Learning to Optimize

Ke Li
Jitendra Malik
{ke.li,malik}@eecs.berkeley.edu

Introduction
- Optimization problems are ubiquitous in science and engineering.
- Designing 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 \(f_1, \ldots, f_T\), a distribution \(D^T\) for initializing the iterate and a meta-loss \(L(f_1, \ldots, f_T)\) that measures the quality of the iterates \(x^0, \ldots, x^T\).
- An optimization algorithm \(A\) 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 \(A\) such that \(E_{D^T} x \in [L(f, A(f(x^0)))]\) is minimized.
- We choose \(L(f_1, \ldots, f_T) = \sum_{i} f_i(x^i)\)

Parameterizing Optimization Algorithms

Algorithm 1 General structure of optimization algorithms

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

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

Reinforcement Learning
- The goal of RL is to find: \[x^* = \arg \min_{x} \sum_{t=0}^{T-1} q(x_t, a_t) \]
where the expectation is taken w.r.t.

Experiment
- 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.
- Trained on the problems of training a neural net on Toronto Faces, CIFAR-10 and CIFAR-100.

Future Work
- 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.