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

\[
\begin{align*}
\min_{\lambda \in \Lambda} & \quad E^\text{val}(w^*(\lambda)) \\
\text{s.t.} & \quad w^*(\lambda) \in \arg \min_{w} E^\text{train}(w, \lambda)
\end{align*}
\]

- **Aims**: Automate ML, increase generalization, allow 'freen' design of models and problems
- **Difficulties**: Cost of evaluating \( E^\text{val} \), structure of \( \lambda \)
- **Approaches**: Manual, grid, random search; Bayesian optimization; gradient-based optimization

Gradient-based HO

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

Algorithms

**Algorithm 1** RTHO, based on Forward-HG

\[
\begin{align*}
\text{for } k = 1 \to \cdots & \text{ do} \\
Z_0 & \leftarrow 0 \\
\text{for } t = 1 \to \cdots & \text{ do} \\
Z_t & \leftarrow A_t Z_{t-1} + B_t \\
y_t & \leftarrow \Phi_t(s_{t-1}, \lambda) \\
\text{end for} \\
\text{end for} \\
\text{return } s
\end{align*}
\]

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

\[
\begin{align*}
\text{for } k = 1 \to \cdots & \text{ do} \\
Z_0 & \leftarrow 0 \\
\text{for } t = 1 \to \cdots & \text{ do} \\
Z_t & \leftarrow A_t Z_{t-1} + B_t \\
y_t & \leftarrow \Phi_t(s_{t-1}, \lambda) \\
\text{end for} \\
\text{end for} \\
\text{return } s
\end{align*}
\]

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

CNN and phone classification experiments

RTHO on a small convolutional neural net trained on MNIST. \( \Phi \) is ADAM. We optimize learning rate \( \alpha \) and \( L^2 \) regularization coefficient \( \rho \). RTHO decreases in average 25% the test error over the baseline.

**Future work**

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