Real-time Hyperparameter Optimization

Luca Franceschi[1,2], Michele Donini[1], Riccardo Grazzi[3], Paolo Frasconi[3], Massimiliano Pontil[1,2]

(1) Istituto Italiano di Tecnologia, Genoa, I (2) University College London, London, UK
(3) Università degli Studi di Firenze, I

Overview

The gradient of a validation error with respect to real-valued hyperparameters can be computed with two different procedures (reverse-mode and forward-mode) which have different trade-off in terms of running time and space requirements. Both procedures are suitable for **real-time updates**, which speed up significantly hyperparameter optimization (HO) on large models such as deep neural networks. We show applications of these novel gradient-based HO procedures in different scenarios.

Hyperparameter optimization

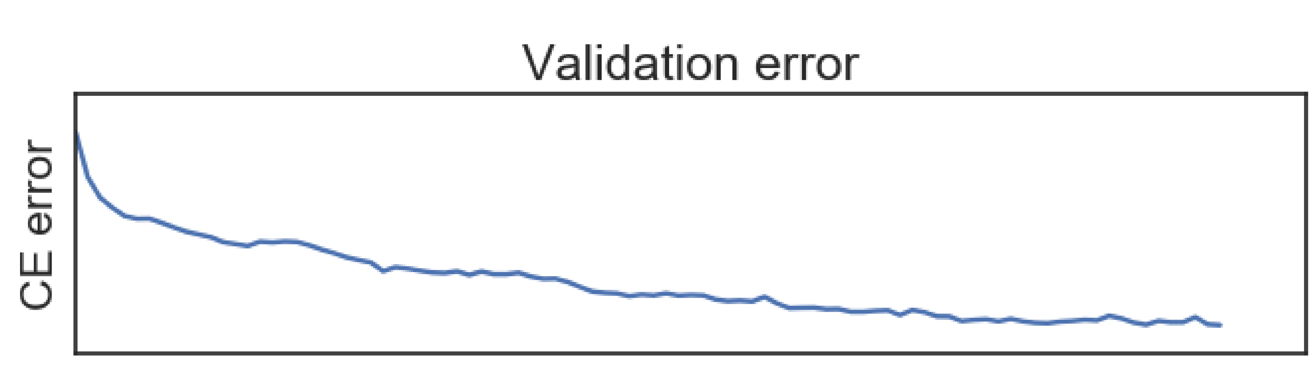
$$\min_{\lambda \in \Lambda} E^{val}(w^*(\lambda)) \qquad (1)$$

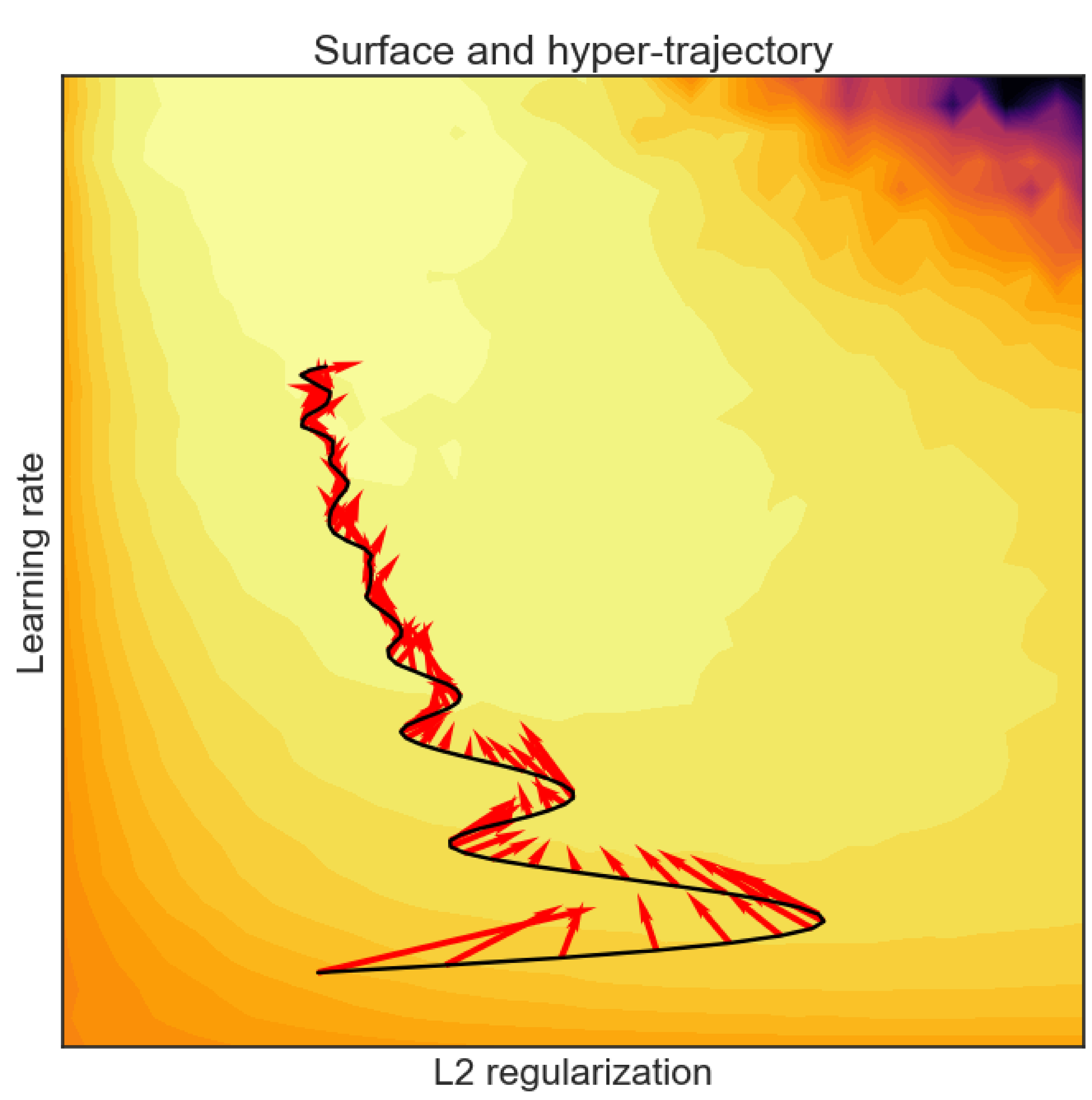
$$\text{s.t. } w^*(\lambda) \in \arg\min_{w} E^{train}(w, \lambda) \qquad (2)$$

- **Aims:** Automate ML, increase generalization, allow "freer" design of models and problems
- **Difficulties:** Cost of evaluating E^{val}, structure of Λ
- **Approaches:** Manual, grid, random search; Bayesian optimization; gradient-based optimization

Gradient-based HO

Other HO methods consider E^{val} as a black-box function. Gradient-based HO **opens the box** by explicitly considering the algorithm (training dynamics) used to solve (2).





- **Requirements:** Differentiable parametric model and iterative training dynamics; real-valued λ
- Two **algorithms** to compute ∇E^{val}, linked to *RNN training* and *algorithmic differentiation*: `Forward-HG` (memory efficient, suitable when $\dim(w)$ is large and $\dim(\lambda)$ is small) and `Reverse-HG` (time efficient, suitable when $\dim(\lambda)$ is large).

Algorithms

- State: s ($\sim$ weights of the model)
- Hyperparameter vector: λ
- Training dynamics: $s_t = \Phi_t(s_{t-1}, \lambda)$
- $A_t = \frac{\partial \Phi_t(s_{t-1},\lambda)}{\partial s_{t-1}}$, $\quad B_t = \frac{\partial \Phi_t(s_{t-1},\lambda)}{\partial \lambda}$
- Choose *hyper-batch* size Δ

Algorithm 1 RTHO, based on Forward-HG

$Z_0 \leftarrow 0$
for $t = 1$ **to** $\ldots$ **do**
  $Z_t \leftarrow A_t Z_{t-1} + B_t$
  $s_t \leftarrow \Phi_t(s_{t-1}, \lambda)$
  if $t \equiv 0 \ (\bmod\ \Delta)$ **then**
    $\lambda \leftarrow \lambda - \eta \nabla E^{val}(s_t) Z_t$
  end if
end for
return s

Algorithm 2 Truncated-reverse HO, based on Reverse-HG

for $k = 1$ **to** $\ldots$ **do**
  for $t = 1 + (k-1)\Delta$ **to** $k\Delta$ **do**
    $s_t \leftarrow \Phi_t(s_{t-1}, \lambda)$
  end for
  $\alpha_t \leftarrow \nabla E^{val}(s_t)$
  $g \leftarrow \alpha_t B_t$
  for $t = k\Delta - 1$ **downto** $1 + (k-1)\Delta$ **do**
    $\alpha_t \leftarrow \alpha_{t+1} A_{t+1}$
    $g \leftarrow g + \alpha_t B_t$
  end for
  $\lambda \leftarrow \lambda - \eta g$
end for
return s

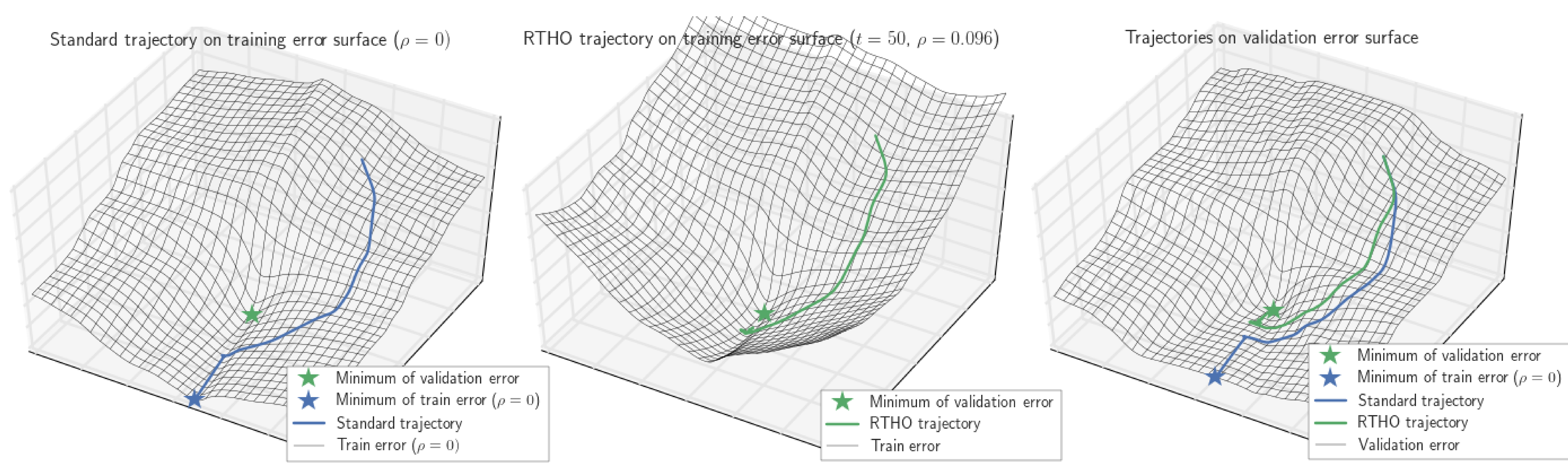
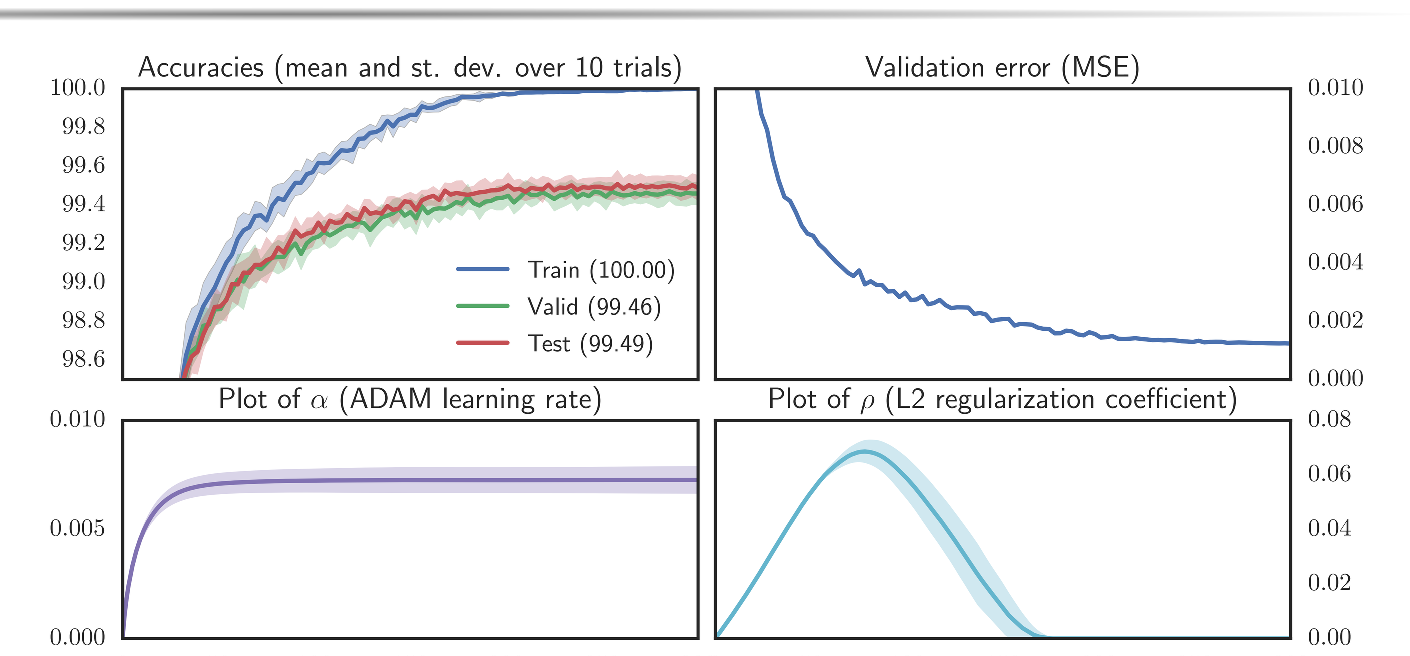
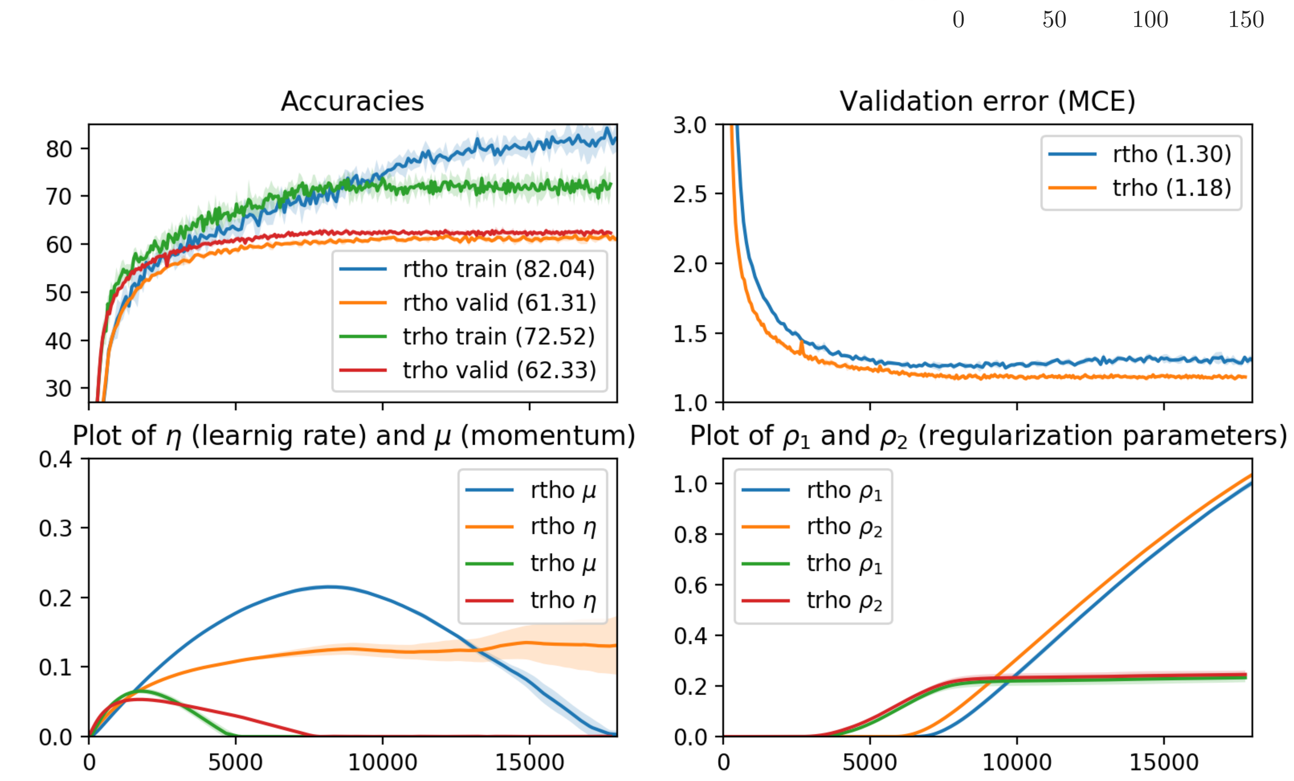


Figure 1: Representation of RTHO on a toy problem; learning rate and a regularization coefficient ρ are optimized. The plots represent optimization trajectories in the weight space. Note that, while executing RTHO, the training loss $E^{train} = MSE + \rho\|w\|^2$ changes at each hyper-batch (center), since ρ is updated while the training proceeds.

CNN and phone classification experiments

RTHO on a small convolutional neural net trained on MNIST. Φ is ADAM. We optimize learning rate α and L^2 regularization coefficient ρ. RTHO decreases in average 25% the test error over the baseline.





"Large scale" experiment on TIMIT dataset with **RTHO** and **TRHO**, with a 5-layers FFNN ($\sim 15 \times 10^6$ params). We optimize 2 algorithmic and 2 regularization hyper-parameters. Even with only 4 hyper-parameters **RTHO** is slower than **TRHO** (98 seconds for hyperiteration versus 59), but occupies less memory.

For details on previous work, please refer to: Franceschi, L., Donini, M., Frasconi, P. & Pontil, M.. Forward and Reverse Gradient-Based Hyperparameter Optimization. ICML 2017

Code at: https://github.com/lucfra/RFHO

Future work

- Study **proprieties and convergence** of algorithms, **improve** reliability (adaptiveness)
- Connections with *learning to learn*
- Applications in reinforcement learning?