0}^{T} \mathbb{E}_{\psi}\left[c(s_t) - \lambda_t^T a_t\right] + \nu_t \mathbb{E}_{\psi}\left[D_{KL}\left(\psi\left(a_t \middle| s_t, t; \eta\right) \middle\| \pi\left(a_t \middle| s_t; \theta\right)\right)\right]$$

$$\theta \leftarrow \arg\min_{\theta} \sum_{t=0}^{T} \lambda_t^T \mathbb{E}_{\psi}\left[\mathbb{E}_{\pi}\left[a_t \middle| s_t\right]\right] + \nu_t \mathbb{E}_{\psi}\left[D_{KL}\left(\pi\left(a_t \middle| s_t; \theta\right) \middle\| \psi\left(a_t \middle| s_t, t; \eta\right)\right)\right]$$

$$\lambda_t \leftarrow \lambda_t + \alpha \nu_t \left(\mathbb{E}_{\psi}\left[\mathbb{E}_{\pi}\left[a_t \middle| s_t\right]\right] - \mathbb{E}_{\psi}\left[a_t\right]\right) \ \forall t$$

- The optimization in the first update can be solved in closed form using a modification of linear-quadratic regulator (LQR).

Berkeley
UNIVERSITY OF CALIFORNIA

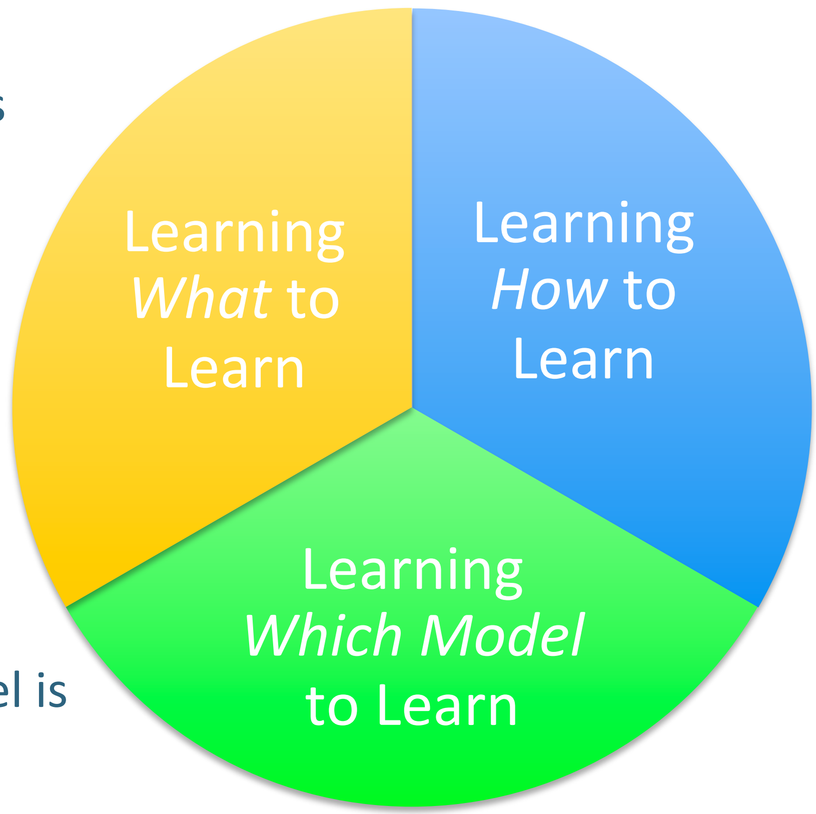
Landscape of Meta-Learning Methods

Berkeley
UNIVERSITY OF CALIFORNIA

Forms of Learning to Learn

Learn parameter values of the base-model that are useful across tasks.

- Transfer Learning
- Multi-Task Learning
- Few-Shot Learning



Learn how to train the base-model.

Learn which base-model is best suited for a task.

- Hyperparameter Optimization

Berkeley
UNIVERSITY OF CALIFORNIA

Learning *What* to Learn

Goal	• Learn what parameter values of the base-model are useful across tasks.
Meta-knowledge	• Intermediate features that are shared by tasks across the family, e.g. Gabor filters for vision tasks.
Extent of Generalization	• Need to generalize across *similar* tasks.
Parameterization Challenges	• Need to parameterize the space of intermediate features – this is straightforward.
Examples	• Transfer & multi-task learning, e.g. (Suddarth & Kergosien, 1990) • Few-shot learning, e.g. (Finn et al., 2017), (Snell et al., 2017)

Berkeley
UNIVERSITY OF CALIFORNIA

Learning *Which Model* to Learn

Goal	• Learn which base-model is best suited for a task.
Meta-knowledge	• Correlations between different base-models and their performance on different tasks.
Extent of Generalization	• Need to generalize across base-models, and *ideally*, across tasks.
Parameterization Challenges	• Need to parameterize the space of base-models – unclear how we can do this.
Examples	• Hyperparameter optimization – does not generalize across tasks • (Bradzil et al., 2003), (Schmidhuber, 2004), (Hochreiter et al., 2001)

Berkeley
UNIVERSITY OF CALIFORNIA

Learning *Which Model* to Learn

- (Bradzil et al., 2003): Enumerate a small set of base-models – not expressive.
- (Schmidhuber, 2004): Search over the space of all possible programs – takes exponential time.
- (Hochreiter et al., 2001): Search over base-models represented by a single step of a recurrent neural net – not expressive.
- Hyperparameter optimization: Search over a predefined set of hyperprameters – not expressive.

Goal	• Learn which base-model is best suited for a task.
Meta-knowledge	• Correlations between different base-models and their performance on different tasks.
Extent of Generalization	• Need to generalize across base-models, and *ideally*, across tasks.
Parameterization Challenges	• Need to parameterize the space of base-models – unclear how we can do this.
Examples	• Hyperparameter optimization – does not generalize across tasks • (Bradzil et al., 2003), (Schmidhuber, 2004), (Hochreiter et al., 2001)

Berkeley
UNIVERSITY OF CALIFORNIA

Learning *How* to Learn

Goal	• Learn how to train the base-model. • Learn about the *process*, rather the *outcome* of learning.
Meta-knowledge	• Commonalities in the behaviours of learning algorithms that achieve good performance.
Extent of Generalization	• Need to generalize across *dissimilar* tasks and/or *similar* base-models.
Parameterization Challenges	• Need to parameterize the space of learning algorithms. • Key Idea: Parameterize the update formula in optimizer.
Examples	• (Bengio et al., 1991) – learned algorithm indep. of tasks/base-models • (Li & Malik, 2016), (Andrychowicz et al., 2016), etc.

For More Details…

Learning to Optimize

Ke Li, Jitendra Malik

arXiv:1606.01885, 2016 and *ICLR*, 2017

Learning to Optimize Neural Nets

Ke Li, Jitendra Malik

arXiv:1703.00441, 2017

Berkeley
UNIVERSITY OF CALIFORNIA