Guarantees for Tuning the Step Size using a Learning-to-Learn Approach

Xiang Wang, Shuai Yuan, Chenwei Wu, Rong Ge
Duke University

How to train a neural net?

- How to train neural networks?
  - Just use SGD/Adam!

How to train an optimizer?

- Meta-gradient Explosion/ Vanishing
  - Objective: \( \min f(w) = \frac{1}{T} \sum_{t=1}^{T} \nabla f_{w}(t) \)
  - Algorithm: gradient descent with constant step
  - Naive meta-objective: loss at last step \( F(\eta) = f(w_{T}) \)
  - Theorem: For almost all values of \( \eta \), the meta-gradient \( F'(\eta) \) is either exponentially large or exponentially small in \( T \).

- Meta-stepping
  - Idea: use a meta-learning approach to tune hyper-parameters or learn a new optimizer!
  - Goal: find a good optimizer for a distribution of tasks.
  - Idea: Abstract the optimization algorithm as a mapping from the current state to the next state with parameter \( \theta \). Optimize the parameter \( \theta \) for the distribution of task.
  - Optimizer can be as simple as SGD with tunable step size, can also be as complicated as a deep neural network.

- Generalization of Trained Optimizer
  - Setting: least squares problem \( y = w^{\intercal} x + \xi, \|w\| = 1, x \sim N(0, I_{d}), \xi \sim N(0, \sigma^{2}) \)
  - Objective: squared loss on training data
    \[
    f(w) = \frac{1}{2n} \sum_{i=1}^{n} (y_i - w^{\intercal} x_i)^2
    \]
  - Algorithm: gradient descent with constant step size (similar for SGD)
    \[
    w_{t+1} = w_t - \eta f(w_t)
    \]
  - Two ways to define meta-objective
    1. Train-by-train: train meta-objective on training set, e.g., simply choose \( F(\eta) = f(w_{T}) \)
    2. Train-by-validation [Metz et al. 2019]
      - Use a separate validation set \((x_{i,v}^{\prime}, y_{i,v}^{\prime})\)…\((x_{i,v}^{V}, y_{i,v}^{V})\)
      - Define
        \[
        G(\eta) = \frac{1}{2|\mathcal{V}|} \sum_{i=1}^{n} (y_i^{\prime} - w_{T}^{\intercal} x_i^{\prime})^2
        \]
  - When do we need train-by-validation?
    1. Large noise and small sample size
      - The ERM solution is close to \( w^{*} \)
    2. Small noise and large sample size
      - \( w^{*} \) is either exponentially large or exponentially small in \( T \)

Learning to learn

- Learning to learn
  - \( w \)
  - Learning to learn
  - Optimizer \((\theta)\)
  - \( \Delta w \)
  - \( w = w + \Delta w \)
  - \( \nabla f(\theta) \)

Empirical Verification

- Step size tuning on least squares problems

MLP optimizer on MNIST dataset

Empirical Verification

- MLP optimizer on MNIST dataset

- Step size tuning on least squares problems

Graphical Overview

- Algorithm: gradient descent with constant step size
  - \( w_{t+1} = w_t - \eta f(w_t) \)
  - \( \eta \) is either exponentially large or exponentially small in \( T \)

- \( G(\eta) \) for all values of \( \eta \), the meta-gradient \( F'(\eta) \) is either exponentially large or exponentially small in \( T \).

- Theorem: For almost all values of \( \eta \), the meta-gradient \( F'(\eta) \) is either exponentially large or exponentially small in \( T \).

- Idea: meta-gradient is exponentially large (small) because the meta-objective is exponentially large (small) in \( T \).

- New objective: \( G(\eta) = \frac{1}{2|\mathcal{V}|} \sum_{i=1}^{n} (y_i^{\prime} - w_{T}^{\intercal} x_i^{\prime})^2 \)

- Theorem: The meta-gradient \( G'(\eta) \) is always polynomial in all relevant parameters.

- \( G'(\eta) = \frac{1}{2|\mathcal{V}|} F'(\eta) \), both terms are exponentially large or small, but they cancel each other.

- This is exactly how one would compute \( G'(\eta) \) using backpropagation → numerical issues!