Paper: Self-Tuning Networks: Bilevel Optimization of Hyperparameters Using Structured Best-Response Functions [1]

Jiaxin Liu

Group Reading

August 10, 2021

Overview

Introduction

- w: parameters (e.g. weights and biases).
- λ: hyperparameters (e.g. weight decay).
- L_V: validation loss. L_T: training loss.
- Hyperparameter optimization $\rightarrow$ bilevel optimization as:

$$\lambda^* = \arg\min_{\lambda} = L_V(\lambda, w^*) \quad \text{subject to } w^* = \arg\min_{w} L_T(\lambda, w)$$

Introduction

- w: parameters (e.g. weights and biases).
- λ: hyperparameters (e.g. weight decay).
- L_V: validation loss. L_T: training loss.
- Hyperparameter optimization $\rightarrow$ bilevel optimization as:

$$\lambda^* = \arg\min_{\lambda} = L_V(\lambda, w^*) \quad \text{subject to } w^* = \arg\min_{w} L_T(\lambda, w)$$

- *Best-response function*: $w^*(\lambda) = \arg\min_w L_T(\lambda, w)$

$$\lambda^* = \arg\min_{\lambda} = L_V(\lambda, w^*(\lambda)) \rightarrow \text{Single-level}$$

Introduction

- w: parameters (e.g. weights and biases).
- λ: hyperparameters (e.g. weight decay).
- L_V: validation loss. L_T: training loss.
- Hyperparameter optimization $\rightarrow$ bilevel optimization as:

$$\lambda^* = \arg\min_{\lambda} = L_V(\lambda, w^*) \quad \text{subject to } w^* = \arg\min_{w} L_T(\lambda, w)$$

- *Best-response function*: $w^*(\lambda) = \arg\min_w L_T(\lambda, w)$

$$\lambda^* = \arg\min_{\lambda} = L_V(\lambda, w^*(\lambda)) \rightarrow \text{Single-level}$$

- Approximate the best-response w^*: function $\hat{w}_\phi(\lambda; \phi) \approx w^*$

$$\lambda^* = \arg\min_{\lambda} = L_V(\lambda, \hat{w}_\phi(\lambda)) \rightarrow \text{updating } \phi \text{ and} \lambda$$

Bilevel optimization Problem

$$\min_{\lambda \in \mathbb{R}^n} F(\lambda, w) \qquad \text{Upper-level}$$

$$\text{subject to } w \in \arg\min_{w \in \mathbb{R}^m} f(\lambda, w) \qquad \text{Lower-level}$$

- Lower-level
  - Evaluate on $f(\lambda, w) \Longleftrightarrow L_T(\lambda, w)$
  - Optimize parameters w with fixed hyperparameters λ.
  - w^*
- Upper-level
  - Based on w^*
  - Evaluate on $F(\lambda, w) \Longleftrightarrow L_V(\lambda, w^*)$
  - Optimize $\lambda \Longleftrightarrow$ hyperparameters.

Gradient descent (GD) via the best-response function

$$\min_{\lambda \in \mathbb{R}^n} F(\lambda, w) \qquad \text{Upper-level}$$

$$\text{subject to } w \in \arg\min_{w \in \mathbb{R}^m} f(\lambda, w) \qquad \text{Lower-level}$$

- Assume the lower-level problem has a unique optimum $w^*(\lambda)$ for each λ.

$$\min_{\lambda \in \mathbb{R}^n} F^*(\lambda) := F(\lambda, w^*(\lambda)) \qquad \text{Single-level}$$

- Can we minimize this problem using GD on $F^*(\lambda)$ w.r.t. λ?
  1. Differentiability on λ.
  2. There is a unique optimum $w^*(\lambda)$ for the lower-level problem for each λ.

Gradient descent (GD) via the best-response function

$$\min_{\lambda \in \mathbb{R}^n} F^*(\lambda) := F(\lambda, w^*(\lambda)) \qquad \text{Single-level}$$

We give sufficient conditions for the above two points to hold in a neighbourhood of a point (λ_0, w_0).

Lemma 1.

Let w_0 solve the lower-level problem for λ_0. Suppose f is C^2 in a neighborhood of (λ_0, w_0), and the Hessian $\partial^2 f / \partial w^2(\lambda_0, w_0)$ is positive definite. Then for some neighborhood U of λ_0, there exists a continuously differentiable function $w^* : U \to \mathbb{R}^m$ such that $w^*(\lambda)$ is the unique solution to lower-level problem for each each $\lambda \in U$ and $w^*(\lambda_0) = w_0$.

Gradient of F^*

$$\frac{\partial F^*}{\partial \lambda}(\lambda_0) = \frac{\partial F}{\partial \lambda}(\lambda_0, w^*(\lambda_0)) + \frac{\partial F}{\partial w}(\lambda_0, w^*(\lambda_0)) \frac{\partial w^*}{\partial \lambda}(\lambda_0)$$

Gradient descent of F^*

Gradient of F^*

$$\frac{\partial F^*}{\partial \lambda}(\lambda_0) = \frac{\partial F}{\partial \lambda}(\lambda_0, w^*(\lambda_0)) + \frac{\partial F}{\partial w}(\lambda_0, w^*(\lambda_0))\frac{\partial w^*}{\partial \lambda}(\lambda_0)$$

- $\frac{\partial F}{\partial \lambda}(\lambda_0, w^*(\lambda_0)) \to$ direct gradient.
- $\frac{\partial F}{\partial w}(\lambda_0, w^*(\lambda_0))\frac{\partial w^*}{\partial \lambda}(\lambda_0) \to$ response gradient.

Summary

1. Use best-response function $w^*(\lambda)$ to substitute w.
2. Assume two conditions hold in a neighborhood of a point (λ_0, w_0) where w_0 is the solution to the lower-level problem given λ_0.
3. Chain rule.

Question: how to get this $w^*(\lambda)$?

Approximate the best-response function

$$\frac{\partial F}{\partial w}(\lambda_0, w^*(\lambda_0))\frac{\partial w^*}{\partial \lambda}(\lambda_0) \qquad \text{response gradient}$$

- Approximate w^*
  - Global approximation
  - Local approximation
- Approximate $\frac{\partial w^*}{\partial \lambda}$

Global and local approximation

- Approximate the global best-response w^*: function $\hat{w}_\phi(\lambda) \approx w^*$
  - $\hat{w}_\phi$ is a hypernetwork.
  - Lower-level: $\min_\phi \mathbb{E}_{\lambda \sim p(\lambda)}[f(\lambda, \hat{w}_\phi(\lambda))]$
  - Upper-level: $\min_{\lambda \in \mathbb{R}^n} F(\lambda, \hat{w}_\phi(\lambda))$

Global and local approximation

- Approximate the global best-response w^*: function $\hat{w}_\phi(\lambda) \approx w^*$
  - $\hat{w}_\phi$ is a hypernetwork.
  - Lower-level: $\min_\phi \mathbb{E}_{\lambda \sim p(\lambda)}[f(\lambda, \hat{w}_\phi(\lambda))]$
  - Upper-level: $\min_{\lambda \in \mathbb{R}^n} F(\lambda, \hat{w}_\phi(\lambda))$
- Approximate the local best-response w^*: function $\hat{w}_\phi(\lambda) \approx w^*$
  - Approximate w^* in a neighborhood around the current λ.
  - Lower-level: $\min_\phi \mathbb{E}_{\varepsilon \sim p(\varepsilon|\sigma)}[f(\lambda + \varepsilon, \hat{w}_\phi(\lambda + \varepsilon))]$
  - $p(\varepsilon|\sigma)$ is a Gaussian noise distribution.
  - Perturb the upper-level λ and get the approximate $\hat{w}_\phi$.

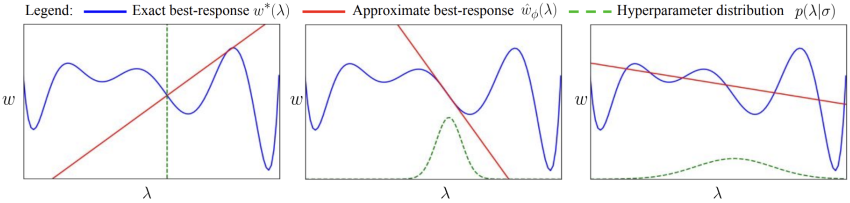


Figure: The effect of the sampled neighborhood. Left: fixed λ. Middle: proper sampled range. Right: large sampled range.

Self-Tuning Networks (STN)

1. How to construct the best-response approximation $\hat{w}_\phi$?
2. Automatically adjust the scale of the neighborhood when training the ϕ.

- Weight matrix: $W \in \mathbb{R}^{D_{out} \times D_{in}}$; bias vector: $b \in \mathbb{R}^{D_{out}}$.
- Hyperparameter $\lambda \in \mathbb{R}^n$
- Best-response for W and b:

$$\hat{W}_\phi(\lambda) = W_{elem} + (V\lambda) \odot_{row} W_{hyper} \qquad D_{out}(2D_{in} + n)$$

$$\hat{b}_\phi(\lambda) = b_{elem} + (C\lambda) \odot b_{hyper} \qquad D_{out}(2 + n)$$

Best-response for W and b:

$$\hat{W}_\phi(\lambda) = W_{elem} + (V\lambda) \odot_{row} W_{hyper} \qquad D_{out}(2D_{in}+n)$$

$$\hat{b}_\phi(\lambda) = b_{elem} + (C\lambda) \odot b_{hyper} \qquad D_{out}(2+n)$$

Collect the equations:

$$\hat{W}_\phi(\lambda)x + \hat{b}_\phi(\lambda) = [W_{elem}x + b_{elem}] + [(V\lambda) \odot (W_{hyper}x) + (C\lambda) \odot b_{hyper}]$$

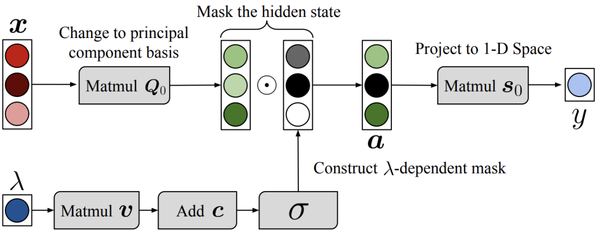


Figure: Best-response structure.

Example: construct the best-response for two-layer linear Networks.

- 2-layer NN: $w = (Q, s) \in \mathbb{R}^{D \times D} \times \mathbb{R}^{D}$
- input $x \in \mathbb{R}^{D}$, target $t \in \mathbb{R}$.
- Best-response function: $\hat{w}(\lambda) : \mathbb{R}^{n} \to \mathbb{R}^{m}$.
- First-layer:
- Second-layer:
- Loss:

$$L_T(\lambda, w) = \sum_{(x,t)\in D} (y(x; w) - t)^2 + \frac{1}{|D|} \exp(\lambda) \| \frac{\partial y}{\partial x}(x; w) \|^2$$

Optimal solution for $w^*(\lambda)$

Chose the $\sigma(\cdot)$ to get Q^*.

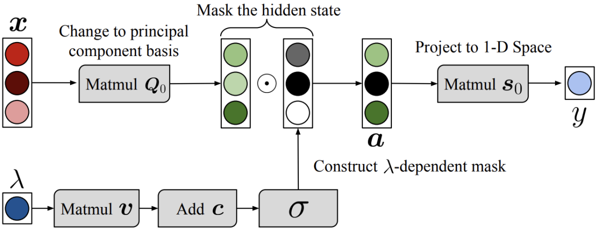
Exact best-response for the example.

Theorem 2.

Let $w_0 = (Q_0, s_0)$, where Q_0 is the change-of-basis matrix to the principal components of the data matrix and s_0 solves the regularized version of l_T in this example given $Q_)$. Then there exist $v, c \in \mathbb{R}^D$ such that the best-response function $w^*(\lambda) = (Q^*(\lambda), s^*(\lambda))$ is

$$Q^*(\lambda) = \sigma(\lambda v + c) \odot_{row} Q_0$$

$$s^*(\lambda) = s_0$$



Sampled neighborhood.

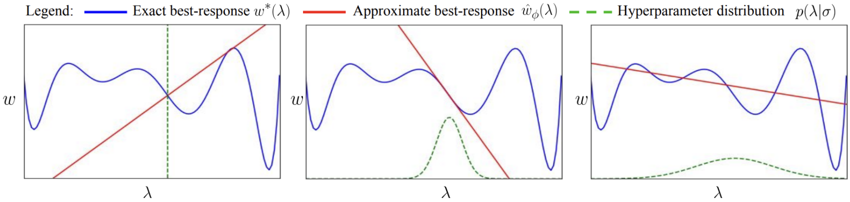


Figure: The effect of the sampled neighborhood. Left: fixed λ. Middle: proper sampled range. Right: large sampled range.

Include an entropy term for the upper-level optimization:

$$\mathbb{E}_{\varepsilon\sim p(\varepsilon|\sigma)}[F(\lambda+\varepsilon,\hat{w}_\phi(\lambda+\varepsilon))]-\tau\mathbb{H}[p(\varepsilon|\sigma)]$$

Algorithm

Algorithm 1 STN Training Algorithm

Initialize: Best-response approximation parameters ϕ, hyperparameters $\boldsymbol{\lambda}$, learning rates $\{\alpha_i\}_{i=1}^3$
while not converged **do**
  for $t = 1, \dots, T_{train}$ **do**
    $\boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon}|\boldsymbol{\sigma})$
    $\boldsymbol{\phi} \leftarrow \boldsymbol{\phi} - \alpha_1 \frac{\partial}{\partial \boldsymbol{\phi}} f(\boldsymbol{\lambda} + \boldsymbol{\epsilon}, \hat{\mathbf{w}}_{\boldsymbol{\phi}}(\boldsymbol{\lambda} + \boldsymbol{\epsilon}))$
  for $t = 1, \dots, T_{valid}$ **do**
    $\boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon}|\boldsymbol{\sigma})$
    $\boldsymbol{\lambda} \leftarrow \boldsymbol{\lambda} - \alpha_2 \frac{\partial}{\partial \boldsymbol{\lambda}} \left(F(\boldsymbol{\lambda} + \boldsymbol{\epsilon}, \hat{\mathbf{w}}_{\boldsymbol{\phi}}(\boldsymbol{\lambda} + \boldsymbol{\epsilon})) - \tau \mathbb{H}[p(\boldsymbol{\epsilon}|\boldsymbol{\sigma})]\right)$
    $\boldsymbol{\sigma} \leftarrow \boldsymbol{\sigma} - \alpha_3 \frac{\partial}{\partial \boldsymbol{\sigma}} \left(F(\boldsymbol{\lambda} + \boldsymbol{\epsilon}, \hat{\mathbf{w}}_{\boldsymbol{\phi}}(\boldsymbol{\lambda} + \boldsymbol{\epsilon})) - \tau \mathbb{H}[p(\boldsymbol{\epsilon}|\boldsymbol{\sigma})]\right)$

Figure: STN training algorithm.

Drop out

Method	Val	Test
$p = 0.68$, Fixed	85.83	83.19
$p = 0.68$ w/ Gaussian Noise	85.87	82.29
$p = 0.68$ w/ Sinusoid Noise	85.29	82.15
$p = 0.78$ (Final STN Value)	89.65	86.90
STN	**82.58**	**79.02**

Figure: Validation and test perplexity with different dropout settings.

Drop out

	PTB		CIFAR-10	
Method	**Val Perplexity**	**Test Perplexity**	**Val Loss**	**Test Loss**
Grid Search	97.32	94.58	0.794	0.809
Random Search	84.81	81.46	0.921	0.752
Bayesian Optimization	72.13	69.29	0.636	0.651
STN	**70.30**	**67.68**	**0.575**	**0.576**

Figure: Validation and test perplexity for different methods.