

ON HYPERPARAMETER OPTIMIZATION IN LEARNING SYSTEMS

Luca Franceschi^{1,2}, Michele Donini¹, Paolo Frasconi³, Massimiliano Pontil^{1,2}

(1) Istituto Italiano di Tecnologia, Genoa, 16163 Italy

(2) Dept of Computer Science, University College London, London, WC1E 6BT, UK

(3) Dept of Information Engineering, Università degli Studi di Firenze, Firenze, 50139, Italy

ABSTRACT

We study two procedures (reverse-mode and forward-mode) for computing the gradient of the validation error with respect to the hyperparameters of any iterative learning algorithm. These procedures mirror two ways of computing gradients for recurrent neural networks and have different trade-offs in terms of running time and space requirements. The reverse-mode procedure extends previous work by Maclaurin et al. (2015) and offers the opportunity to insert constraints on the hyperparameters in a natural way. The forward-mode procedure is suitable for real-time hyperparameter updates, which may significantly speedup the overall hyperparameter optimization process.

1 INTRODUCTION

We study the problem of selecting the hyperparameters of a learning algorithm by gradient-based optimization (Bengio, 2000). We see the training procedure by stochastic gradient descent or its variants (momentum, RMSProp, ADAM, etc.) as a dynamical system with a state $s_t \in \mathbb{R}^d$ that collects weights and possibly accessory variables such as velocities and accumulated squared gradients. The dynamic is defined by the system of equations

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

where T is the number of iterations, s_0 is a starting state, and $\Phi_t : (\mathbb{R}^d \times \mathbb{R}^m) \rightarrow \mathbb{R}^d$ is a smooth mapping that represents the operation performed by the t -th step of the optimization algorithm (i.e. on mini-batch t). Finally, $\lambda \in \mathbb{R}^m$ is the vector of hyperparameters that we wish to tune. As a simple example consider training a neural network by momentum, in which case $s_t = (v_t, w_t) = (\mu v_{t-1} - \eta \nabla J_t(w_{t-1}), \mu v_{t-1} - \eta \nabla J_t(w_{t-1}) + w_{t-1})$, where J_t is the objective associated with the t -th mini-batch. In this example, $\lambda = (\mu, \gamma)$.

Note that the iterates s_1, \dots, s_T implicitly depend on the vector of hyperparameters λ . Our goal is to optimize the hyperparameters according to a certain error function E evaluated at the last iterate s_T . Specifically, we wish to solve the problem

$$\min_{\lambda} f_T(\lambda) \quad (2)$$

where the function $f_T : \mathbb{R}^m \rightarrow \mathbb{R}$ is defined at $\lambda \in \mathbb{R}^m$ as $f_T(\lambda) = E(s_T(\lambda))$.

We highlight the generality of the framework. The vector of hyperparameters λ may include components associated with the training objective, and components associated with the iterative algorithm. For example, the training objective may depend on hyperparameters used to design the loss function as well as multiple regularization parameters. Yet other components of λ may be associated with the space of functions used to fit the training data (e.g. number of layers and weights of a neural network, parameters associated with the kernel function used within a kernel-based method, etc.).

2 FORWARD-MODE COMPUTATION

Our first approach to compute the hypergradient, displayed in the appendix as FORWARD-HO, uses the chain rule for the derivative of composite functions (we regard the gradient of a scalar function

as a row vector) to obtain that

$$\nabla f_T(\lambda) = \nabla E(s_T) \frac{ds_T}{d\lambda} \quad (3)$$

where $\frac{ds_T}{d\lambda}$ is the $d \times m$ matrix formed by the total derivative of the components of s_T (regarded as rows) with respect to the components of λ (regarded as columns).

The operators Φ_t depends on the hyperparameter λ both directly by its expression and indirectly through the state s_{t-1} . Using again the chain rule we have, for every $t \in \{1, \dots, T\}$, that

$$\frac{ds_t}{d\lambda} = \frac{\partial \Phi_t(s_{t-1}, \lambda)}{\partial s_{t-1}} \frac{ds_{t-1}}{d\lambda} + \frac{\partial \Phi_t(s_{t-1}, \lambda)}{\partial \lambda}. \quad (4)$$

For every $t \in \{1, \dots, T\}$, we define the matrices $Z_t = \frac{ds_t}{d\lambda}$, $A_t = \frac{\partial \Phi_t(s_{t-1}, \lambda)}{\partial s_{t-1}}$ and $B_t = \frac{\partial \Phi_t(s_{t-1}, \lambda)}{\partial \lambda}$. Using these, we rewrite equation (4) as the recursion

$$Z_t = A_t Z_{t-1} + B_t, \quad t \in \{1, \dots, T\}. \quad (5)$$

Combining equations (3) and (5) we obtain that

$$\nabla f_T(\lambda) = \nabla E(s_T) Z_T = \nabla E(s_T) (A_T Z_{T-1} + B_T) = \dots = \nabla E(s_T) \sum_{t=1}^T (A_{t+1} \dots A_T) B_t. \quad (6)$$

From the above derivation it is apparent that $\nabla f_T(\lambda)$ can be computed by an iterative algorithm which runs in parallel as the training algorithm. Furthermore, it is apparent that *partial* hypergradients

$$\nabla f_t(\lambda) = \frac{dE(s_t)}{d\lambda} = \nabla E(s_t) Z_t \quad (7)$$

are available at each time step $t = 1, \dots, T$ of FORWARD-HO and not only at the end. This means that we are allowed to update hyperparameters several times in a single optimization epoch, without having to wait until time T . This approach is reminiscent of (Williams & Zipser, 1989, Eq. (2.10)) for real time recurrent learning and may be suitable in the case of a data stream (i.e. $T = \infty$), where the REVERSE-HO algorithm (see below) would be hardly applicable.

3 REVERSE-MODE COMPUTATION

The second approach to compute the hypergradient leads to an extension of the backward algorithm presented in (Maclaurin et al., 2015), which turns out to be structurally identical to backpropagation through time (Werbos, 1990). We start by reformulating problem (2) as the constrained optimization problem

$$\min_{\lambda, s_1, \dots, s_T} E(s_T) \quad \text{s.t.} \quad s_t = \Phi_t(s_{t-1}, \lambda), \quad t \in \{1, \dots, T\}. \quad (8)$$

This formulation closely follows a classical Lagrangian formalism used to derive the backpropagation algorithm (LeCun, 1988). Furthermore, the framework naturally allows us to incorporate constraints on the hyperparameters. The Lagrangian of problem (8) is $\mathcal{L}(s, \lambda, \alpha) = E(s_T) + \sum_{t=1}^T \alpha_t (\Phi_t(s_{t-1}, \lambda) - s_t)$, where, for each $t \in \{1, \dots, T\}$, $\alpha_t \in \mathbb{R}^d$ is a row vector of Lagrange multipliers associated with the t -th step of the dynamic.

Setting the partial derivatives of the Lagrangian w.r.t. s_t to zero gives $\alpha_T = \nabla E(s_T)$ and $\alpha_t = \nabla E(s_T) A_T \dots A_{t+1}$ if $t \in \{1, \dots, T-1\}$. Furthermore a direct computation gives that $\frac{\partial \mathcal{L}}{\partial \lambda} = \nabla E(s_T) \sum_{t=1}^T (A_{t+1} \dots A_T) B_t$, which coincides with the expression for the gradient of f in equation (6). The pseudocode of the algorithm is reported in the appendix and denoted REVERSE-HO.

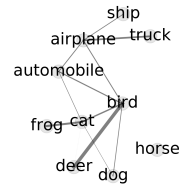
4 EXPERIMENTS

We present two preliminary experiments which highlight the potential of the HO framework.

In the first experiment, we used the CIFAR-10 dataset (Krizhevsky & Hinton, 2009). As features we employed the pre-activation of the second last layer of Inception-V3 model trained on ImageNet¹.

¹<http://download.tensorflow.org/models/image/imagenet/>

| | Accuracy |
|--------|----------|
| STL | 0.7064 |
| NMTL | 0.7160 |
| HMTL | 0.7284 |
| HMTL-S | 0.7301 |



| | Accuracy | CPU Time |
|----------|----------|----------|
| Baseline | 0.4567 | 712 |
| RT-HO | 0.4616 | 412 |

Figure 1: **Left:** test accuracy on CIFAR-10 for Single Task Learning (STL), Naive MTL (NMTL) and our approach without (HMTL) and with (HMTL-S) the L1 constraint on A . **Center:** relationship graph of CIFAR-10 classes. **Right:** frame level phone state classification accuracy on the test set (Accuracy) and CPU time in seconds for grid-search on ρ (Baseline), and real-time hyperparameter optimization with FORWARD-HO (RT-HO). In RT-HO we update the hyperparameters every 20 steps of SGDM and we employ an early stopping mechanism to prevent overfitting on the training set.

We extracted 50 examples as the training set, 50 examples as the validation set, and the remaining 49900 examples as the test set. We are interested in learning matrix $W = [w_1, \dots, w_K]$, where each column is the model for a specific class and K is the number of classes. In order to leverage information among the different classes we add a multi-task learning (MTL) regularizer (Evgeniou et al., 2005) $\Omega_{A,\rho}(W) = \sum_{j,k=1}^K A_{j,k} \|w_j - w_k\|_2^2 + \rho \sum_{k=1}^K \|w_k\|^2$, where the symmetric matrix A models the interactions between the classes/tasks. We compared this model to the Naive MTL (NMTL) in which the tasks are equally related, that is $A_{j,k} = a$ for every $1 \leq i, j \leq K$. In this case we have two hyperparameters ($a, \rho \geq 0$). We used a regularized training error defined as $E_{\text{tr}}(W) = \sum_{i \in \mathcal{D}_{\text{tr}}} \ell(Wx_i + b, y_i) + \Omega_{A,\rho}(W)$ where $\ell(\cdot, \cdot)$ is the softmax regression loss. We wish solve the following optimization problem $\min \{E_{\text{val}}(W_T, b_T) \text{ subject to } \rho \geq 0, A \geq 0\}$, where (w_T, b_T) is the T -th iteration obtained by running ADAM on the training objective. The hyperparameters are optimized by projected ADAM on the set $\{(\rho, A) : \rho \geq 0, A \geq 0\}$. We compare the following methods: Single Task Learning (STL) using a validation set to tune the optimal value of ρ for each task; Naive MTL (NMTL) where the hyperparameters a and ρ are validated; our hyperparameter optimization method in REVERSE-HO to tune A and ρ (HMTL); as a way to remove spurious relationships among the tasks we further imposed the constraint $\sum_{j,k=1}^K A_{j,k} \leq R$, where $R = 10^{-3}$. This last method is denoted HMTL-S. The accuracy of each method is presented in Figure 1 (left). Figure 1 (Center) shows matrix A (obtained using HMTL-S) as an adjacency graph, highlighting the discovered task relationships.

The aim of the second experiment is to assess the efficacy of the real-time FORWARD-HO algorithm. We run experiments on a small subset of the TIMIT phonetic recognition dataset² (Garofolo et al., 1993). The primary target is to learn a mapping $f_P : \mathbb{R}^{123} \rightarrow \{0, 1\}^{183}$ from frame vectors of Mel-filter banks to mono-phone states. The training set \mathcal{D}_{tr} is augmented to include 300-dimensional real vectors of context-dependent phonetic embeddings, which serve as targets for a secondary task of learning a mapping $f_S : \mathbb{R}^{123} \rightarrow \mathbb{R}^{300}$. It is assumed that f_P and f_S share an intermediate representation, see (Badino, 2016) and references therein. We implemented f_P and f_S as two five-layers feed-forward neural networks with partially shared parameters W^P and W^S . The networks are trained to jointly minimize $E_{\text{tot}}(W^P, W^S) = E_P(W^P) + \rho E_S(W^S)$, where the primary error E_P is given by cross-entropy loss on the phone states y , the secondary error E_S is given by mean squared error on the embedding vectors and $\rho \geq 0$ is a design hyperparameter. Since we are ultimately interested in learning f_P , we formulate the hyperparameter optimization problem as $\min \{E_{\text{val}}(W_T^P) \text{ subject to } \rho, \eta \geq 0, 0 \leq \mu \leq 1\}$, where $E_{\text{val}}(W_T^P)$ is the cross entropy loss computed on a validation set after T iterations of real-time gradient descent with momentum (RT-HO), and η and μ are respectively the learning rate and momentum factor of the learning dynamics. The optimization of the hyperparameters is carried out as in the previous experiment with the exception that we use FORWARD-HO algorithm to compute online partial hypergradients. Preliminary results reported in Figure 1 (Right) suggest that the simultaneous optimization of parameters and hyperparameters (the latter optimized via real-time hypergradient descent on the validation error) could bring to faster convergence and improved generalization performances. Current efforts aim to experiment the method on larger and deeper networks trained on full TIMIT dataset.

²Roughly a twelfth of the full dataset. The test set employed is instead the standard one.

APPENDIX

We report the two algorithms presented in the main body of the paper.

Algorithm 1 FORWARD-HO

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

Algorithm 2 REVERSE-HO

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

Let $g(d, m)$ and $h(d, m)$ denote time and space required to evaluate Φ , respectively. By basic facts from the algorithmic differentiation literature (Griewank & Walther, 2008) (see longer version of this paper for details), the FORWARD-HO algorithm runs in time $O(Tmg(d, m))$ and space $O(h(d, m))$, whereas REVERSE-HO runs in time $O(Tg(d, m))$ and space $O(Th(d, m))$.

ACKNOWLEDGEMENTS

We wish to thank Leonardo Badino for useful comments and providing the TIMIT dataset.

REFERENCES

- Leonardo Badino. Phonetic context embeddings for dnn-hmm phone recognition. In *Proceedings of Interspeech*, pp. 405–409, 2016.
- Yoshua Bengio. Gradient-based optimization of hyperparameters. *Neural computation*, 12(8):1889–1900, 2000.
- Theodoros Evgeniou, Charles A Micchelli, and Massimiliano Pontil. Learning multiple tasks with kernel methods. *Journal of Machine Learning Research*, 6(Apr):615–637, 2005.
- John S Garofolo, Lori F Lamel, William M Fisher, Jonathon G Fiscus, and David S Pallett. Darpa timit acoustic-phonetic continuous speech corpus cd-rom. nist speech disc 1-1.1. *NASA STI/Recon technical report n*, 93, 1993.
- Andreas Griewank and Andrea Walther. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation, Second Edition*. Society for Industrial and Applied Mathematics, second edition, 2008.
- Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009.

Yann LeCun. A Theoretical Framework for Back-Propagation. In Geoffrey Hinton and Terrence Sejnowski (eds.), *Proc. of the 1988 Connectionist models summer school*, pp. 21–28. Morgan Kaufmann, 1988.

Dougal Maclaurin, David K Duvenaud, and Ryan P Adams. Gradient-based hyperparameter optimization through reversible learning. In *ICML*, pp. 2113–2122, 2015.

Paul J Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.

Ronald J. Williams and David Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2):270–280, 1989.