BILEVEL OPTIMIZATION REPORT

Ivan Cheltsov

February 2022

Introduction

Consider the LASSO optimization problem:

$$\min_{x\in\mathbb{R}^n} F(x,\lambda) \triangleq f(x) + g(x,\lambda), \tag{1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is some smooth function. Both for the case, when g is smooth and non-smooth, various algorithms have been developed to tackle (1). However, the choice of the hyperparameter λ remains an open question. One approach, that we will be exploring below, is that of *hyperparameter optimization*. We are interested in solving bilevel problems of the form

$$\min_{\lambda\in\mathcal{D}\subset\mathbb{R}^r} L(\lambda) \triangleq C(\hat{x}(\lambda),\lambda) \tag{2}$$

$$\text{s.t. } \hat{x}(\lambda) = \arg\min_{x\in\mathbb{R}^n} F(x,\lambda), \tag{3}$$

where C is some criterion for the optimal λ for the inner problem solution. One type of approach to this is to consider the above as a single-level problem with constraints the optimality conditions of the inner problem. However, depending on the inner problem, the number of constraints may be large and the feasible set may be non-convex. This brings us to the other approach - solving (2) directly with iterative algorithms. We present a few of the these algorithms below.

Smooth Inner Problem

We first consider the case, where both C and F are smooth.

Assumptions

We will use a set of assumptions:

- We assume that the first derivatives of C and the second derivatives of F are Lipschitz continuous.
- We assume that the hessian $\nabla^2_x F$ is invertible at $(x(\lambda),\lambda)$.
- We assume that the hyperparameter domain $\mathcal{D}$ is a convex non-empty and compact subset in $\mathbb{R}^r$.

Prerequisites

All of the algorithms presented rely on two preliminary results. Firstly, we need to compute the gradient of the outer function C with respect to a point $\lambda \in \mathcal{D}$. By definition of $\hat{x}(\lambda)$, we have by the optimality condition, that

$$\nabla_x F(\hat{x}(\lambda),\lambda) = 0.$$

We can take the derivative of both sides. By chain rule, and implicit function theorem, we get

$$\nabla^2_{x\lambda} F(\hat{x}(\lambda),\lambda) + \nabla^2_{xx} F(\hat{x}(\lambda),\lambda)\nabla\hat{x}(\lambda) = 0,$$

that is,

$$\nabla \hat{x}(\lambda) = -\left[\nabla^2_{xx} F(\hat{x}(\lambda))\right]^{-1} \nabla^2_{x\lambda} F(\hat{x}(\lambda)).$$

Also, by chain rule we have

$$\nabla C(\hat{x}(\lambda), \lambda) = \nabla_\lambda C(\hat{x}(\lambda), \lambda) + \nabla \hat{x}(\lambda)^\top \nabla_x C(\hat{x}(\lambda), \lambda),$$

and so

$$\nabla C(\hat{x}(\lambda), \lambda) = \nabla_\lambda C(\hat{x}(\lambda) - \nabla^2_{x\lambda} F(\hat{x}(\lambda))^\top - \left[\nabla^2_{xx} F(\hat{x}(\lambda))\right]^{-\top} \nabla_x C(\hat{x}(\lambda), \lambda).$$

In the following algorithms $\nabla C(\hat{x}(\lambda), \lambda)$ will be computed in every outer-loop iteration. Of interest to us is the matvec product $\left[\nabla^2_{xx} F(\hat{x}(\lambda))\right]^{-\top} \nabla_x C(\hat{x}(\lambda)$. Computing and inverting the second derivative $\nabla^2_{yy} g$ is computationally expensive and so we consider methods that numerically approximate either of these operations.

This leads to our second preliminary result, regarding matrix inversions: if A is a positive-definite matrix, such that $\|A\| \leq 1$, then

$$A^{-1} = \sum_{i=0}^{\infty} (I - A)^i. \tag{4}$$

BA - Ghadimi (NEED TO UPDATE NOTATION)

Data: Input $x_0 \in \mathbb{R}^n, y_0 \in \mathbb{R}^m$, nonnegative sequences $\{\alpha_k\}_{k\geq 0}, \{\beta_t\}_{t\geq 0}$, integer sequence $\{t_k\}_{k\geq 0}$
for $k = 0, 1, \ldots$ **do**
  for $t = 0, 1, \ldots, t_k - 1$ **do**
    Iterate inner loop steps

$$y_{t+1} = y_t - \beta_t \nabla_y g(x_k, y_t).$$

  end
  Set $\bar{y}_k = y_{t_k}$ and iterate one outer loop step

$$x_{k+1} = \arg\min_{u \in X} \left\{ \langle \bar{\nabla} f(x_k, \bar{y}_k), u \rangle + \frac{1}{2\alpha_k} \|u - x_k\|^2 \right\}$$

  or, equivalently,

$$x_{k+1} = x_k - \alpha_k \bar{\nabla} f(x_k, \bar{y}_k)$$

end

Algorithm 1: Bilevel Approximation (BA) Method

To elaborate, here we have that

$$\bar{\nabla} f(\bar{x}, \bar{y}) = \nabla_x f(\bar{x}, \bar{y}) - \nabla^2_{yx} g(\bar{x}, \bar{y})^\top \left[\nabla^2_{yy} g(\bar{x}, \bar{y})\right]^{-\top} \nabla_y f(\bar{x}, \bar{y}).$$

We have to solve the inner problem to a high enough accuracy, so that the gradient approximation $\bar{\nabla} f(x_k, y_k)$ is close enough to $\nabla f(x_k, y^*(x_k))$. Otherwise we might have divergence in the outer problem. Hence we have the complexity-accuracy trade-off between solving the inner problem and solving the outer problem. To be more precise, we have the following result about the bound on the gradient error for f:

$$\|\bar{\nabla} f(x_k, y_k)\| \leq C \|\nabla f(x_k, y^*(x_k))\|,$$

where C depends on the Lipschitz constants and strong convexity parameters of the gradients of f, g. In the BA algorithm, every single gradient is computed directly. While this is feasible for first-order gradients, the complexity of second-order gradient calculations is $\mathcal{O}(n^3)$ (or $\mathcal{O}(mn^2)$). Furthermore, inverting the Hessian $\nabla^2_{yy} g$ has complexity $\mathcal{O}(n^3)$. As such, we are interested in incorporating numerical approximations in these steps, especially when working with larger datasets. Hence we may introduce a couple modifications to BA.

BA - Neumann

BA - SVRG-QUAD

Hyperparameter Optimization (HOAG)

We can summarize the algorithm in four steps as follows:

Data: Input $x_0 \in \mathbb{R}^n, \lambda_0 \in \mathcal{D} \subset \mathbb{R}^r$, nonnegative sequence $\{\epsilon_k\}_{k\geq 0}$, tolerance decrease sequence $\epsilon_k = 0.1 \times k^{-n}$ for $n = 2$ or 3; alternatively use exponential decrease sequence $\epsilon_k = 0.1 \times 0.9^k$.

for $k = 0, 1, \ldots$ **do**

- Solve inner optimization problem up to tolerance ϵ_k, for some x_k, i.e., such that:

$$\|\hat{x}(\lambda) - x_k\| \leq \epsilon_k;$$

- Solve the linear system

$$\nabla^2_{xx} F(x_k, \lambda_k) q_k = \nabla_x C(x_k, \lambda_k)$$

  for q_k up to tolerance ϵ_k:

$$\|\nabla^2_{xx} F(x_k, \lambda_k) q_k - \nabla_x C(x_k, \lambda_k)\| \leq \epsilon_k;$$

- Find approximate gradient p_k:

$$p_k = \nabla_\lambda C(x_k, \lambda_k) - \nabla^2_{x\lambda} F(x_k, \lambda_k)$$

- Update hyperparameters:

$$\lambda_{k+1} = P_{\mathcal{D}} \left(\lambda_k - \frac{1}{L} p_k \right)$$

end

Algorithm 2: HOAG Method

1 Non-Smooth Inner Problem

So far we worked with the assumption that the inner problem (3) is smooth. For example, when we use ridge regression. We now consider when this is not the case, i.e., with lasso.

References

[1] Ghadimi, S., & Wang, M. (2018). *Approximation Methods for Bilevel Programming.* arXiv:1802.02246, Optimization and Control.

Comparison of convergence of bilevel methods for Lasso problem. Fixed step 0.001 for outer loop.