Efficient Hyperparameter Tuning with Dynamic Accuracy
Derivative-Free Optimization

Matthias J. Ehrhardt
Institute for Mathematical Innovation and Department of Mathematical Sciences, University of Bath

Lindon Roberts
Mathematical Sciences Institute, Australian National University

Abstract

Many machine learning solutions are framed as optimization problems which rely on good hyperparameters. Algorithms for tuning these hyperparameters usually assume access to exact solutions to the underlying learning problem, which is typically not practical. Here, we apply a recent dynamic accuracy derivative-free optimization method to hyperparameter tuning, which allows inexact evaluations of the learning problem while retaining convergence guarantees. We test the method on the problem of learning elastic net weights for a logistic classifier, and demonstrate its robustness and efficiency compared to a fixed accuracy approach. This demonstrates a promising approach for hyperparameter tuning, with both convergence guarantees and practical performance.

1. Introduction

Tuning model hyperparameters is a common problem encountered in machine learning for estimating regularization parameters, learning rates, and many other important quantities. In many cases, hyperparameter tuning can be viewed as a bilevel optimization problem for hyperparameters $\theta$:

$$ \min_{\theta, w} F(\theta, w) \quad \text{subject to} \quad w \in \arg \min_{\tilde{w}} \Phi(\tilde{w}, \theta), $$

where the lower-level objective $\Phi$ corresponds to a learning problem for weights $w$, and the upper-level objective $F$ typically measures a test error. Hyperparameter tuning often uses global optimization methods [13], but recent work has also considered local gradient-based [24, 28] and derivative-free [15, 21] methods. The lower-level problem is usually solved with iterative methods, so only inexact evaluations of feasible $w$ are available. As a result, few algorithms have convergence theory; exceptions include [23], [28] under specific assumptions and [15] (in light of [9]). Bilevel learning is also used in inverse problems (e.g. [1, 20, 26, 29]), although other approaches exist [5, 12, 17].

Recently in [11], a model-based DFO method [2, 10] for bilevel learning (in an inverse problems context) was proposed, which has convergence guarantees while still allowing inexact lower-level minimizers (but with a dynamic accuracy controlled by the algorithm). Here, we apply this algorithm to bilevel optimization for hyperparameter tuning (1). Specifically, we apply the framework of [11] to the case where the lower-level objective is nonsmooth (but strongly convex). Our approach has similarities to Hyperband [23], which considers an inexact Bayesian optimization method, whereas our approach is designed to exploit the bilevel structure to find a local minimum. We can achieve the dynamic accuracy requirements for the upper-level algorithm, but delegate full convergence theory to future work (noting that model-based DFO methods for nonsmooth problems
exist; e.g. \([14, 16, 19]\)). We show results for tuning elastic net weights of a logistic classifier. Compared to fixed-accuracy variants of the same DFO method, our approach has strong computational performance and robustness, and avoids manually tuning the lower-level problem solution accuracy.

2. Lower-Level Problem

First, we consider a general formulation for solving nonsmooth but strongly convex learning problems. Suppose that we wish to learn weights \(w \in \mathbb{R}^d\) by minimizing a loss function which depends on parameters \(\theta \in \mathbb{R}^m\); that is, we wish to solve

\[
\hat{w}(\theta) := \arg \min_{w \in \mathbb{R}^d} \Phi(w, \theta) := f(w, \theta) + g(w, \theta),
\]

where \(f(\cdot, \theta)\) is \(\mu(\theta)\)-strongly convex and smooth with \(L(\theta)\)-Lipschitz continuous gradient, and \(g(\cdot, \theta)\) is convex and possibly nonsmooth. As a consequence, we note that \(\Phi(w, \theta)\) is also \(\mu(\theta)\)-strongly convex [3, Lemma 5.20]. This is a standard problem type, and in this work we solve (2) using a variant of FISTA [4] designed for strongly convex objectives [8, Algorithm 5]:

\[
t_{k+1} = \frac{1 - \eta t_k^2 + \sqrt{(1 - \eta t_k^2)^2 + 4 \eta I_k^2}}{2}, \quad \beta_{k+1} = \frac{[t_k - 1] (1 - t_{k+1} q)}{[t_{k+1} (1 - q)]},
\]

\[
z^{k+1} = w^k + \beta_{k+1} (w^k - w^{k-1}), \quad w^{k+1} = \text{prox}_{\tau g(\cdot, \theta)}(z^{k+1} - \tau \nabla w f(z^{k+1}, \theta)),
\]

where \(w^{-1} := w^0, q := \tau \mu(\theta)\), and we choose \(\tau = 1/L(\theta)\) and \(t_0 = 0\). This algorithm achieves linear convergence to \(\hat{w}(\theta)\) with the a priori estimate (from [8, Theorem 4.10] and \(\Phi(w^k, \theta) - \Phi(\hat{w}(\theta), \theta) \geq (\mu/2)\|w^k - \hat{w}(\theta)\|^2\) using strong convexity, and defining \(\kappa(\theta) := L(\theta)/\mu(\theta)\):

\[
\|w^k - \hat{w}(\theta)\|^2_2 \leq \left(1 - \kappa(\theta)^{-1/2}\right)^k \left[\kappa(\theta) \left(1 + \kappa(\theta)^{-1/2}\right) \|w^0 - \hat{w}(\theta)\|^2\right],
\]

Guaranteeing Sufficient Accuracy  In the dynamic accuracy DFO framework [11], evaluations of \(\hat{w}\) need to sufficiently accurate in that sense that we terminate the iteration (3) once \(\|w^k - \hat{w}(\theta)\|^2_2 \leq \epsilon\) is achieved, for some \(\epsilon > 0\) specified by the upper-level solver. A priori, we can determine the required number of iterations using (4). However, since it is difficult to estimate \(\|w^0 - \hat{w}(\theta)\|_2\) in practice, we also consider the a posteriori termination criterion

\[
\|w^k - \hat{w}(\theta)\|^2_2 \leq d(w^k)^2/\mu(\theta)^2 \leq \epsilon,
\]

where \(d(w^k) \in \partial \Phi(w^k, \theta)\), which ensures the required accuracy from the \(\mu(\theta)\)-strong convexity of \(\Phi(\cdot, \theta)\) [3, Theorem 5.24]. In FISTA, we calculate \(d(w^k)\) as

\[
d(w^k) = \nabla w f(w^{k+1}, \theta) - \nabla w f(z^{k+1}, \theta) + \frac{1}{\tau}(z^{k+1} - w^{k+1}) \in \partial \Phi(w^{k+1}, \theta),
\]

which follows from noting \(z^{k+1} - \tau \nabla w f(z^{k+1}, \theta) - w^{k+1} \in \partial (\tau g(\cdot, \theta))\), which follows from the properties of the proximal operator [3, Theorem 6.39].

In Figure 1, we compare the a priori (4) and a posteriori error (5) bounds on \(\|w^k - \hat{w}(\theta)\|^2_2\) on the linear inverse problem

\[
\Phi(w, \theta) = \frac{1}{2} \|Aw - b\|^2_2 + \frac{\theta_1}{2} \|w\|^2_2 + \theta_2 \|w\|_1,
\]
for \( \theta = [\theta_1, \theta_2]^T \), where we use a randomly generated \( A \in \mathbb{R}^{100 \times 200}, w^0 \in \mathbb{R}^{200} \) and \( b \in \mathbb{R}^{100} \), and parameters \( \theta = [10, 10]^T \). For this problem, we have \( L(\theta) \approx 1.96e4 \) and \( \mu(\theta) = 10 \). Despite the a priori bound using the correct value of \( \|w^0 - \hat{w}(\theta)\|_2 \), the a posteriori subgradient bound gives a much tighter estimate of \( \|w^k - \hat{w}(\theta)\|_2^2 \). Because of this and not being able to estimate \( \|w^0 - \hat{w}(\theta)\|_2 \) accurately in practice, in our numerical results we only use the subgradient bound to ensure the required accuracy.

3. Upper-Level Problem

We now describe our algorithm for solving the upper-level problem of learning \( \theta \). Suppose we have access to \( n \) lower-level problems which depend on the same parameters \( \theta \in \mathbb{R}^m \), and that each lower-level minimizer \( \hat{w}_j(\theta) \in \mathbb{R}^{d_j} \) has an associated loss function \( \ell_j : \mathbb{R}^{d_j} \rightarrow \mathbb{R} \); in Section 4, this will be a measure of test error. Then, our upper-level problem is

\[
\min_{\theta \in \mathbb{R}^m} F(\theta) := \sum_{j=1}^n \ell_j(\hat{w}_j(\theta)) + \mathcal{J}(\theta), \quad \text{subject to} \quad \hat{w}_j(\theta) := \arg \min_{w \in \mathbb{R}^d} \Phi_j(w, \theta), \tag{7}
\]

where \( \mathcal{J}(\theta) \) is an optional regularization term for \( \theta \), and \( \Phi_j \) is strongly convex in \( w \) (but possibly nonsmooth) and smooth in \( \theta \). Under these assumptions, there are various results showing that \( \hat{w}_j(\theta) \) is locally Lipschitz in \( \theta \) for a variety of problems; e.g. [18, 27].

To solve (7) we follow [11] and use a model-based DFO method. Here, at every iteration \( k \) we have a collection of points \( \{y_0^{(k)}, \ldots, y_m^{(k)}\} \) at which we have evaluated \( \hat{w}_j \) (for each \( j \)), to some accuracy. We use this information to build a local quadratic model \( m_k(\theta) \) which approximates \( F(\theta) \) near \( \theta^{(k)} \) by requiring that the model interpolate \( F \) at each of \( y_0^{(k)}, \ldots, y_m^{(k)} \). Our upper-level iteration can be summarized as:

1. Build the interpolating local quadratic model \( m_k(\theta) \approx F(\theta) \) using (inexact) evaluations of \( \hat{w}_j(y_0^{(k)}), \ldots, \hat{w}_j(y_m^{(k)}) \).
2. Calculate a minimizer \( s^{(k)} \) of the model inside a trust region, namely solve (possibly inexactly) \( \min_{\|s\|_2 \leq \Delta_k} m_k(\theta^{(k)} + s) \), for a given trust-region radius \( \Delta_k > 0 \).
3. Ensure \( F(\theta^{(k)}) \) and \( F(\theta^{(k)} + s^{(k)}) \) are evaluated to sufficiently high accuracy, and then if sufficient decrease is achieved, set \( \theta^{(k+1)} = \theta^{(k)} + s^{(k)} \) and increase \( \Delta_k \), otherwise set \( \theta^{(k+1)} = \theta^{(k)} \) and decrease \( \Delta_k \).
4. Replace one interpolation point with $\theta^{(k)} + s^{(k)}$.

This algorithm typically requires that our inexact lower-level evaluations $w_j(\theta) \approx \hat{w}_j(\theta)$ satisfy $\|w_j(\theta) - \hat{w}_j(\theta)\|_2 \leq c \Delta_k^2$ for some $c > 0$ (where we use $c = 100$ in Section 4 below).

Full details of a suitable dynamic accuracy upper-level algorithm are given in [11]. There, it is shown that, if $f_j$ and each $\ell_j$ and $\Phi_j$ are sufficiently smooth, and each $\Phi_j$ is strongly convex in $w$, then the model-based DFO algorithm finds a $\theta$ with $\|\nabla F(\theta)\|_2 \leq \epsilon$ in at most $\mathcal{O}(\epsilon^{-2})$ upper-level iterations, or at most $\mathcal{O}(\epsilon^{-2} \log \epsilon)$ lower-level iterations.

4. Numerical Results

**Summary of Test Problem**  We test our approach by attempting to learn elastic net regularizer [30] weights $\theta \in \mathbb{R}^2$ for a logistic classifier of each digit 0–5 in MNIST [22] (i.e. each lower-level problem corresponds to a classifier for a different digit and the upper-level problem measures a test set error). Below, we validate the results of our approach by considering the remaining digits. To test our approach, we follow [11] and use a version of DFO-LS [6, 7] which is modified to handle dynamic accuracy evaluations as per Section 3. We use FISTA (3) for the lower-level solver, and compare the dynamic accuracy approach with a combination of FISTA and DFO-LS, but using a fixed number of FISTA iterations $K$ for every lower-level solve. Full details are given in Appendix A.

**General Comparison**  Figure 2 compares the dynamic-accuracy variant against fixed-accuracy variants with low ($K = 20$), medium ($K = 200$) and high accuracy ($K = 2000$), with upper-level initialization $\theta^{(0)} = [1, 1]^T$ in all cases. Overall, dynamic accuracy is comparable to medium accuracy: it reaches the same $\theta$ at a similar speed. By comparison, low accuracy gives a quite different value of $\theta$ to the others, and high accuracy takes substantially more computational effort to converge. A key point to note here, is that dynamic accuracy performs similar to the best fixed-accuracy variant without requiring the necessary accuracy a priori.

**Robustness**  It was observed in [11] that dynamic accuracy is more robust to initialization and a good initialization can accelerate its initial objective decrease. To test these features here, we perform the same tests as above, but vary the initial $\ell_2$ penalty by choosing $\theta_{2}^{(0)} \in \{-3, -2, \ldots, 3\}$. Figure 3 shows that only high accuracy and dynamic accuracy give stable results, whereas the low and medium accuracy converge to different $\theta$ depending on the starting point. Figure 4 shows that as $\theta_{2}^{(0)}$ increases, the relative performance of dynamic accuracy improves. Thus, since dynamic
Figure 3: Final values of $\theta_1$ (left) and $\theta_2$ (right) reached for different starting points $\theta_2^{(0)}$.

Figure 4: Comparison of objective reduction for different choices of starting point $\theta_2^{(0)}$.

Accuracy is stable with respect to initialization, selecting $\theta^{(0)}$ to give well-conditioned lower-level problems is appropriate and the associated reduction in computational work can be achieved.

**Validation** Finally, we measure the generalizability of the upper-level minimizers. Here, the learned values of $\theta$ from each accuracy variant and each initialization $\theta^{(0)}$ are used to train logistic classifiers for MNIST (using a new set of 5000 training images), and measure their test error (on another new set of 1000 test images). Despite the upper-level problem only considering digits 0–5, here we look at the performance of each $\theta$ in training classifiers for all digits. We measure the test error by looking at the proportion of test images classified correctly (using the predictor $\hat{y}_i^{(j)} = 1$ if $p_i^{(j)}(\hat{w}_j(\theta)) \geq 0.5$; see Appendix A). Figure 5 shows the validation for each digit $j \in \{0, \ldots, 9\}$ and initialization $\theta_2^{(0)}$. The overall accuracy is generally lower for digits 6–9 than the digits used in the upper-level objective (0–5). However, we see consistently that dynamic and high accuracy achieve better prediction accuracy than low and medium accuracy.

Figure 5: Comparison of validation results for different choices of starting point $\theta_2^{(0)}$. 

5. Conclusions and Further Work

We have extended the approach in [11] to the case of nonsmooth lower-level problems—we can guarantee solutions to the lower-level problem are sufficiently accurate for the upper-level algorithm. When applied to learning elastic net regularizer weights for MNIST, we find that the dynamic accuracy DFO approach gives a robust method for hyperparameter tuning. Compared to the fixed-accuracy variants, which have strong dependency on the selected number of iterations (both in computational speed and quality of minimizer), the dynamic-accuracy version can achieve both fast progress and high-quality minimizers, without needing to be manually tuned. We delegate to future work a full study of the convergence of the upper-level algorithm in this setting, and comparisons between the dynamic accuracy DFO approach and other methods for hyperparameter tuning.

5.1. Acknowledgements

MJE acknowledges support from the EPSRC (EP/S026045/1, EP/T026693/1), the Faraday Institution (EP/T007745/1) and the Leverhulme Trust (ECF-2019-478).

References


Appendix A. Details of Numerical Test Model

General Problem Formulation In this work, we focus on a specific choice of problem, namely selecting regularizer weights for logistic regression [25, Chapter 8]. Specifically, for upper-level problem \( j \in \{1, \ldots, n\} \), suppose we have training pairs \( \{(x_i^{(j)}, y_i^{(j)})\}_{i=1}^{N_j} \) with features \( x_i^{(j)} \in \mathbb{R}^{d_j} \) and binary labels \( y_i^{(j)} \in \{-1, 1\} \). Minimizing the negative log-likelihood of the corresponding logistic classifier, with an elastic net regularizer [30], corresponds to the lower-level problem

\[
\hat{w}_j(\theta) := \arg \min_{w \in \mathbb{R}^{d_j}} \Phi_j(w, \theta) := \frac{1}{N_j} \sum_{i=1}^{N_j} \log(1 + \exp(-y_i^{(j)} w^T x_i^{(j)})) + \frac{10^{\theta_1}}{2} \|w\|^2 + 10^{\theta_2} \|w\|_1,
\]

for (undetermined) regularizer log-weights \( \theta_1, \theta_2 \in \mathbb{R} \). That is, we form \( \Phi_j(w, \theta) \) (2) by taking

\[
f_j(w, \theta) := \frac{1}{N_j} \sum_{i=1}^{N_j} \log(1 + \exp(-y_i^{(j)} w^T x_i^{(j)})) + \frac{10^{\theta_1}}{2} \|w\|^2,
\]

and so \( f_j(\cdot, \theta) \) is \( \mu_j(\theta) \)-strongly convex and \( L_j(\theta) \)-smooth with

\[
\mu_j(\theta) := 10^{\theta_1}, \quad \text{and} \quad L_j(\theta) := \frac{1}{4N_j} \|X^{(j)}\|_2^2 + 10^{\theta_1},
\]

where \( X^{(j)} \in \mathbb{R}^{N_j \times d_j} \) has columns \( x_i^{(j)} \).

As our upper-level objective, suppose for each lower-level problem \( j \) we have learned weights \( w_j(\theta) \in \mathbb{R}^{d_j} \) (in practice \( w_j(\theta) \) is our approximation to \( \hat{w}_j(\theta) \) from solving (8) to some finite accuracy). We then suppose we have a test set \( \{\tilde{x}_i^{(j)}, \tilde{y}_i^{(j)}\}_{i=1}^{\tilde{N}_j} \), and we calculate the estimated probabilities

\[
p_i^{(j)}(\hat{w}_j(\theta)) = \text{sigm}(\hat{w}_j(\theta)^T \tilde{x}_i^{(j)}) \text{ is the probability that } \tilde{y}_i^{(j)} = 1 \text{ as determined by the}
\]
Efficient Hyperparameter Tuning with DFO

classifier. We then take our loss for the lower-level problem \( j \) to be a smoothed approximation of the test accuracy:

\[
\ell_j(\hat{w}_j(\theta)) := \sum_{i=1}^{\tilde{N}_j} \left[ p_i^{(j)}(\hat{w}_j(\theta)) - \tilde{p}_i^{(j)} \right]^2, \quad \text{where} \quad \tilde{p}_i^{(j)} = \begin{cases} 1, & \tilde{y}_i^{(j)} = 1, \\ 0, & \text{otherwise}. \end{cases}
\]

That is, \( \ell_j(\theta) \) is the squared Euclidean distance from the vector \( [p_1^{(j)}, \ldots, p_{\tilde{N}_j}^{(j)}]^T \) to the ‘perfect’ classifier’s probabilities \( [\tilde{p}_1^{(j)}, \ldots, \tilde{p}_{\tilde{N}_j}^{(j)}]^T \).

**Example Problem** We take our training data from MNIST [22], where we attempt to learn a classifier for each digit separately (i.e. for digit \( j \in \{0, \ldots, 9\} \), learn a classifier for whether image \( i \) is of digit \( j \) or not). In this case, we have the same features for all lower-level problems, and we choose \( \bar{N}_j = 5000 \) and \( \tilde{N}_j = 1000 \) images for the training and test sets respectively. To test the generalizability of our approach, we only solve the upper-level problem for digits 0–5 (i.e. \( n = 6 \)). We validate the final learned \( \theta \) on the same problem for the remaining digits 6–9.

We augment our upper-level problem with a regularizer which encourages well-conditioned lower-level problems and a large value of \( \theta_2 \) (which should yield sparse weights \( w_j \)):

\[
\mathcal{J}(\theta) := \alpha_1 \left( \frac{L(\theta)}{\mu(\theta)} \right)^2 + \alpha_2 10^{-\theta_2},
\]

for weights \( \alpha_1 = 10^{-8} \) and \( \alpha_2 = 1 \).

To test our approach, we follow [11] and use a version of DFO-LS [6, 7] which is modified to handle dynamic accuracy evaluations as per Section 3. We use FISTA (3) for the lower-level solver, and compare the dynamic accuracy approach with a combination of FISTA and DFO-LS, but using a fixed number of FISTA iterations \( K \) for every lower-level solve. We impose bounds \( \theta \in [10^{-8}, 10^8]^2 \), and terminate after 80 upper-level evaluations or the trust-region radius in DFO-LS reaches \( 10^{-5} \).