Gradient-based hyperparameter optimization

MIPT

2023

Model selection: coherent inference

First level: select optimal parameters:

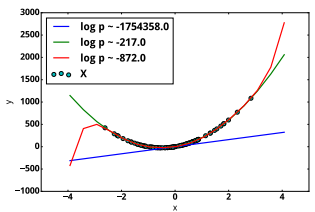
$$\mathbf{w} = \arg\max \frac{p(\mathfrak{D}|\mathbf{w})p(\mathbf{w}|\mathbf{h})}{p(\mathfrak{D}|\mathbf{h})},$$

Second level: select optimal model (hyperparameters).

Evidence:

$$p(\mathfrak{D}|\mathbf{h}) = \int_{\mathbf{w}} p(\mathfrak{D}|\mathbf{w})p(\mathbf{w}|\mathbf{h})d\mathbf{w}.$$

Model selection scheme



Example: polynoms

Hyperparameters

Definition

Prior for parameters $\mathbf{w}$ and structure $\boldsymbol{\Gamma}$ of the model $\mathbf{f}$ is a distrubution $p(\mathbf{W}, \boldsymbol{\Gamma}|\mathbf{h}) : \mathbb{W} \times \Gamma \times \mathbb{H} \to \mathbb{R}^+$, where $\mathbb{W}$ is a parameter space, Γ is a structure space.

Definition

Hyperparameters $\mathbf{h} \in \mathbb{H}$ of the models are the parameters of $p(\mathbf{w}, \boldsymbol{\Gamma}|\mathbf{h})$ (parameters of prior $\mathbf{f}$).

Problem statement

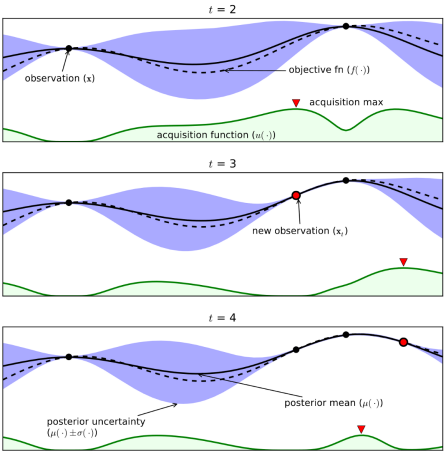
Let $\boldsymbol{\theta} \in \mathbb{R}^s$ be the set of all the optimized parameters (including variational parameters if needed).
$L(\boldsymbol{\theta}, \mathbf{h})$ is a differential loss function $\mathbf{f}$.
$Q(\boldsymbol{\theta}, \mathbf{h})$ is a differential validation function.
The problem is to find optimal parameters $\boldsymbol{\theta}^*$ and hyperparameters $\mathbf{h}^*$ of the model that minimze

$$\mathbf{h}^* = \arg\min \mathbf{h} \in \mathbb{H} Q(\boldsymbol{\theta}^*(\mathbf{h}), \mathbf{h}),$$

$$\boldsymbol{\theta}(\mathbf{h})^* = \arg\min_{\boldsymbol{\theta} \in \mathbb{R}^s} L(\boldsymbol{\theta}, \mathbf{h}).$$

Non-gradient based methods: cons

- Need to run multiple times to get optimal hyperparameter values
- Computational complexity:
  - GP: $O(m^3)$, m is the number of observations.
  - TPE: $O(m \log m)$



Shahriari et. al, 2016. GP example.

Bengio, 1999

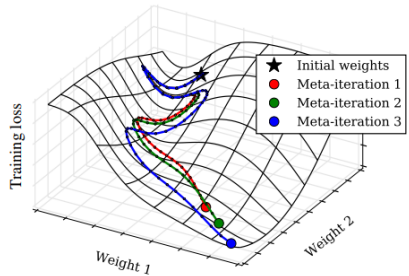
- Consider a quadratic loss: $Q = \mathbf{a}(\mathbf{h}) + \mathbf{b}(\mathbf{h})\boldsymbol{\theta} + \boldsymbol{\theta}^{\mathsf{T}}\mathbf{H}(\mathbf{h})\boldsymbol{\theta}$
- The basic equation: $\mathbf{b}(\mathbf{h}) + \mathbf{H}(\mathbf{h})\boldsymbol{\theta} = 0 \quad \boldsymbol{\theta}(\mathbf{h}) = -\mathbf{H}^{-1}(\mathbf{h})\mathbf{b}(\mathbf{h})$.
- Gradient w.r.t. to $\mathbf{h}$ can be calculated differently, the most effective way: using backpropagation, $O(s^3)$ (not very effective, actually).

Gradient methods

Idea: Optimize hyperparameters using the full parameter optimization trajectory.
Pros:

- Hyperparameter optimization will consider the features of the parameter optimization.
- The complexity is linear on the number of hyperparameters.

Cons: the computational cost is very expensive. Also can be problems with numerical precision of reverse method.



Maclaurin et. al, 2015. Example.

Problem statement: gradient-based optimization

Definition

A SGD operator T is an operator providing η steps of optimization:

$$\hat{\boldsymbol{\theta}} = T \circ T \circ \cdots \circ T(\boldsymbol{\theta}_0, \mathbf{h}) = T^{\eta}(\boldsymbol{\theta}_0, \mathbf{h}), \tag{1}$$

where

$$T(\boldsymbol{\theta}, \mathbf{h}) = \boldsymbol{\theta} - \lambda \nabla L(\boldsymbol{\theta}, \mathbf{h})|_{\hat{\mathfrak{D}}},$$

λ is a learning rate, $\boldsymbol{\theta}_0$ is an initial value of $\boldsymbol{\theta}$, $\hat{\mathfrak{D}}$ is a random subset of $\mathfrak{D}$.

Rewrite the optimization problem:

$$\mathbf{h}^* = \arg\max_{\mathbf{h} \in \mathbb{H}} Q(T^{\eta}(\boldsymbol{\theta}_0, \mathbf{h})).$$

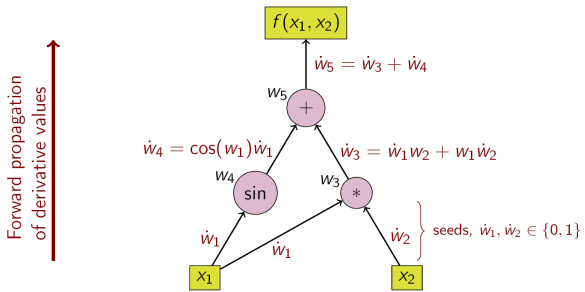
Forward-mode differentiation

Main idea:

$$\frac{\partial y}{\partial x} = \frac{\partial y}{\partial w_{n-1}} \frac{\partial w_{n-1}}{\partial x} = \frac{\partial y}{\partial w_{n-1}} \left(\frac{\partial w_{n-1}}{\partial w_{n-2}} \frac{\partial w_{n-2}}{x} \partial x \right) = \dots .$$

Example (wiki):

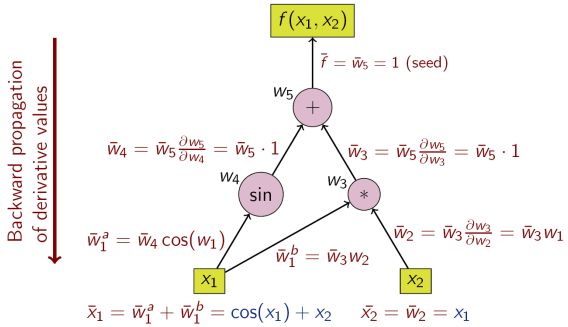
$$x_1 x_2 + \sin(x_1)$$



Reverse-mode differentiation

Idea:

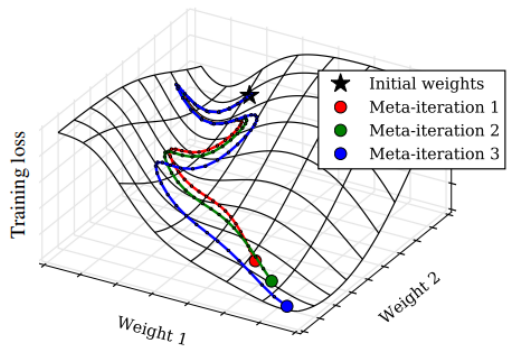
$$\frac{\partial y}{\partial x} = \frac{\partial y}{\partial w_1}\frac{\partial w_1}{\partial x} = \left(\frac{\partial y}{\partial w_2}\frac{\partial w_2}{\partial w_1}\right)\frac{\partial w_1}{\partial x} = \dots.$$



RMAD, Maclaurin et. al, 2015

1. Run η steps of memontum optimization γ: $\boldsymbol{\theta} = T(\boldsymbol{\theta}_0, \mathbf{h})$.
2. Let $\hat{\nabla}\mathbf{h} = \nabla_{\mathbf{h}} Q(\boldsymbol{\theta}, \mathbf{h})$.
3. Let $d\mathbf{v} = \mathbf{0}$.
4. For $\tau = \eta \dots 1$ repeat:
5. calucalate $\boldsymbol{\theta}^{\tau-1}$.
6. Calculate gradient on the step $\tau - 1$, using RMD.

RMAD algorithm is based on the Reverse-mode differentiation.



DrMAD

DrMad is a simplified RMAD. Idea: the parameter optimization trajectory is linear.

1. Run η steps of memontum optimization γ: $\boldsymbol{\theta} = T(\boldsymbol{\theta}_0, \mathbf{h})$.
2. Let $\hat{\nabla}\mathbf{h} = \nabla_{\mathbf{h}} Q(\boldsymbol{\theta}, \mathbf{h})$.
3. Let $d\mathbf{v} = \mathbf{0}$.
4. For $\tau = \eta \dots 1$ repeat:
5. Calculate $\boldsymbol{\theta}^{\tau-1}$.
6. Calculate gradient on the step $\tau - 1$, using RMD.

1. Run η steps of memontum optimization γ: $\boldsymbol{\theta} = T(\boldsymbol{\theta}_0, \mathbf{h})$.
2. Let $\hat{\nabla}\mathbf{h} = \nabla_{\mathbf{h}} Q(\boldsymbol{\theta}, \mathbf{h})$.
3. Let $d\mathbf{v} = \mathbf{0}$.
4. For $\tau = \eta \dots 1$ repeat:
5. $\boldsymbol{\theta}^{\tau-1} = \boldsymbol{\theta}_0 + \frac{\tau-1}{\eta}\boldsymbol{\theta}^{\eta}$.
6. Calculate gradient on the step $\tau - 1$, using RMD.

Optimization closed form

Statement (Pedregosa, 2016)

Let L be a differential function such that all the stationary points of L are global minima. Let the Hessian $\mathbf{H}$ of L be invertible in each stationary point.
Then

$$\nabla_{\mathbf{h}} Q(T(\boldsymbol{\theta}_0, \mathbf{h}), \mathbf{h}) = \nabla_{\mathbf{h}} Q(\boldsymbol{\theta}^\eta, \mathbf{h}) - \nabla_{\mathbf{h}} \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}^\eta, \mathbf{h})^\mathsf{T} \mathbf{H}^{-1} \nabla_{\boldsymbol{\theta}} Q(\boldsymbol{\theta}^\eta, \mathbf{h}).$$

Proof sketch

1. Since $\boldsymbol{\theta}^\eta$ is stationary, then $\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}^\eta, \mathbf{h}) = 0$.
2. Take a derviative w.r.t. $\mathbf{h}$:

$$\nabla_{\mathbf{h}} \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}^\eta, \mathbf{h}) + \mathbf{H} \nabla_{\mathbf{h}} T(\boldsymbol{\theta}_0, \mathbf{h}) = 0.$$

3. Using chain rule:

$$\nabla_{\mathbf{h}} Q(T(\boldsymbol{\theta}_0), \mathbf{h}) = \nabla_{\mathbf{h}} Q(\boldsymbol{\theta}^\eta, \mathbf{h}) + \nabla_{\mathbf{h}} T(\boldsymbol{\theta}_0, \mathbf{h})^\mathsf{T} \nabla_{\boldsymbol{\theta}} Q(\boldsymbol{\theta}^\eta, \mathbf{h}).$$

4. Move expression from 3 to 2 and get result.

Greedy optimization of hyperparameters

On each step of $\boldsymbol{\theta}$:

$$\mathbf{h}' = \mathbf{h} - \lambda_{\mathbf{h}} \nabla_{\mathbf{h}} Q\big(T^{\eta}(\boldsymbol{\theta}, \mathbf{h}), \mathbf{h}\big) \approx \mathbf{h} - \lambda_{\mathbf{h}} \nabla_{\mathbf{h}} Q\big(T(\boldsymbol{\theta}^{\eta-1}, \mathbf{h}), \mathbf{h}\big) =$$

$$= \mathbf{h} - \lambda_{\mathbf{h}} \nabla_{\mathbf{h}} Q\big(\boldsymbol{\theta} - \lambda \nabla L(\boldsymbol{\theta}, \mathbf{h}), \mathbf{h}\big)\big) = \mathbf{h} - \lambda_{\boldsymbol{\theta}} \nabla_{\mathbf{h}} Q\big(\boldsymbol{\theta} - \lambda \nabla L(\boldsymbol{\theta}, \mathbf{h}), \mathbf{h}\big)\big) \nabla_{\mathbf{h}} \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}^{\eta}, \mathbf{h}),$$

where $\lambda_{\mathbf{h}}$ is a learning rate of hyperparameter optimization.

- The method can be considered as a simplified RMAD that uses only one step of parameter optimization.
- The method can be considered as an approximation of the closed form expression for $\mathbf{H}^{-1} \sim \mathbf{I}$ when $\boldsymbol{\theta}$ is near optimum.
- Complexity: $O(|\boldsymbol{\theta}| \cdot |\mathbf{h}|)$
- Can be simplified using finite differences, see DARTS, $O(|\boldsymbol{\theta}| + |\mathbf{h}|)$

HOAG

Approximation of closed-form formula:

$$\nabla_{\mathbf{h}} Q(\boldsymbol{\theta}^{\eta}, \mathbf{h}) - \nabla_{\mathbf{h}} \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}^{\eta}, \mathbf{h})^{\mathsf{T}} \mathbf{H}^{-1} \nabla_{\boldsymbol{\theta}} Q(\boldsymbol{\theta}^{\eta}, \mathbf{h}).$$

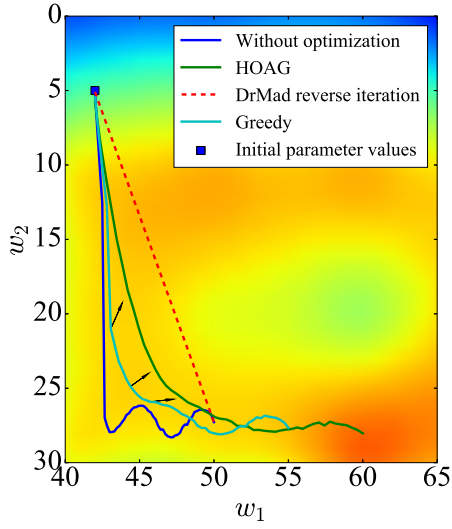
1. run η optimization steps: $\boldsymbol{\theta} = T(\boldsymbol{\theta}_0, \mathbf{h})$.
2. Solve linear system for $\boldsymbol{\lambda}$: $\mathbf{H}(\boldsymbol{\theta})\boldsymbol{\lambda} = \nabla_{\boldsymbol{\theta}} Q(\boldsymbol{\theta}, \mathbf{h})$.
3. Calculate approximate value of hyperparameter gradients $\hat{\nabla}_{\mathbf{h}} Q = \nabla_{\mathbf{h}} Q(\boldsymbol{\theta}, \mathbf{h}) - \nabla_{\boldsymbol{\theta},\mathbf{h}} L(\boldsymbol{\theta}, \mathbf{h})^{T} \boldsymbol{\lambda}$.

Update formula:

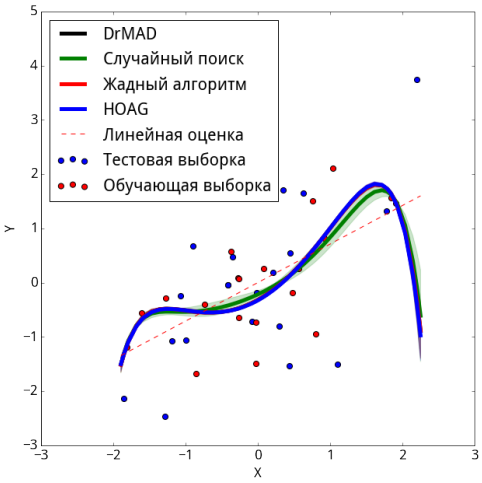
$$\mathbf{h}' = \mathbf{h} - \gamma_{\mathbf{h}} \hat{\nabla}_{\mathbf{h}} Q.$$

Algorithm comparison

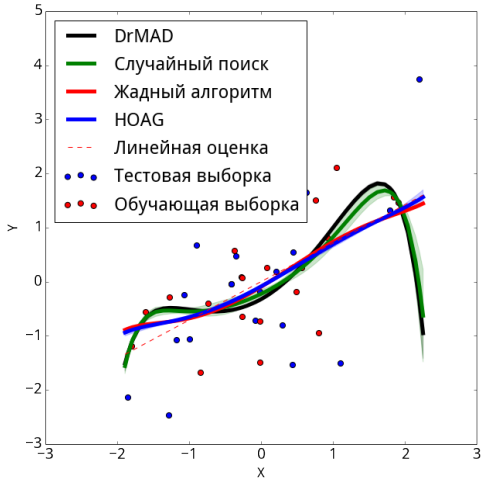
Algorithm	+	-
Random search	Easy to implement	Curse of dimmensionality
Greedy optimization	Easy to implement, can be run inside common gradient optimization loop	Non-optimality.
HOAG	Fast convergence.	Results quality depends on the the linear system solution precision $\mathbf{H}(\boldsymbol{\theta})\boldsymbol{\lambda} = \nabla_{\boldsymbol{\theta}} Q(\boldsymbol{\theta}, \mathbf{h})$.
DrMAD	Considers optimization features. Can be used for metaparameter optimization.	Not very tobust. Strong assumptions about linearity of the optimization trajectory.


Without optimization
HOAG
DrMad reverse iteration
Greedy
Initial parameter values
0
5
10
15
20
25
30
40
45
50
55
60
65
w_1
w_2

Experiment: polynoms

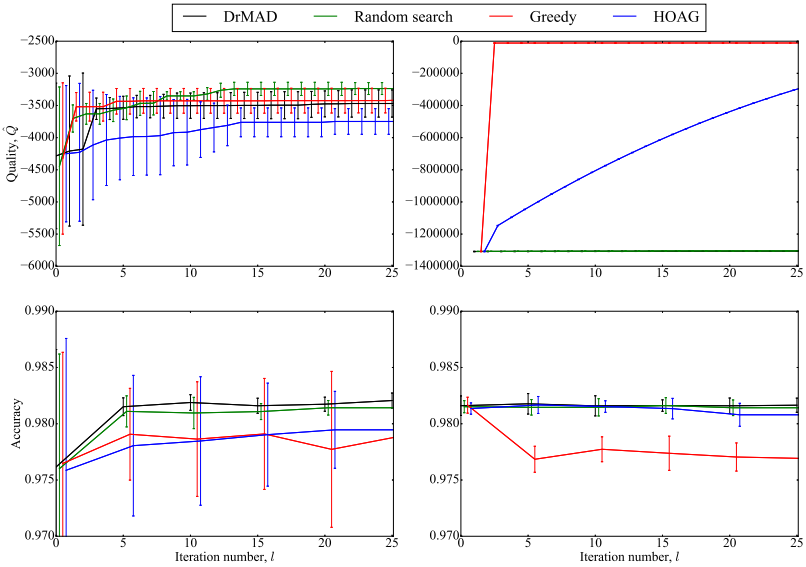


CV



Evidence

Experiment: MNIST



Experiment: MNIST

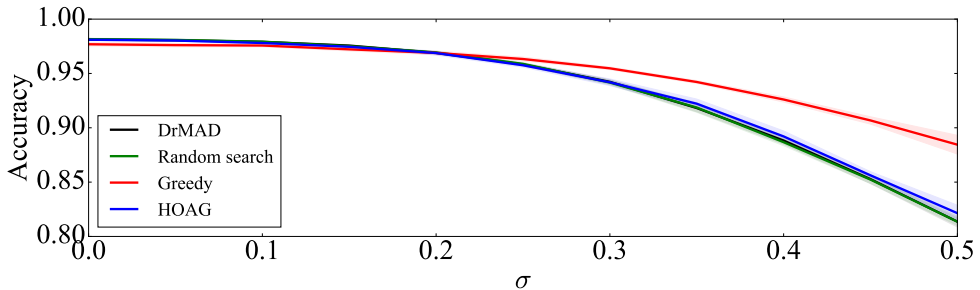
Adding Gaussian noise $\mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$:

Без шума

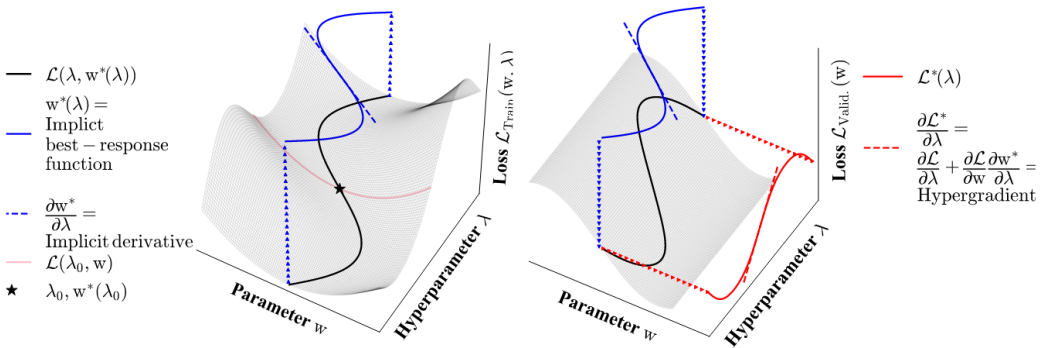
$\sigma = 0.1$

$\sigma = 0.25$

$\sigma = 0.5$



Optimizing Millions of Hyperparameters by Implicit Differentiation



See Victor Pankratov's talk, 2021

Optimizing Millions of Hyperparameters by Implicit Differentiation

Theorem 1 (Cauchy, Implicit Function Theorem). *If for some* $(\boldsymbol{\lambda}', \mathbf{w}')$, $\frac{\partial \mathcal{L}_{\mathrm{T}}}{\partial \mathbf{w}}|_{\boldsymbol{\lambda}', \mathbf{w}'} = 0$ *and regularity conditions are satisfied, then surrounding* $(\boldsymbol{\lambda}', \mathbf{w}')$ *there is a function* $\mathbf{w}^*(\boldsymbol{\lambda})$ *s.t.* $\frac{\partial \mathcal{L}_{\mathrm{T}}}{\partial \mathbf{w}}|_{\boldsymbol{\lambda}, \mathbf{w}^*(\boldsymbol{\lambda})} = 0$ *and we have:*

$$\frac{\partial \mathbf{w}^*}{\partial \boldsymbol{\lambda}}\Big|_{\boldsymbol{\lambda}'} = -\Big[\underbrace{\frac{\partial^2 \mathcal{L}_{\mathrm{T}}}{\partial \mathbf{w} \partial \mathbf{w}}}_{\text{training Hessian}}\Big]^{-1} \times \underbrace{\frac{\partial^2 \mathcal{L}_{\mathrm{T}}}{\partial \mathbf{w} \partial \boldsymbol{\lambda}}}_{\text{training mixed partials}}\Big|_{\boldsymbol{\lambda}', \mathbf{w}^*(\boldsymbol{\lambda}')} \quad \text{(IFT)}$$

$$\left[\frac{\partial^2 \mathcal{L}_{\mathrm{T}}}{\partial \mathbf{w} \partial \mathbf{w}}\right]^{-1} = \lim_{i \to \infty} \sum_{j=0}^{i} \left[I - \frac{\partial^2 \mathcal{L}_{\mathrm{T}}}{\partial \mathbf{w} \partial \mathbf{w}}\right]^j$$

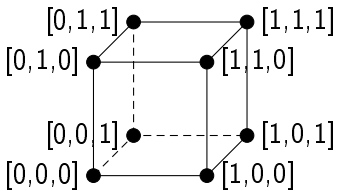
Optimizing Millions of Hyperparameters by Implicit Differentiation

Jonathan Lorraine, Paul Vicol, David Duvenaud

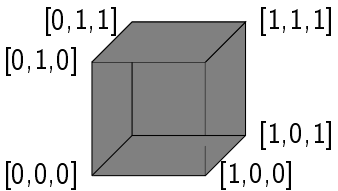
Method	Steps	Eval.	Hypergradient Approximation
Exact IFT	∞	$\mathbf{w}^*(\boldsymbol{\lambda})$	$\frac{\partial \mathcal{L}_V}{\partial \boldsymbol{\lambda}} - \frac{\partial \mathcal{L}_V}{\partial \mathbf{w}} \times \left[\frac{\partial^2 \mathcal{L}_T}{\partial \mathbf{w} \partial \mathbf{w}}\right]^{-1} \frac{\partial^2 \mathcal{L}_T}{\partial \mathbf{w} \partial \boldsymbol{\lambda}}\Big\vert_{\mathbf{w}^*(\boldsymbol{\lambda})}$
Unrolled Diff. [Maclaurin et al., 2015]	i	$\mathbf{w}_0$	$\frac{\partial \mathcal{L}_V}{\partial \boldsymbol{\lambda}} - \frac{\partial \mathcal{L}_V}{\partial \mathbf{w}} \times \sum_{j \leq i} \left[\prod_{k<j} I - \frac{\partial^2 \mathcal{L}_T}{\partial \mathbf{w} \partial \mathbf{w}}\Big\vert_{\mathbf{w}_{i-k}}\right] \frac{\partial^2 \mathcal{L}_T}{\partial \mathbf{w} \partial \boldsymbol{\lambda}}\Big\vert_{\mathbf{w}_{i-j}}$
L-Step Truncated Unrolled Diff.	i	$\mathbf{w}_L$	$\frac{\partial \mathcal{L}_V}{\partial \boldsymbol{\lambda}} - \frac{\partial \mathcal{L}_V}{\partial \mathbf{w}} \times \sum_{L \leq j \leq i} \left[\prod_{k<j} I - \frac{\partial^2 \mathcal{L}_T}{\partial \mathbf{w} \partial \mathbf{w}}\Big\vert_{\mathbf{w}_{i-k}}\right] \frac{\partial^2 \mathcal{L}_T}{\partial \mathbf{w} \partial \boldsymbol{\lambda}}\Big\vert_{\mathbf{w}_{i-j}}$
Larsen et al. [1996]	∞	$\widehat{\mathbf{w}^*}(\boldsymbol{\lambda})$	$\frac{\partial \mathcal{L}_V}{\partial \boldsymbol{\lambda}} - \frac{\partial \mathcal{L}_V}{\partial \mathbf{w}} \times \left[\frac{\partial \mathcal{L}_T}{\partial \mathbf{w}} \frac{\partial \mathcal{L}_T}{\partial \mathbf{w}}^T\right]^{-1} \frac{\partial^2 \mathcal{L}_T}{\partial \mathbf{w} \partial \boldsymbol{\lambda}}\Big\vert_{\widehat{\mathbf{w}^*}(\boldsymbol{\lambda})}$
Bengio [2000]	∞	$\widehat{\mathbf{w}^*}(\boldsymbol{\lambda})$	$\frac{\partial \mathcal{L}_V}{\partial \boldsymbol{\lambda}} - \frac{\partial \mathcal{L}_V}{\partial \mathbf{w}} \times \left[\frac{\partial^2 \mathcal{L}_T}{\partial \mathbf{w} \partial \mathbf{w}}\right]^{-1} \frac{\partial^2 \mathcal{L}_T}{\partial \mathbf{w} \partial \boldsymbol{\lambda}}\Big\vert_{\widehat{\mathbf{w}^*}(\boldsymbol{\lambda})}$
$T1-T2$ [27]	1	$\widehat{\mathbf{w}^*}(\boldsymbol{\lambda})$	$\frac{\partial \mathcal{L}_V}{\partial \boldsymbol{\lambda}} - \frac{\partial \mathcal{L}_V}{\partial \mathbf{w}} \times [I]^{-1} \frac{\partial^2 \mathcal{L}_T}{\partial \mathbf{w} \partial \boldsymbol{\lambda}}\Big\vert_{\widehat{\mathbf{w}^*}(\boldsymbol{\lambda})}$
Ours	i	$\widehat{\mathbf{w}^*}(\boldsymbol{\lambda})$	$\frac{\partial \mathcal{L}_V}{\partial \boldsymbol{\lambda}} - \frac{\partial \mathcal{L}_V}{\partial \mathbf{w}} \times \left(\sum_{j \leq i} \left[I - \frac{\partial^2 \mathcal{L}_T}{\partial \mathbf{w} \partial \mathbf{w}}\right]^j\right) \frac{\partial^2 \mathcal{L}_T}{\partial \mathbf{w} \partial \boldsymbol{\lambda}}\Big\vert_{\widehat{\mathbf{w}^*}(\boldsymbol{\lambda})}$
Conjugate Gradient (CG) $\approx$	-	$\widehat{\mathbf{w}^*}(\boldsymbol{\lambda})$	$\frac{\partial \mathcal{L}_V}{\partial \boldsymbol{\lambda}} - \left(\arg\min_{\mathbf{x}} \Vert \mathbf{x} \frac{\partial^2 \mathcal{L}_T}{\partial \mathbf{w} \partial \mathbf{w}} - \frac{\partial \mathcal{L}_V}{\partial \mathbf{w}} \Vert\right) \frac{\partial^2 \mathcal{L}_T}{\partial \mathbf{w} \partial \boldsymbol{\lambda}}\Big\vert_{\widehat{\mathbf{w}^*}(\boldsymbol{\lambda})}$
Hypernetwork	-	-	$\frac{\partial \mathcal{L}_V}{\partial \boldsymbol{\lambda}} + \frac{\partial \mathcal{L}_V}{\partial \mathbf{w}} \times \frac{\partial \mathbf{w}_\phi^*}{\partial \boldsymbol{\lambda}}$ where $\mathbf{w}_\phi^*(\boldsymbol{\lambda}) = \arg\min_\phi \mathcal{L}_T(\boldsymbol{\lambda}, \mathbf{w}_\phi(\boldsymbol{\lambda}))$
Bayesian Optimization	-	-	$\frac{\partial \mathbb{E}[\mathcal{L}_V^*]}{\partial \boldsymbol{\lambda}}$ where $\mathcal{L}_V^* \sim \text{Gaussian-Process}(\{\boldsymbol{\lambda}_i, \mathcal{L}_V(\boldsymbol{\lambda}_i, \mathbf{w}^*(\boldsymbol{\lambda}_i))\})$

Discrete parameter relaxation

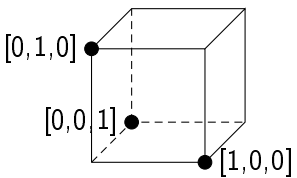
An example of restrictions for structure parameter γ, $|\gamma| = 3$.



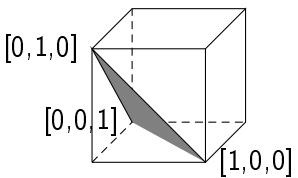
Cube vertices



Cube interior

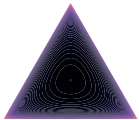


Simplex vertices



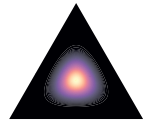
Simplex interior

Discrete distribution: relaxation

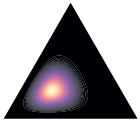


$\bar{\boldsymbol{\alpha}} = [1, 1, 1], t = 0.9$

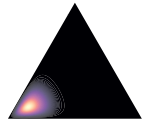
$t = 1.0$



$t = 10.0$



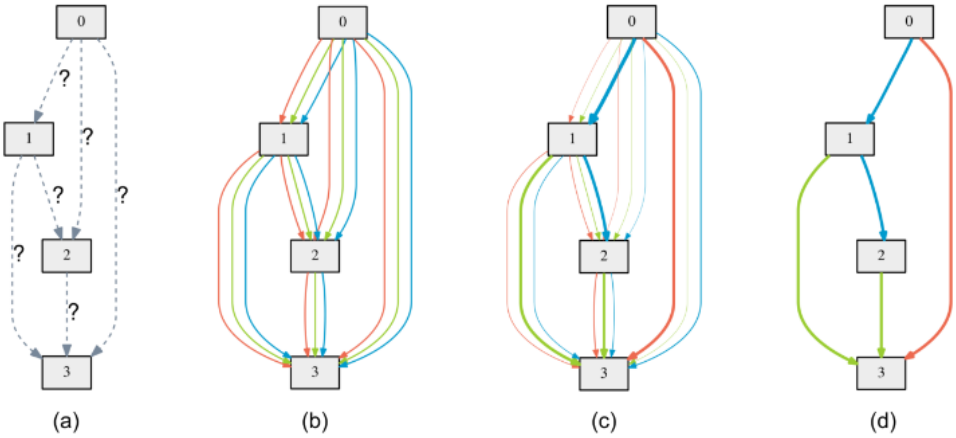
$\bar{\boldsymbol{\alpha}} = [0.5, 0.25, 0.25], t = 30.0$



$[0.75, 0.125, 0.125]$

$[0.9, 0.05, 0.05]$

Application example: DARTS



Application example: DARTS

Applying chain rule to the approximate architecture gradient (equation 6) yields

$$\nabla_\alpha \mathcal{L}_{val}(w', \alpha) - \xi \nabla^2_{\alpha,w} \mathcal{L}_{train}(w, \alpha) \nabla_{w'} \mathcal{L}_{val}(w', \alpha) \tag{7}$$

where $w' = w - \xi \nabla_w \mathcal{L}_{train}(w, \alpha)$ denotes the weights for a one-step forward model. The expression above contains an expensive matrix-vector product in its second term. Fortunately, the complexity can be substantially reduced using the finite difference approximation. Let ϵ be a small scalar[2] and $w^\pm = w \pm \epsilon \nabla_{w'} \mathcal{L}_{val}(w', \alpha)$. Then:

$$\nabla^2_{\alpha,w} \mathcal{L}_{train}(w, \alpha) \nabla_{w'} \mathcal{L}_{val}(w', \alpha) \approx \frac{\nabla_\alpha \mathcal{L}_{train}(w^+, \alpha) - \nabla_\alpha \mathcal{L}_{train}(w^-, \alpha)}{2\epsilon} \tag{8}$$

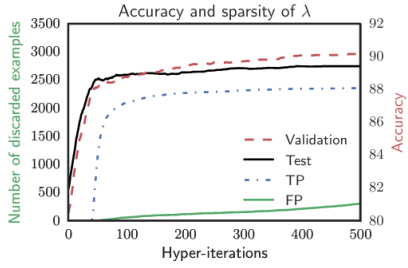
Evaluating the finite difference requires only two forward passes for the weights and two backward passes for α, and the complexity is reduced from $O(|\alpha||w|)$ to $O(|\alpha| + |w|)$.

First-order Approximation When $\xi = 0$, the second-order derivative in equation 7 will disappear. In this case, the architecture gradient is given by $\nabla_\alpha \mathcal{L}_{val}(w, \alpha)$, corresponding to the simple heuristic of optimizing the validation loss by assuming the current w is the same as $w^*(\alpha)$. This leads to some speed-up but empirically worse performance, according to our experimental results in Table 1 and Table 2. In the following, we refer to the case of $\xi = 0$ as the first-order approximation, and refer to the gradient formulation with $\xi > 0$ as the second-order approximation.

Data Hyper-cleaning

- Toy but illustrative example
- Given a very noisy training dataset with cleaning validation + test part, the task is to discard noisy examples from training part using sparsity constraints.
- Original experiment: MNIST, 5000 objects per train/validation/test, 2500 objects are corrupted.

	Accuracy %	F_1
Oracle	90.46	1.0000
Baseline	87.74	-
DH-1000	90.07	0.9137
DH-1500	90.06	0.9244
DH-2000	90.00	0.9211
DH-2500	90.09	0.9217



References

- Bengio, Yoshua. "Gradient-based optimization of hyperparameters."Neural computation 12.8 (2000): 1889-1900.
- Bishop C. M. Pattern recognition //Machine learning. – 2006. – Т. 128. – №. 9.
- Bakhteev O. Y., Strijov V. V. Comprehensive analysis of gradient-based hyperparameter optimization algorithms //Annals of Operations Research. – 2020. – Т. 289. – №. 1. – С. 51-65.
- Graves A. Practical variational inference for neural networks //Advances in neural information processing systems. – 2011. – Т. 24.
- Bergstra et al., Random Search for Hyper-Parameter Optimization, 2012
- Dougal Maclaurin et. al, Gradient-based Hyperparameter Optimization through Reversible Learning, 2015
- Jelena Luketina et. al, Scalable Gradient-Based Tuning of Continuous Regularization Hyperparameters, 2016
- Jie Fu et. al, DrMAD: Distilling Reverse-Mode Automatic Differentiation for Optimizing Hyperparameters of Deep Neural Networks, 2016
- Fabian Pedregosa, Hyperparameter optimization with approximate gradient, 2016
- Bobak Shahriari et. al, Taking the Human Out of the Loop: A Review of Bayesian Optimization, 2016
- Liu H., Simonyan K., Yang Y. Darts: Differentiable architecture search //arXiv preprint arXiv:1806.09055. – 2018.
- Lorraine, J., Vicol, P., & Duvenaud, D. (2020, June). Optimizing millions of hyperparameters by implicit differentiation. In International Conference on Artificial Intelligence and Statistics (pp. 1540-1552). PMLR. (see also Victor Pankratov's slides, BMM-2021)
- Franceschi, Luca, et al. "Forward and reverse gradient-based hyperparameter optimization."Interr