Calculating Hypergradient

Jingchang Liu

November 13, 2019

HKUST

Table of Contents

Background

Bilevel optimization

Forward and Reverse Gradient-Based Hyperparameter Optimization

Conclusion

Q & A

Background

Hyperparameter Optimization

Tradeoff parameter

- The dataset is split in two: $\mathcal{S}_{\text{train}}$ and $\mathcal{S}_{\text{test}}$.
- Suppose we add ℓ_2 norm as the regulation term, then

$$\operatorname*{arg\,min}_{\lambda \in \mathcal{D}} \text{loss}(\mathcal{S}_{\text{test}}, X(\lambda)) \tag{1}$$

$$s.t. \;\; X(\lambda) \in \operatorname*{arg\,min}_{x \in \mathbb{R}^p} \text{loss}(\mathcal{S}_{\text{train}}, x) + e^{\lambda}\|x\|^2.$$

Stepsize

For gradient descent with momentum:

$$\begin{aligned} v_t &= \mu v_{t-1} + \nabla J_t(w_{t-1}), \\ w_t &= w_{t-1} - \eta\left(\mu v_{t-1} - \nabla J_t(w_{t-1})\right). \end{aligned}$$

Hyperparameters are μ and η.

Group Lasso

Traditional Group Lasso

To seduce the group sparse effect of parameter w, we do

$$\hat{w} \in \underset{w \in \mathbb{R}^p}{\arg\min} \frac{1}{2}\|y - Xw\|^2 + \lambda \sum_{l=1}^{L} \|w_{\mathcal{G}_l}\|_2, \tag{2}$$

where we partition features in L groups $\{\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_L\}$.

- But we need to do the partition by ourself beforehand.
- How to learn the partition?

Group Lasso

- Encapsulate the group structure by an hyperparameter $\theta = [\theta_1, \theta_2, \dots, \theta_L] \in \{0,1\}^{P \times L}$, where L is max number of groups and P is the number of features.
- $\theta_{p,l} = 1$ if the p-th feature belongs to the l-th group, and 0 otherwise.

Formulations for learning θ:

$$\hat{\theta} \in \underset{\theta \in \in \{0,1\}^{P \times L}}{\arg\min}\ C(\hat{w}(\theta)), \tag{3}$$

where $C(\hat{w}(\theta))$ can be the validation function $C(\hat{w}(\theta)) = \frac{1}{2}\left\|y^{'} - X^{'}w\right\|^2$, and

$$\hat{w}(\theta) = \underset{w \in \mathbb{R}^{P \times L}}{\arg\min} \frac{1}{2}\left\|y - Xw\right\|^2 + \lambda \sum_{l=1}^{L} \|\theta_l \odot w\ \|_2 \tag{4}$$

Bilevel optimization

Bilevel Optimization

We can conclude the following optimization problem:

$$\begin{aligned} \min_x \quad & f^U(x, y) \\ s.t. \quad & y \in \arg\min_{y'} f^L(x, y'), \end{aligned} \tag{5}$$

- f^U is the upper-level objective, over two variables x and y.
- f^L is the lower-level objective, which binds y as a function of x.
- (5) can be simply viewed as a special case of constrained optimization.
- If we can get the analytic solution $y^*(x)$ of y, then we just need to solve the single-level problem $\min_x f^U(x, y^*(x))$.

Gradient

Compute the gradient of the solution to the lower-level problem with respect to variables in the upper-level problem:

$$x = x - \eta \left(\frac{\partial f^U}{\partial x} + \frac{\partial f^U}{\partial y} \frac{\partial y}{\partial x} \right) |_{(x, y^*)}. \tag{6}$$

How to calculate $\frac{\partial y}{\partial x}$?

Theorem

Let $f : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$ be a continuous function with first and second derivatives. Let $g(x) = \arg\min_y f(x, y)$. Then the derivative of g with respect to x is

$$\frac{dg(x)}{dx} = -\frac{f_{XY}(x, g(x))}{f_{YY}(x, g(x))}. \tag{7}$$

where $f_{XY} = \frac{\partial^2 f}{\partial x \partial y}$ and $f_{YY} = \frac{\partial^2 f}{\partial y^2}$,

Proof

1. Since $g(x) = \arg\min_y f(x,y)$, we get $\frac{\partial f(x,y)}{\partial y}|_{y=g(x)} = 0$;
2. Differentiating lhs and rhs, we get $\frac{d}{dx}\frac{\partial f(x,g(x))}{\partial y} = 0$;
3. While by the chain rule, we have

$$\frac{d}{dx}\frac{\partial f(x,g(x))}{\partial y} = \frac{\partial^2 f(x,g(x))}{\partial x \partial y} + \frac{\partial^2 f(x,g(x))}{\partial y^2}\frac{dg(x)}{dx}; \tag{8}$$

   Equating to zero and rearranging gives:

$$\begin{aligned} \frac{dg(x)}{dx} &= \left(\frac{\partial^2 f(x,g(x))}{\partial y^2}\right)^{-1} \frac{\partial^2 f(x,g(x))}{\partial x \partial y} && (9) \\ &= -\frac{f_{XY}(x,g(x))}{f_{YY}(x,g(x))}. && (10) \end{aligned}$$

Lemma

Lemma 1

Let $f : \mathbb{R} \times \mathbb{R}^{\ltimes} \to \mathbb{R}$ be a continuous function with first and second derivatives. Let $g(x) = \arg\min_{y \in \mathbb{R}^n} f(x, y)$. Then the derivative of g with respect to x is

$$g^{'}(x) = -f_{XY}(x, g(x))^{-1} f_{YY}(x, g(x)). \tag{11}$$

where $f_{XY} = \nabla^2_{yy} f(x, y) \in \mathbb{R}^{n \times n}$ and $f_{YY} = \frac{\partial}{\partial x} \nabla_y f(x, y) \in \mathbb{R}^n$,

Application to hyperparameter optimization (icml 16)

Hyperparameter optimization

$$\arg\min_{\lambda \in \mathcal{D}} \operatorname{loss}(\mathcal{S}_{\text{test}}, X(\lambda)) \tag{12}$$

$$s.t.\ \ X(\lambda) \in \arg\min_{x \in \mathbb{R}^p} \operatorname{loss}(\mathcal{S}_{\text{train}}, x) + e^{\lambda}\|x\|^2.$$

Gradient descent for bilevel problem

$$\begin{aligned} x &= x - \eta \left(\frac{\partial f^U}{\partial x} + \frac{\partial f^U}{\partial y} \frac{\partial y}{\partial x} \right) |_{(x, y^*)} && (13) \\ &= x - \eta \left(\frac{\partial f^U}{\partial x} - \frac{\partial f^U}{\partial y} \left(\frac{\partial^2 f(x, g(x))}{\partial y^2} \right)^{-1} \frac{\partial^2 f(x, g(x))}{\partial x \partial y} \right) && (14) \end{aligned}$$

Gradient

$$\nabla f = \nabla_2 g - \left(\nabla^2_{1,2} h\right)^T \left(\nabla^2_1 h\right)^{-1} \nabla_1 g$$

HOAG

Algorithm 1 (HOAG). *At iteration $k = 1, 2, \ldots$ perform the following:*

(i) Solve the inner optimization problem up to tolerance ε_k. That is, find x_k such that

$$\left\|X(\lambda_k) - x_k\right\| \leq \varepsilon_k \quad .$$

(ii) Solve the linear system $\nabla_1^2 h(x_k, \lambda_k) q_k = \nabla_1 g(x_k, \lambda_k)$ for q_k up to tolerance ε_k. That is, find q_k such that

$$\left\|\nabla_1^2 h(x_k, \lambda_k) q_k - \nabla_1 g(x_k, \lambda_k)\right\| \leq \varepsilon_k \quad .$$

(iii) Compute approximate gradient p_k as

$$p_k = \nabla_2 g(x_k, \lambda_k) - \nabla_{1,2}^2 h(x_k, \lambda_k)^T q_k \quad ,$$

(iv) Update hyperparameters:

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

Conclusion

- If the sequence $\{\epsilon_i\}_{i=1}^{\infty}$ is summable, then this implies the convergence to a stationary point of f.

Theorem
If the sequence $\{\epsilon_i\}_{i=1}^{\infty}$ is positive and verifies

$$\sum_{i=1}^{\infty} \epsilon_i < \infty,$$

then the sequence λ_k of iterates in the HOAG algorithm has limit $\lambda^* \in \mathcal{D}$. In particular, if λ^* belongs to the interior of $\mathcal{D}$, it is verified then

$$\nabla f(\lambda^*) = 0.$$

Forward and Reverse Gradient-Based Hyperparameter Optimization

Formulation I

- Focus on training procedures of an objective function $J(w)$ with respect to w.
- The training procedures of SGD or its variants like momentum, RMSProp and ADAM can be regarded as a dynamical system with a state $s_t \in \mathbb{R}^d$.

$$s_t = \Phi_t(s_{t-1}, \lambda),\ \ t = 1, \ldots, T$$

- For gradient descent with momentum:

$$\begin{aligned} v_t &= \mu v_{t-1} + \nabla J_t(w_{t-1}), \\ w_t &= w_{t-1} - \eta\left(\mu v_{t-1} - \nabla J_t(w_{t-1})\right). \end{aligned}$$

- 1. $s_t = (w_t, v_t), s_t \in \mathbb{R}^d$.
  2. $\lambda = (\mu, \eta)$, $\lambda \in \mathbb{R}^m$.
  3. $\Phi_t : (\mathbb{R}^d \times \mathbb{R}^m) \to \mathbb{R}^d$.

Formulation II

- The iterates $s_1, \dots, s_T$ implicitly depend on the vector of hyperparameters λ.
- Goal: optimize the hyperparameters according to a certain error function E evaluated at the last iterate s_T.
- We wish to solve the problem

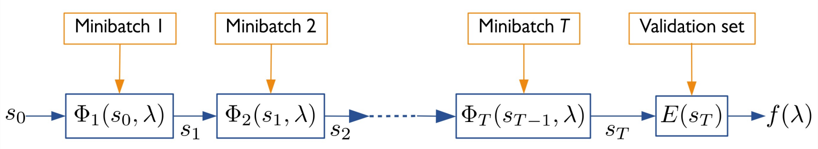
$$\min_{\lambda \in \Lambda} f(\lambda),$$

  where the set $\Lambda \subset \mathbb{R}^m$ incorporates constraints on the hyperparameters.
- The response function $f : \mathbb{R}^m \to \mathbb{R}$, defined at $\lambda \in \mathbb{R}^m$

$$f(\lambda) = E(s_T(\lambda)).$$

Diagram

Figure 1: The iterates $s_1, \ldots, s_T$ depend on the hyperparameters λ



- Change the bilevel program to use the parameters at the last iterate s_T rather than $\hat{w}$,

$$\min_{\lambda \in \Lambda} f(\lambda),$$

where

$$f(\lambda) = E(s_T(\lambda)).$$

- The hypergradient

$$\nabla f(\lambda) = \nabla E(s_T) \frac{d\, s_T}{d\, \lambda}.$$

Forward-Mode to calculate hypergradient

- Chain rule:

$$\nabla f(\lambda) = \nabla E(s_T) \frac{d\, s_T}{d\, \lambda},$$

  where $\frac{d\, s_T}{d\, \lambda}$ is the $d \times m$ matrix.

- Sine $s_t = \Phi_t(s_{t-1}, \lambda)$, Φ_t depends on λ both directly and indirectly through the state s_{t-1}:

$$\frac{d\, s_t}{d\, \lambda} = \frac{\partial\, \Phi_t(s_{t-1}, \lambda)}{\partial s_{t-1}} \frac{d\, s_{t-1}}{d\, \lambda} + \frac{\partial \Phi_t(s_{t-1}, \lambda)}{\partial\, \lambda}.$$

- Defining $Z_t = \frac{d\, s_t}{d\, \lambda}$, we rewrite it as

$$Z_t = A_t Z_{t-1} + B_t,\ \ t \in \{1, \ldots, T\}.$$

Forward-mode Recurrence

Figure 2: Recurrence

$$
\begin{aligned}
\nabla f(\lambda) &= \nabla E(s_T) Z_T \\
&= \nabla E(s_T)(A_T Z_{T-1} + B_T) \\
&= \nabla E(s_T)(A_T A_{T-1} Z_{T-2} + A_T B_{T-1} + B_T) \\
&\vdots \\
&= \nabla E(s_T) \sum_{t=1}^{T} \left(\prod_{s=t+1}^{T} A_s \right) B_t. \qquad (15)
\end{aligned}
$$

Figure 3: Forward-HG algorithm

Algorithm 2 FORWARD-HG

Input: λ current values of the hyperparameters, s_0 initial optimization state
Output: Gradient of validation error w.r.t. λ
$Z_0 = 0$
for $t = 1$ **to** T **do**
  $s_t = \Phi_t(s_{t-1}, \lambda)$
  $Z_t = A_t Z_{t-1} + B_t$
end for
return $\nabla E(s) Z_T$

Reverse-Mode to calculate hypergradient

Reformulate original problem as the constrained opt problem

$$\begin{aligned} \min_{\lambda, s_1, \ldots, s_T} \quad & E(s_T), \\ s.t. \quad & s_t = \Phi_t(s_{t-1}, \lambda),\ t \in \{1, \ldots, T\}. \end{aligned}$$

Lagrangian

$$\mathcal{L}(s, \lambda, \alpha) = E(s_T) + \sum_{t=1}^{T} \alpha_t (\Phi_t(s_{t-1}, \lambda) - s_t).$$

Partial derivation of the Lagrangian

$$
\begin{aligned}
\frac{\partial \mathcal{L}}{\partial \alpha_t} &= \Phi_t(s_{t-1}, \lambda) - s_t, \quad t \in \{1, \ldots, T\} \\
\frac{\partial \mathcal{L}}{\partial s_t} &= \alpha_{t+1} A_{t+1} - \alpha_t, \quad t \in \{1, \ldots, T-1\} \\
\frac{\partial \mathcal{L}}{\partial s_T} &= \nabla E(s_T) - \alpha_T \\
\frac{\partial \mathcal{L}}{\partial \lambda} &= \sum_{t=1}^{T} \alpha_t B_t,
\end{aligned}
$$

Derivations

Notation

$$A_t = \frac{\partial \Phi_t(s_{t-1}, \lambda)}{\partial s_{t-1}},\ B_t = \frac{\partial \Phi_t(s_{t-1}, \lambda)}{\partial \lambda},$$

note that $A_t \in \mathbb{R}^{d \times d}$ and $B_t \in \mathbb{R}^{d \times m}$.

$\frac{\partial \mathcal{L}}{\partial s_t} = 0$ **and** $\frac{\partial \mathcal{L}}{\partial s_T} = 0$

$$\alpha_t = \begin{cases} \nabla E(s_T) & \text{if } t = T, \\ \nabla E(s_T) A_T \cdots A_{t+1} & \text{if } t \in \{1, \ldots, T-1\}. \end{cases} \tag{15}$$

Since $\frac{\partial \mathcal{L}}{\partial \lambda} = \sum_{t=1}^{T} \alpha_t B_t$,

$$\frac{\partial \mathcal{L}}{\partial \lambda} = \nabla E(s_T) \sum_{t=1}^{T} \left(\prod_{s=t+1}^{T} A_s \right) B_t.$$

Reverse-HG algorithm

Figure 4: Reverse-HG algorithm

Algorithm 1 REVERSE-HG

Input: λ current values of the hyperparameters, s_0 initial optimization state
Output: Gradient of validation error w.r.t. λ
for $t = 1$ **to** T **do**
  $s_t = \Phi_t(s_{t-1}, \lambda)$
end for
$\alpha_T = \nabla E(s_T)$
$g = 0$
for $t = T - 1$ **downto** 1 **do**
  $g = g + \alpha_{t+1} B_{t+1}$
  $\alpha_t = \alpha_{t+1} A_{t+1}$
end for
return g

TRUNCATED BACK-PROPAGATION

$$t = T - 1 \text{ to } T - k.$$

Real-Time HO

- For $t \in \{1, \ldots, T\}$, define

$$f_t(\lambda) = E(s_t(\lambda)).$$

- Partial hypergradients are avaliable in forward mode

$$\nabla f_t(\lambda) = \frac{d\, E(s_t)}{d\, \lambda} = \nabla E(s_t) Z_t.$$

- Significant: we can update hyperparameters in a single epoch, without having to wait until time T.

Figure 5: The iterates $s_1, \ldots, s_T$ depend on the hyperparameters λ

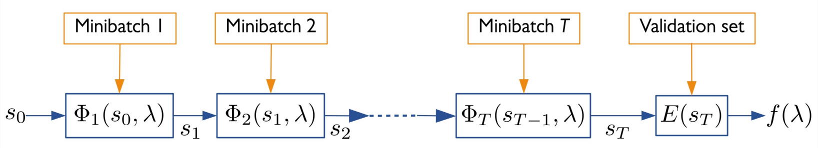


Figure 6: Real-Time HO algorithm

RTHO(λ, s_0)

Inputs: initial hyperparameters, λ, initial state, s_0
Outputs: Final parameters, s_T
$Z_0 = 0$
for $t = 1$ to T
    $s_t = \Phi_t(s_{t-1}, \lambda)$ // d vector
    $A_t = \frac{\partial \Phi_t(s_{t-1}, \lambda)}{\partial s_{t-1}}$ // $d \times d$ matrix
    $B_t = \frac{\partial \Phi_t(s_{t-1}, \lambda)}{\partial \lambda}$ // $d \times m$ matrix
    $Z_t = A_t Z_{t-1} + B_t$ // $d \times m$ matrix
    // Memory for A_t, B_t, Z_t can be reused!
    if $t == 0 \pmod{\Delta}$
        $\lambda = \lambda - \eta \nabla E(s_t) Z_t$
return s_T

Analysis

- Forward and inverse mode have different time/space tradeoffs.
- Reverse mode needs to store the whole history of parameters.
- Forward mode need to calculate mat multipy mat in each step.

Conclusion

Conclusions

- Calculating the hypergradients, the gradients with respect to hyperparameters, is very important in selecting a good hyperparameter.
- We talk about two ways for calculating hypergradients: bilevel optimization and forward/inverse mode.
- In bilevel optimization, we suppose an optimal solutions set of lower level function; while in forward/inverse mode, we cosider the whole process of the lower level iterations.
- Calculating hypergradients in bilevel optimization invovles solving the lower level problem and two second-order derivatives, both are very heavy cost.
- Forward/inverse mode uses chain rule, just like for deep nets training.

Q & A