OPT2024: 16th Annual Workshop on Optimization for Machine Learning

Second-Order Forward-Mode Automatic Differentiation for Optimization

Adam D. Cobb ADAM.COBB@SRI.COM
Neuro-symbolic Computing and Intelligence, SRI, USA

Atılım Güneş Baydin GUNES@ROBOTS.OX.AC.UK
University of Oxford, United Kingdom

Barak A. Pearlmutter BARAK@PEARLMUTTER.NET
Dept of Computer Science, Maynooth University, Ireland

Susmit Jha SUSMIT.JHA@SRI.COM
Neuro-symbolic Computing and Intelligence, SRI, USA

Abstract

This paper introduces a second-order hyperplane search, a novel optimization step that generalizes a second-order line search from a line to a K-dimensional hyperplane. This, combined with the forward-mode stochastic gradient method, yields a second-order optimization algorithm that consists of forward passes only, completely avoiding the storage overhead of backpropagation. Unlike recent work that relies on directional derivatives (or Jacobian–Vector Products, JVPs), we use hyper-dual numbers to jointly evaluate both directional derivatives and their second-order quadratic terms. As a result, we introduce forward-mode weight perturbation with Hessian information for K-dimensional hyperplane search (FoMoH-KD). We derive the convergence properties of FoMoH-KD and show how it generalizes to Newton's method for $K = D$. We also compare its convergence rate to forward gradient descent (FGD) and show FoMoH-KD has an exponential convergence rate compared to FGD's sub-linear convergence for positive definite quadratic functions. We illustrate the utility of this extension and how it might be used to overcome some of the recent challenges of optimizing machine learning models without backpropagation.

1. Introduction

There is a growing interest in investigating the practical plausibility of forward-mode automatic differentiation (AD) as an alternative to reverse-mode AD (aka backpropagation (BP)) for optimization. Forward gradient descent (FGD) [1] relies on sampling tangent vectors to update function parameters. These parameters are updated by subtracting the tangents that are scaled by their directional derivatives. As a result, their approach only requires forward passes, and avoids the computation and memory costs associated with implementing the backwards pass of reverse-mode AD. While there is still an accuracy gap between forward-mode and reverse-mode optimization approaches [16], there are several recent efforts that have explored alternative ways of leveraging forward-mode AD, with a focus of reducing the variance of the gradient estimator with the increase in dimensions [5, 13]. Thus, the large performance gap between the forward-mode approaches and BP has been shrinking.

In this paper, we explore introducing Hessian information to forward-mode AD for optimization. Second-order derivative information provides optimization routines with information about the local curvature. These methods often require access to the Hessian, whose storage is not feasible for high-dimensional problems. Instead of building the full Hessian, we introduce a new second-order approach that leverages a partial Hessian. Our approach, FoMoH-KD, builds a $K \times K$ Hessian in a sub-space of the full function space. FoMoH-KD is a forward-mode AD approach that outperforms the previous forward-mode-only first-order gradient descent method, FGD. We demonstrate and prove that our approach bridges the gap between a line search and a full Newton step. By incorporating Hessian information, FoMoH-KD leverages second-order derivatives to provide curvature approximations of the objective function. This allows the method to adjust the step size and direction with greater precision compared to a simple line search. Simultaneously, FoMoH-KD avoids the computational complexity of a full Newton step, which requires calculation of the inverse Hessian gradient multiplication. Our approach balances the efficiency of a line search with the accuracy of a full Newton method, offering a robust and versatile optimization technique. In summary:

Approach: We introduce FoMoH-KD, a novel forward-mode-only hyperplane search that uses second-order information.

Theory: We show and prove that FoMoH-KD generalizes to Newton's method, without the need for backpropagation. We also compare the convergence rate of FGD, to FoMoH-KD.

Experiments: We demonstrate FoMoH-KD on optimization problems of different parameter sizes and difficulty to show the advantage of second-order information.

Open-Source Code: We release an AD backend in PyTorch that implements nested forward AD and interfaces with PyTorch models at https://github.com/SRI-CSL/fomoh.

2. Related Work

There has been considerable interest in developing approaches that avoid reverse-mode AD and its backwards pass. In moving away from BP, it might be possible to build optimization algorithms that more closely align with biological systems [2, 8], or enable neural networks to run on emerging hardware architectures, such as analog optical systems [12]. Baydin et al. [1] introduced FGD as a possible replacement for BP in neural network training by relying on weight perturbations. This removed the truncation error of previous weight perturbation approaches [10] by moving to forward-mode AD. While FGD is a promising BP-free approach, the scaling of FGD to high-dimensional models is challenging due the variance of the gradient estimator (see derivation in §B.4). This challenge has led to multiple efforts that have focused on reducing this variance [5, 13, 16]. In particular, a common approach has been to rely on local BP steps to reduce the variance and/or to provide a better guess for the perturbation direction. In our work, we focus on second-order forward-mode optimization approaches, but highlight that these other approaches on variance reduction are orthogonal to our approach and could also be combined together and generalized to FoMoH-KD.

3. Forward-Mode Automatic Differentiation

In this section we focus on forward-mode AD and build to second-order forward-mode AD. For a complete introduction, please refer to Griewank and Walther [7]. $\mathbf{x} \in \mathbb{R}^D$ denotes a column vector.

Forward-Mode AD [18] applies the chain rule in the *forward* direction. The forward-mode evaluation, $F(\boldsymbol{\theta}, \mathbf{v})$, requires an additional tangent vector, $\mathbf{v} \in \mathbb{R}^D$, along with the parameter vector $\boldsymbol{\theta} \in \mathbb{R}^D$ for a function $\boldsymbol{f} : \mathbb{R}^D \to \mathbb{R}^O$. The result of the evaluation, $[\boldsymbol{f}(\boldsymbol{\theta}), \nabla \boldsymbol{f}(\boldsymbol{\theta})\mathbf{v}]$, is a function evaluation and the corresponding Jacobian vector product (JVP), where $\nabla \boldsymbol{f}(\boldsymbol{\theta}) \in \mathbb{R}^{O \times D}$. For a unidimensional output function, the JVP is the directional derivative, $\nabla \boldsymbol{f}(\boldsymbol{\theta}) \cdot \mathbf{v}$. The time and space (memory cost) complexity are linear, both approximately twice that of a single function evaluation.[1] A common implementation of forward-mode AD is to use dual numbers. A dual number $a + b\epsilon \in \mathbb{D}(\mathbb{R})$ contains a real (primal) component, $a \in \mathbb{R}$, and a dual component, $b \in \mathbb{R}$. We can think of this as representing a truncated Taylor series, $a + b\epsilon + \mathcal{O}(\epsilon^2)$, notationally simplified by the rule $\epsilon^2 = 0$. Using this, $f(a + b\epsilon) = f(a) + \nabla f(a) b\epsilon$. A simple example can be shown for the function, $f(a_1, a_2) = a_1 \times a_2$. Using dual numbers, $(a_1 + b_1\epsilon)(a_2 + b_2\epsilon)$, we retrieve the function evaluation and the corresponding familiar product rule: $a_1 a_2 + (a_1 b_2 + b_1 a_2)\epsilon$. This can be extended to multiple dimensions, and is the basis of forward-mode AD: lift all real numbers $\mathbb{R}$ to dual numbers $\mathbb{D}(\mathbb{R})$.

Higher-Order Forward-Mode AD extends dual numbers to truncate at a higher order, or to not truncate at all; and to allow nesting by supporting multiple distinct formal ϵ variables [11]. Specifically focusing on second-order terms is often referred to as hyper-dual numbers (for example in the Aeronautics and Astronautics community [4]). A hyper-dual number is made up from four components, which is written as $\boldsymbol{\theta} + \mathbf{v}_1\epsilon_1 + \mathbf{v}_2\epsilon_2 + \mathbf{v}_{12}\epsilon_1\epsilon_2$. In the same manner that we look at imaginary and real parts of complex numbers, we can look at the first derivative parts of a hyper-dual number by inspecting the ϵ_1 and ϵ_2 components, and we can look at the second derivative by inspecting the $\epsilon_1\epsilon_2$ component. To understand how this formulation arises, we can introduce the definitions $\epsilon_1^2 = \epsilon_2^2 = (\epsilon_1\epsilon_2)^2 = 0$ and replace the Taylor series expansion of a function, $f : \mathbb{R}^D \to \mathbb{R}$ around $\boldsymbol{\theta}$ with an evaluation of a hyper-dual number:

$$\begin{aligned} f(\boldsymbol{\theta} + \mathbf{v}_1\epsilon_1 + \mathbf{v}_2\epsilon_2 + \mathbf{v}_{12}\epsilon_1\epsilon_2) = f(\boldsymbol{\theta}) &+ \nabla f(\boldsymbol{\theta})\mathbf{v}_1\epsilon_1 + \nabla f(\boldsymbol{\theta})\mathbf{v}_2\epsilon_2 \\ &+ \nabla f(\boldsymbol{\theta})\mathbf{v}_{12}\epsilon_1\epsilon_2 + \mathbf{v}_1^\top \nabla^2 f(\boldsymbol{\theta})\mathbf{v}_2\epsilon_1\epsilon_2 + \cdots . \end{aligned}$$

An alternative but isomorphic view is to regard $a + b\epsilon_1 + c\epsilon_2 + d\epsilon_1\epsilon_2 = (a + b\epsilon_1) + (c + d\epsilon_1)\epsilon_2$ as an element of $\mathbb{D}(\mathbb{D}(\mathbb{R}))$, with subscripts to distinguish the inner vs outer $\mathbb{D}$s; from an implementation perspective, hyperduals can be regarded as inlining the nested structures into a single flat structure.

4. Forward-Mode Optimization with Second Order Information

We now introduce FoMoH-KD, our new Forward-Mode Second-Order Hyperplane Search. Rather than performing a second order line search along direction $\mathbf{v}$, we perform a K-dimensional hyperplane search. Starting with the $K = 2$ example, if we take two search directions, $\mathbf{v}_1$ and $\mathbf{v}_2$, we build a 2×2 matrix to form a Hessian in the affine hyperplane defined by $\boldsymbol{\theta} + \kappa_1\mathbf{v}_1 + \kappa_2\mathbf{v}_2$. We evaluate a function, $f(\cdot)$, with a hyper-dual number using the pairs $\{\mathbf{v}_1, \mathbf{v}_1\}$, $\{\mathbf{v}_1, \mathbf{v}_2\}$, and $\{\mathbf{v}_2, \mathbf{v}_2\}$ for the ϵ_1, ϵ_2 coefficients. The result is the Hessian, $\tilde{\mathbf{H}}_{2\times2}$, in the 2×2 plane, and corresponding step sizes, κ_1 and κ_2, to take in each search direction for minimization:

$$\tilde{\mathbf{H}}_{2\times2} = \begin{bmatrix} \mathbf{v}_1^\top \nabla^2 f(\boldsymbol{\theta})\mathbf{v}_1 & \mathbf{v}_1^\top \nabla^2 f(\boldsymbol{\theta})\mathbf{v}_2 \\ \mathbf{v}_2^\top \nabla^2 f(\boldsymbol{\theta})\mathbf{v}_1 & \mathbf{v}_2^\top \nabla^2 f(\boldsymbol{\theta})\mathbf{v}_2 \end{bmatrix}, \qquad \begin{bmatrix} \kappa_1 \\ \kappa_2 \end{bmatrix} = -\tilde{\mathbf{H}}_{2\times2}^{-1} \begin{bmatrix} \mathbf{v}_1^\top \nabla f(\boldsymbol{\theta}) \\ \mathbf{v}_2^\top \nabla f(\boldsymbol{\theta}) \end{bmatrix} \tag{1}$$

1. The basic time complexity of a forward evaluation is constant $\in [2, 2.5]$ times that of the function call [7].

As a result, we formulate a new update step,

$$\boldsymbol{\theta}' = \boldsymbol{\theta} + \kappa_1 \mathbf{v}_1 + \kappa_2 \mathbf{v}_2.$$

We then extend the above result to any K-dimensional hyperplane by sampling K search directions and evaluating the corresponding update step:

$$\boldsymbol{\theta}' = \boldsymbol{\theta} + \eta \sum_{k=1}^{K} \kappa_k \mathbf{v}_k. \tag{2}$$

This resulting generalized hyperplane update step allows one to trade-off computational cost with the size of the search space. For example, the cost of evaluating a K-dimensional Hessian and then invert it is $\mathcal{O}(K^3)$, which is feasible for small enough K.[2] Our new forward-mode hyperplane search, FoMoH-KD, opens up the possibility of transitioning between a line search, when $K = 1$, all the way to a full Newton step, when $K = D$, where we include a theorem in §4.1 (with proof in §B.1) and demonstrate empirically in §5. The pseudo-code for a single update step is given in Algorithm 1. The overall FoMoH-KD routine is given in Algorithm 2.

4.1. Convergence Properties of FoMoH-KD

Here, we will simply state convergence properties of FoMoH-KD, where we provide proofs in §B.

Theorem 1 *Let $f : \mathbb{R}^D \to \mathbb{R}$ be a convex quadratic function with a global minimizer $\boldsymbol{\theta}^*$. Then $\min_{t \in \{1,\dots,\infty\}} \mathbb{E}[\boldsymbol{\theta}_t] = \boldsymbol{\theta}^*$ (see §B.1).*

Theorem 2 *Let $f : \mathbb{R}^D \to \mathbb{R}$ be a convex quadratic function with a global minimizer $\boldsymbol{\theta}^*$ and Hessian $2\mathbf{Q}$. Then the norm of the expected error converges geometrically with bounds,*

$$\frac{D-K}{D}\left(\sqrt{\text{cond}(\mathbf{Q})}\right)^{-1} \|\mathbf{e}_t\|_2 \leq \|\mathbb{E}[\mathbf{e}_{t+1}]\|_2 \leq \frac{D-K}{D}\sqrt{\text{cond}(\mathbf{Q})}\|\mathbf{e}_t\|_2,$$

that depend on the condition number of $\mathbf{Q}$ where $\|\mathbf{e}_{t+1}\| = \|\boldsymbol{\theta}_{t+1} - \boldsymbol{\theta}^*\|$ and K is the dimension of the hyperplane (see §B.1).

As a direct consequence of Theorem 2, we have the following corollaries:

Corollary 3 *When $D = K$, the update step of FoMoH-KD follows Newton's method.*

Corollary 4 *FoMoH-KD's convergence is exponential in expectation with the rate $\frac{D-K}{D}$.*

Corollary 4 can be directly compared with the sub-linear convergence (in expectation) of FGD that we derive in §B.5, where we also derive the variance in FGD's estimator.

2. For instances where $\tilde{\mathbf{H}}_{K\times K}$ is not invertible, we add jitter to the diagonal. This seems to work well.

5. Experiments

In this section, we summarize the main empirical results, where we include all the details in §C.

Rosenbrock Function [14] is a challenging non-convex optimization problem. Figure 1 illustrates the behavior of a single update step in expectation. This figure highlights the sensitivity of first-order FGD to the step size and shows that for the 2D function, FoMoH-2D takes the same step as Newton's method. The figure also overlays the scatter of the individual sampled update steps. Figure 2 shows the performance of FoMoH-KD as we increase K from 2 to the input dimension of the function. Figure 2 shows this comparison for the 10D Rosenbrock function, where we use 10 random initializations for the different K. The median performance for each K is then shown, where we see a perfect ordering of performance that aligns with the dimension of the hyperplane. The best performing FoMoH-KD is for $K = D$, directly aligning with Newton's method, consistent with Corollary 3. Further experiments in §C compare to reverse-mode optimization approaches.

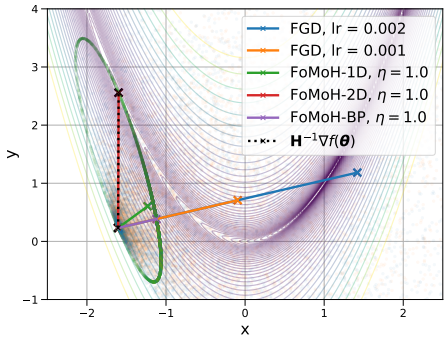

Figure 1: Comparison of forward-mode update steps.

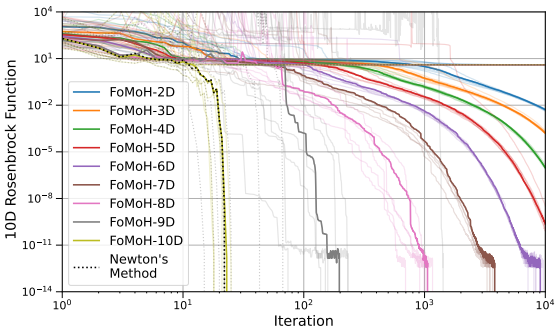

Figure 2: FoMoH-KD's generalization to Newton's method.

Multinomial Logistic Regression increases the dimension of the optimization problem to 784 when applied to the MNIST dataset [9]. Table 1 and Figure 4 in the Appendix display the training and validation performance for the forward-mode approaches as well as the additional reverse-mode approaches. We see an improvement in speed of convergence as K increases. However, we also see that FoMoH-KD, with a fixed learning rate, can degrade in performance after reaching a local minimum. We therefore introduce a learning rate scheduler to improve on this behavior. For this task, FGD is competitive with the FoMoH variants but is slower to converge.

Convolutional Neural Network (CNN) applied to MNIST has 431,080 parameters. Both Table 3 and Figure 6 in the appendix highlight the advantage of FoMoH-KD over FGD. For the larger parameter space FGD requires more epochs to converge compared to all FoMoH variants. The learning rate scheduler further improves FoMoH and FoMoH-KD by helping to avoid getting stuck in low performance regions. Here, we see the clear trend that the best performing **forward-mode-only** approach comes from the largest K, which was $K = 3$ for this experiment. Overall, these results highlight that second-order information helps scale the performance of forward-mode optimization to larger dimensions.

6. Conclusion

The results in §5 highlight the potential of the use of second-order forward-mode AD for optimization tasks. For the Rosenbrock function, we directly showed how FoMoH-KD tends to Newton's method, which is consistent with our theory. This significant result is shown in Figure 2. For logistic regression and CNN classification, we see how the first-order optimization approach of FGD degrades with increasing dimension of the parameter space. We do not see this degradation for FoMoH-KD, and we also observe that the second-order information means fewer epochs are needed to reach a better performance. This has the broader impact of improving efficiency, reducing cost, and increasing accuracy in ML optimization routines. In conclusion, we introduced a novel approach that uses second-order forward-mode AD for optimization and demonstrated strong empirical performance compared to FGD. We also made a contribution to the theory and we provide the Python package, `Fomoh`,[3] that implements the AD backend and interfaces with PyTorch.

Acknowledgments

This work was supported in part by the United States Air Force and Defense Advanced Research Projects Agency (DARPA) under Contract No. FA8750-23-C-0519, the Defense Advanced Research Projects Agency (DARPA) under Agreement No. HR0011-24-9-0424, and the U.S. Army Research Laboratory Cooperative Research Agreement W911NF-17-2-0196. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the Department of Defense or the United States Government.

References

[1] Atılım Güneş Baydin, Barak A Pearlmutter, Don Syme, Frank Wood, and Philip Torr. Gradients without backpropagation. *arXiv preprint arXiv:2202.08587*, 2022.

[2] Yoshua Bengio, Dong-Hyun Lee, Jorg Bornschein, Thomas Mesnard, and Zhouhan Lin. Towards biologically plausible deep learning. *arXiv preprint arXiv:1502.04156*, 2015.

[3] Lukas Biewald. Experiment tracking with weights and biases, 2020. URL https://www.wandb.com/. Software available from wandb.com.

[4] Jeffrey Fike and Juan Alonso. The development of hyper-dual numbers for exact second-derivative calculations. In *49th AIAA aerospace sciences meeting including the new horizons forum and aerospace exposition*, page 886.

[5] Louis Fournier, Stéphane Rivaud, Eugene Belilovsky, Michael Eickenberg, and Edouard Oyallon. Can forward gradient match backpropagation? In *International Conference on Machine Learning*, pages 10249–10264. PMLR, 2023.

[6] William Fulton and Joe Harris. *Representation theory: a first course*, volume 129. Springer Science & Business Media, 2013.

[7] Andreas Griewank and Andrea Walther. *Evaluating derivatives: principles and techniques of algorithmic differentiation*. SIAM, 2008.

3. https://github.com/SRI-CSL/fomoh

[8] Geoffrey Hinton. The forward-forward algorithm: Some preliminary investigations. *arXiv preprint arXiv:2212.13345*, 2022.

[9] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[10] Barak A Pearlmutter. Fast exact multiplication by the Hessian. *Neural Computation*, 6(1): 147–160, 1994.

[11] Barak A Pearlmutter and Jeffrey Mark Siskind. Lazy multivariate higher-order forward-mode AD. *ACM SIGPLAN Notices*, 42(1):155–160, 2007.

[12] D Pierangeli, G Marcucci, and C Conti. Large-scale photonic ising machine by spatial light modulation. *Physical review letters*, 122(21):213902, 2019.

[13] Mengye Ren, Simon Kornblith, Renjie Liao, and Geoffrey Hinton. Scaling forward gradient with local losses. *arXiv preprint arXiv:2210.03310*, 2022.

[14] H. Rosenbrock. An automatic method for finding the greatest or least value of a function. *The Computer Journal*, 3(3):175–184, 1960.

[15] Mark Schmidt. Gradient descent convergence l4. University of British Columbia, 2019. URL https://www.cs.ubc.ca/~schmidtm/Courses/540-W19/L4.pdf. Accessed: 2024-08-27.

[16] David Silver, Anirudh Goyal, Ivo Danihelka, Matteo Hessel, and Hado van Hasselt. Learning by directional gradient descent. In *International Conference on Learning Representations*, 2021.

[17] Roman Vershynin. *High-dimensional probability: An introduction with applications in data science*, volume 47. Cambridge university press, 2018.

[18] Robert Edwin Wengert. A simple automatic derivative evaluation program. *Communications of the ACM*, 7(8):463–464, 1964.

Supplementary Materials

Appendix A. FoMoH-KD Algorithm

Algorithm 1 FoMoH-KD, hyperplane update step for function, f, and parameters, $\boldsymbol{\theta} \in \mathbb{R}^D$.

Define: HyperPlaneStep$(f, \boldsymbol{\theta}, K)$
  Set: $N = (K^2 + K)/2$, $\boldsymbol{\Theta} \in \mathbb{R}^{N \times D}$, $\boldsymbol{V} \in \mathbb{R}^{K \times D}$, $\boldsymbol{V}_1 \in \mathbb{R}^{N \times D}$, $\boldsymbol{V}_2 \in \mathbb{R}^{N \times D}$
  % For loop vectorized in code.
  for $n = 1, \ldots, N$ **do**
    $\boldsymbol{\Theta}[n, :] = \boldsymbol{\theta}$ % Repeat current parameter values for vectorized evaluation.
  end for
  for $k = 1, \ldots, K$ **do**
    $\boldsymbol{V}[k, :] = \mathbf{v} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ % Sample K tangent vectors to build hyperplane.
  end for
  % Vectorize tangent vectors for evaluation of all N elements of $\tilde{\mathbf{H}}_{K \times K}$.
  $l = 1$
  for $i = 1, \ldots, K$ **do**
    for $j = i, \ldots, K$ **do**
      $\boldsymbol{V}_1[l, :] = \boldsymbol{V}[i, :]$
      $\boldsymbol{V}_2[l, :] = \boldsymbol{V}[j, :]$
      $l = l + 1$
    end for
  end for
  $\mathbf{z}_0 + \mathbf{z}_1\epsilon_1 + \mathbf{z}_2\epsilon_2 + \mathbf{z}_{12}\epsilon_1\epsilon_2 = f(\boldsymbol{\Theta} + \boldsymbol{V}_1\epsilon_1 + \boldsymbol{V}_2\epsilon_2 + \mathbf{0}\epsilon_1\epsilon_2)$
  % Build $\tilde{\mathbf{H}}_{K \times K}$ and directional derivatives vector, $\tilde{\mathbf{G}}_K$, in order to evaluate Eq. (2).
  Set: $\tilde{\mathbf{H}}_{K \times K} \in \mathbb{R}^{K \times K}$, $\tilde{\mathbf{G}}_K \in \mathbb{R}^{K \times 1}$
  $l = 1$
  for $i = 1, \ldots, K$ **do**
    $\tilde{\mathbf{G}}_K[i] = \mathbf{z}_1[l]$
    for $j = i, \ldots, K$ **do**
      $\tilde{\mathbf{H}}_{K \times K}[i, j] = \mathbf{z}_{12}[k]$
      $\tilde{\mathbf{H}}_{K \times K}[j, i] = \mathbf{z}_{12}[k]$
      $l = l + 1$
    end for
  end for
  % Returns update direction of vector size D
  return $\sum_k ((-\tilde{\mathbf{H}}^{-1}_{K \times K} \tilde{\mathbf{G}}_K)[k] \cdot \boldsymbol{V}[k, :])$

Appendix B. Theory

The first two subsections are concerned with proving Theorem's 1 and 2 for FoMoH-KD. The latter two are concerned with the variance and convergence properties of FGD..

Algorithm 2 FoMoH-KD

Require: Objective function $f(\boldsymbol{\theta})$, initial point $\boldsymbol{\theta}$, step size η, hyperplane dimension K.
  for $t = 0, 1, 2, \ldots$ until convergence **do**
    $\mathbf{d} = \text{HyperPlaneStep}(f, \boldsymbol{\theta}, K)$ % See Algorithm 1
    $\boldsymbol{\theta} = \boldsymbol{\theta} + \eta\mathbf{d}$
  end for

B.1. Convergence Analysis of FoMoH-KD

Theorem 1 *Let $f : \mathbb{R}^D \to \mathbb{R}$ be a convex quadratic function with a global minimizer $\boldsymbol{\theta}^*$. Then* $\min_{t \in \{1,\ldots,\infty\}} \mathbb{E}[\boldsymbol{\theta}_t] = \boldsymbol{\theta}^*$.

Proof Assuming a convex quadratic function $f : \mathbb{R}^D \to \mathbb{R}$, of form $f(\boldsymbol{\theta}) = \boldsymbol{\theta}^\top \mathbf{Q}\boldsymbol{\theta} + \mathbf{b}^\top\boldsymbol{\theta}$ and defining an affine subspace,

$$\mathcal{V}_{\mathbf{x}}^K = \left\{ \mathbf{x} + \sum_{k=1}^{K} \kappa_k \mathbf{v}_k \middle| k \in \mathbb{N}, \mathbf{v}_k \in \mathbb{R}^D, \kappa \in \mathbb{R}, \mathbf{x} \in \mathbb{R}^D \right\},$$

we derive a new function $g(\boldsymbol{\kappa})$ for $\boldsymbol{\kappa} \in \mathbb{R}^K$ that represents $f(\boldsymbol{\theta})$ for when $\boldsymbol{\theta} \in \mathcal{V}_{\mathbf{x}}^K$ ($\boldsymbol{\theta}$ restricted to the hyperplane[4]). As a result, the linear map is given by $\boldsymbol{\theta} = \mathbf{V}\boldsymbol{\kappa}$, where $\mathbf{V} \in \mathbb{R}^{D\times K}$.

We explicitly write the FoMoH-KD update step from (1) as:

$$\begin{aligned}\boldsymbol{\kappa}_{t+1} &= \boldsymbol{\kappa}_t - \left([\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_K]^\top \nabla^2 f(\boldsymbol{\theta}_t)[\mathbf{v}_1, ..., \mathbf{v}_K]\right)^{-1} [\mathbf{v}_1 \cdot \nabla f(\boldsymbol{\theta}_t), ..., \mathbf{v}_K \cdot \nabla f(\boldsymbol{\theta}_t)]^\top \\ &= \boldsymbol{\kappa}_t - \left(\mathbf{V}^\top \nabla^2 f(\boldsymbol{\theta}_t)\mathbf{V}\right)^{-1} \mathbf{V}^\top \nabla f(\boldsymbol{\theta}_t),\end{aligned}$$

which is equivalent to Newton's method in $\boldsymbol{\kappa}$:

$$\boldsymbol{\kappa}_{t+1} = \boldsymbol{\kappa}_t - \left(\nabla^2 g(\boldsymbol{\kappa}_t)\right)^{-1} \nabla g(\boldsymbol{\kappa}_t),$$

where each k^{th} component of $\nabla g(\boldsymbol{\kappa}_t)$ corresponds to the derivative of $f(\boldsymbol{\theta}_t)$ along $\mathbf{v}_k$ by definition. Following (2), we use $\mathbf{V}$ to project back to $\boldsymbol{\theta}$:

$$\begin{aligned}\mathbf{V}\boldsymbol{\kappa}_{t+1} &= \mathbf{V}\boldsymbol{\kappa}_t - \mathbf{V}\left(\mathbf{V}^\top \nabla^2 f(\boldsymbol{\theta}_t)\mathbf{V}\right)^{-1} \mathbf{V}^\top \nabla f(\boldsymbol{\theta}_t), \\ \boldsymbol{\theta}_{t+1} &= \boldsymbol{\theta}_t - \mathbf{V}\left(\mathbf{V}^\top \nabla^2 f(\boldsymbol{\theta}_t)\mathbf{V}\right)^{-1} \mathbf{V}^\top \nabla f(\boldsymbol{\theta}_t). \end{aligned} \tag{3}$$

The global minmizer $\boldsymbol{\theta}^*$ occurs at $-\frac{1}{2}\mathbf{Q}^{-1}\mathbf{b}$ for the quadratic function, $f(\boldsymbol{\theta}) = \boldsymbol{\theta}^\top\mathbf{Q}\boldsymbol{\theta} + \mathbf{b}^\top\boldsymbol{\theta}$. The minimizer in the hyperplane with the change of variables occurs at $\boldsymbol{\kappa}^* = -\left(\mathbf{V}^\top 2\mathbf{Q}\mathbf{V}\right)^{-1}\mathbf{V}^\top\mathbf{b}$. For a quadratic convex f, g is also convex and quadratic. Therefore, Newton's method in $\boldsymbol{\kappa}$, results in a single step to the minimizer in the sampled hyperplane such that $\boldsymbol{\kappa}_{t+1} = \boldsymbol{\kappa}^*$. We use the spanning matrix, $\mathbf{V} \in \mathbb{R}^{D\times K}$, to transform from $\boldsymbol{\kappa}_{t+1} \in \mathbb{R}^K$, into $\mathbb{R}^D$, such that $\boldsymbol{\theta}_{t+1} = \mathbf{V}^\top\boldsymbol{\kappa}_{t+1}$.

4. E.g., for $K = 2$, $g(\kappa_1, \kappa_2) = f(\mathbf{x} + \kappa_1\mathbf{v}_1 + \kappa_2\mathbf{v}_2)$

At location $\boldsymbol{\kappa}_t$, corresponding to $\boldsymbol{\theta}_t$, if the column vectors of $\mathbf{V}$ are sampled such that $\boldsymbol{\theta}^* \in \mathcal{V}^K_{\boldsymbol{\theta}_t}$, then $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}^*$ as $\boldsymbol{\theta}_{t+1} = \mathbf{V}\boldsymbol{\kappa}^*$. Since the span is sampled according to an isotropic Gaussian distribution, $\mathbf{v}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_D)$, there is full support over the entire unit sphere, which gives the following FoMoH-KD convergence result,

$$\min_{t \in \{1,\dots,\infty\}} \mathbb{E}[\boldsymbol{\theta}_t] = \boldsymbol{\theta}^*.$$

■

B.2. Convergence Rate Analysis of FoMoH-KD

After showing that FoMoH-KD is guaranteed to converge for a convex quadratic function, the next step is to analyze how the convergence rate is affected by the size of the span, K, and the parameter dimension, D. Restating Theorem 2:

Theorem 2 *Let $f : \mathbb{R}^D \to \mathbb{R}$ be a convex quadratic function with a global minimizer $\boldsymbol{\theta}^*$ and Hessian $2\mathbf{Q}$. Then the norm of the expected error converges geometrically with bounds,*

$$\frac{D-K}{D}\left(\sqrt{\text{cond}(\mathbf{Q})}\right)^{-1}\|\mathbf{e}_t\|_2 \leq \|\mathbb{E}[\mathbf{e}_{t+1}]\|_2 \leq \frac{D-K}{D}\sqrt{\text{cond}(\mathbf{Q})}\|\mathbf{e}_t\|_2,$$

that depend on the condition number of $\mathbf{Q}$ where $\|\mathbf{e}_{t+1}\| = \|\boldsymbol{\theta}_{t+1} - \boldsymbol{\theta}^*\|$ and K is the dimension of the hyperplane.

Proof Assuming a convex quadratic function $f : \mathbb{R}^D \to \mathbb{R}$, of form $f(\boldsymbol{\theta}) = \boldsymbol{\theta}^\top\mathbf{Q}\boldsymbol{\theta} + \mathbf{b}^\top\boldsymbol{\theta}$, we measure distance on the function space using the Hessian, $\mathbf{H} = 2\mathbf{Q}$, as the metric. Since $\mathbf{Q}$ is positive definite, we can diagonalize $\mathbf{Q}$ using an eigenvalue decomposition, $\mathbf{Q} = \mathbf{G}\boldsymbol{\Lambda}\mathbf{G}^\top$. As such, without loss of generality, we transform our variables to $\tilde{\boldsymbol{\theta}} = \boldsymbol{\Lambda}^{1/2}\mathbf{G}^\top\boldsymbol{\theta}$. In the transformed space, the location of the closest point, $\mathbf{p}$, in a randomly sampled affine hyperplane, $\mathcal{V}^K_{\tilde{\boldsymbol{\theta}}_t}$ to the point $\tilde{\boldsymbol{\theta}}^*$, is given by $\mathbf{p} = \tilde{\boldsymbol{\theta}}_t + \mathbf{P}(\tilde{\boldsymbol{\theta}}^* - \tilde{\boldsymbol{\theta}}_t)$, where $\mathbf{P}$ is the projection matrix defined by $\mathbf{P} = \mathbf{V}(\mathbf{V}^\top\mathbf{V})^{-1}\mathbf{V}^\top$.

We now show that the closest point on the affine hyperplane to $\tilde{\boldsymbol{\theta}}^*$, is equivalent to the FoMoH-KD update step, where $\mathbf{p} = \mathbf{V}\tilde{\boldsymbol{\kappa}}_{t+1} = \tilde{\boldsymbol{\theta}}_{t+1}$. We start from the FoMoH-KD update step (see (3)) applied to the transformed space $\tilde{\boldsymbol{\theta}}$:

$$\tilde{\boldsymbol{\theta}}_{t+1} = \tilde{\boldsymbol{\theta}}_t - \mathbf{V}\left(\mathbf{V}^\top\nabla^2 f(\tilde{\boldsymbol{\theta}}_t)\mathbf{V}\right)^{-1}\mathbf{V}^\top\nabla f(\tilde{\boldsymbol{\theta}}_t),$$

and then using $\nabla^2 f(\tilde{\boldsymbol{\theta}}_t) = 2\mathbf{I}$, and $\nabla f(\tilde{\boldsymbol{\theta}}_t) = 2\tilde{\boldsymbol{\theta}}_t + \mathbf{b}^\top\mathbf{G}\boldsymbol{\Lambda}^{-1/2}$ gives:

$$\tilde{\boldsymbol{\theta}}_{t+1} = \tilde{\boldsymbol{\theta}}_t - \frac{1}{2}\mathbf{V}\left(\mathbf{V}^\top\mathbf{V}\right)^{-1}\mathbf{V}^\top\left(2\tilde{\boldsymbol{\theta}}_t + \mathbf{b}^\top\mathbf{G}\boldsymbol{\Lambda}^{-1/2}\right),$$

Finally, noting that $\tilde{\boldsymbol{\theta}}^* = -\frac{1}{2}\mathbf{b}^\top\mathbf{G}\boldsymbol{\Lambda}^{-1/2}$ from setting the gradient to zero, we get

$$\begin{aligned}\tilde{\boldsymbol{\theta}}_{t+1} &= \tilde{\boldsymbol{\theta}}_t - \frac{1}{2}\mathbf{V}\left(\mathbf{V}^\top\mathbf{V}\right)^{-1}\mathbf{V}^\top\left(2\tilde{\boldsymbol{\theta}}_t - 2\tilde{\boldsymbol{\theta}}^*\right),\\ &= \tilde{\boldsymbol{\theta}}_t + \mathbf{P}\left(\tilde{\boldsymbol{\theta}}^* - \tilde{\boldsymbol{\theta}}_t\right).\end{aligned}$$

Therefore, we have arrived at the same update rule as in (2) for FoMoH-KD via an alternative route that uses knowledge of the true minimizer along with the use of projection matrices. We can use the equivalence of this rule, $\tilde{\boldsymbol{\theta}}_{t+1} = \tilde{\boldsymbol{\theta}}_t + \mathbf{P}(\tilde{\boldsymbol{\theta}}^* - \tilde{\boldsymbol{\theta}}_t)$, to show how the convergence rate of FoMoH-KD relates to K and D. We use the definition of the expectation of the projection matrix $\mathbb{E}[\mathbf{P}] = \frac{K}{D}\mathbf{I}$ (see §B.3 for proof), such that

$$\begin{aligned}\mathbb{E}[\tilde{\boldsymbol{\theta}}_{t+1}] = &\quad \mathbb{E}[\tilde{\boldsymbol{\theta}}_t + \mathbf{P}(\tilde{\boldsymbol{\theta}}^* - \tilde{\boldsymbol{\theta}}_t)] \\ = &\quad \tilde{\boldsymbol{\theta}}_t + \frac{K}{D}(\tilde{\boldsymbol{\theta}}^* - \tilde{\boldsymbol{\theta}}_t). \end{aligned} \tag{4}$$

This key result is consistent with our observation that FoMoH-KD generalizes to Newton's method, as when $K = D$ Equation (4) reduces to $\tilde{\boldsymbol{\theta}}_{t+1} = \tilde{\boldsymbol{\theta}}^*$, and therefore $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}^*$ (See Corollary 3). Furthermore, we remove $\tilde{\boldsymbol{\theta}}^*$ from both sides of the equation and transform back to the original parameter space:

$$\begin{aligned}\mathbb{E}[\tilde{\boldsymbol{\theta}}_{t+1} - \tilde{\boldsymbol{\theta}}^*] =& \tilde{\boldsymbol{\theta}}_t + \frac{K}{D}(\tilde{\boldsymbol{\theta}}^* - \tilde{\boldsymbol{\theta}}_t) - \tilde{\boldsymbol{\theta}}^* \\ \|\mathbb{E}[\tilde{\boldsymbol{\theta}}_{t+1} - \tilde{\boldsymbol{\theta}}^*]\| =& \frac{D-K}{D}\|\tilde{\boldsymbol{\theta}}_t - \tilde{\boldsymbol{\theta}}^*\| \\ \|\mathbb{E}[\boldsymbol{\Lambda}^{1/2}\mathbf{G}^\top\boldsymbol{\theta}_{t+1} - \boldsymbol{\Lambda}^{1/2}\mathbf{G}^\top\boldsymbol{\theta}^*]\| =& \frac{D-K}{D}\|\boldsymbol{\Lambda}^{1/2}\mathbf{G}^\top\boldsymbol{\theta}_t - \boldsymbol{\Lambda}^{1/2}\mathbf{G}^\top\boldsymbol{\theta}^*\| \\ \|\mathbb{E}[\boldsymbol{\Lambda}^{1/2}\mathbf{G}^\top(\boldsymbol{\theta}_{t+1} - \boldsymbol{\theta}^*)]\| =& \frac{D-K}{D}\|\boldsymbol{\Lambda}^{1/2}\mathbf{G}^\top(\boldsymbol{\theta}_t - \boldsymbol{\theta}^*)\| \\ \|\mathbb{E}[\mathbf{e}_{t+1}]\|_{\mathbf{Q}} =& \frac{D-K}{D}\|\mathbf{e}_t\|_{\mathbf{Q}} \end{aligned} \tag{5}$$

This implies that the Euclidean norm of the expected error converges geometrically, but with bounds that depend on the eigenvalues of $\mathbf{Q}$. Using the relation $\sqrt{\lambda_{\min}}\|\mathbf{x}\|_2 \leq \|\mathbf{x}\|_{\mathbf{Q}} \leq \sqrt{\lambda_{\max}}\|\mathbf{x}\|_2$, where $\lambda_{\min}, \lambda_{\max}$ are the maximum and minimum Eigenvalues of $\mathbf{Q}$, we have the following bounds on the norm of the errors, $\mathbf{e}_t$, and $\mathbb{E}[\mathbf{e}_{t+1}]$:

$$\begin{aligned}\frac{D-K}{D}\sqrt{\lambda_{\min}(\mathbf{Q})}\|\mathbf{e}_t\|_2 \leq \frac{D-K}{D}\|\mathbf{e}_t\|_{\mathbf{Q}} \leq \frac{D-K}{D}\sqrt{\lambda_{\max}(\mathbf{Q})}\|\mathbf{e}_t\|_2, \\ \sqrt{\lambda_{\min}(\mathbf{Q})}\|\mathbb{E}[\mathbf{e}_{t+1}]\|_2 \leq \|\mathbb{E}[\mathbf{e}_{t+1}]\|_{\mathbf{Q}} \leq \sqrt{\lambda_{\max}(\mathbf{Q})}\|\mathbb{E}[\mathbf{e}_{t+1}]\|_2. \end{aligned}$$

We use Eq. (5) to rearrange the above inequalities to bound the Euclidean norm of the update step, $\|\mathbb{E}[\mathbf{e}_{t+1}]\|_2$, in terms of the current error norm, giving an upper bound:

$$\begin{aligned}\sqrt{\lambda_{\min}(\mathbf{Q})}\|\mathbb{E}[\mathbf{e}_{t+1}]\|_2 \leq \frac{D-K}{D}\sqrt{\lambda_{\max}(\mathbf{Q})}\|\mathbf{e}_t\|_2, \\ \|\mathbb{E}[\mathbf{e}_{t+1}]\|_2 \leq \frac{D-K}{D}\frac{\sqrt{\lambda_{\max}(\mathbf{Q})}}{\sqrt{\lambda_{\min}(\mathbf{Q})}}\|\mathbf{e}_t\|_2, \end{aligned}$$

and a lower bound:

$$\begin{aligned}\frac{D-K}{D}\sqrt{\lambda_{\min}(\mathbf{Q})}\|\mathbf{e}_t\|_2 \leq \sqrt{\lambda_{\max}(\mathbf{Q})}\|\mathbb{E}[\mathbf{e}_{t+1}]\|_2, \\ \frac{D-K}{D}\frac{\sqrt{\lambda_{\min}(\mathbf{Q})}}{\sqrt{\lambda_{\max}(\mathbf{Q})}}\|\mathbf{e}_t\|_2 \leq \|\mathbb{E}[\mathbf{e}_{t+1}]\|_2, \end{aligned}$$

leading to the final bounds on the Euclidean norm of the expected error at $t+1$, where the condition number, $\text{cond}(\mathbf{Q}) = {}^{\lambda_{\max}(\mathbf{Q})}/_{\lambda_{\min}(\mathbf{Q})}$:

$$\frac{D-K}{D}\left(\sqrt{\text{cond}(\mathbf{Q})}\right)^{-1}\|\mathbf{e}_t\|_2 \leq \|\mathbb{E}[\mathbf{e}_{t+1}]\|_2 \leq \frac{D-K}{D}\left(\sqrt{\text{cond}(\mathbf{Q})}\right)\|\mathbf{e}_t\|_2 \tag{6}$$

■

This result shows that the update step reduces the error by the fraction $0 \leq \frac{D-K}{D} < 1$, with a linear decrease in the rate of reduction in the error as K increases, and with bounds that depend on the geometry of the function. Therefore FoMoH-KD's convergence is exponential in expectation with the rate $\frac{D-K}{D}$. This compares to the sub-linear convergence (in expectation) of FGD that we derive in §B.5.

B.3. Expectation of Projection Matrix

In Theorem 2, we used the identity $\mathbb{E}[\mathbf{P}] = \frac{K}{D}\mathbf{I}_D$, where we subscript the identity matrix with its corresponding dimension. Here, we prove this identity.

Theorem 3 *Let $\mathbf{V} \in \mathbb{R}^{D\times K}$ be a matrix where each element, $\mathbf{V}_{ij} \sim \mathcal{N}(0,1)$. Then the expectation of the projection matrix, $\mathbf{P} = \mathbf{V}(\mathbf{V}^\top\mathbf{V})^{-1}\mathbf{V}^\top$, is given by $\mathbb{E}[\mathbf{P}] = \frac{K}{D}\mathbf{I}_D$.*

Proof Let $\mathbf{R} \in \mathbb{R}^{D\times D}$ be an orthogonal matrix such that $\mathbf{R}\mathbf{R}^\top = \mathbf{I}_D$. Then rotating $\mathbf{V}$ gives $\mathbf{V}' = \mathbf{R}\mathbf{V}$. We then use the rotational invariance property of the normal distribution (see Prop. 3.3.2 of [17]) to give $\mathbf{V} \overset{d}{=} \mathbf{R}\mathbf{V}$, where $\overset{d}{=}$ denotes equality in distribution. We now define a transformed projection matrix:

$$\mathbf{P}' = \mathbf{V}'(\mathbf{V}'^\top\mathbf{V}')^{-1}\mathbf{V}'^\top = \mathbf{R}\mathbf{V}(\mathbf{V}^\top\mathbf{R}^\top\mathbf{R}\mathbf{V})^{-1}\mathbf{V}^\top\mathbf{R}^\top = \mathbf{R}\mathbf{P}\mathbf{R}^\top.$$

Since $\mathbf{V}' \overset{d}{=} \mathbf{V}$, then $\mathbf{P} \overset{d}{=} \mathbf{P}'$, and $\mathbb{E}[\mathbf{P}'] = \mathbb{E}[\mathbf{R}\mathbf{P}\mathbf{R}^\top] = \mathbf{R}\mathbb{E}[\mathbf{P}]\mathbf{R}^\top = \mathbb{E}[\mathbf{P}]$. As a result of the last equation, we get the relation $\mathbf{R}\mathbb{E}[\mathbf{P}] = \mathbb{E}[\mathbf{P}]\mathbf{R}$ by post-multiplying by $\mathbf{R}$. Therefore, $\mathbb{E}[\mathbf{P}]$ commutes with all $\mathbf{R}$ in the orthogonal group. Then using Schur's Lemma (e.g. as stated in §1.2 of [6])

$$\mathbb{E}[\mathbf{P}] = c\mathbf{I}_D$$

for some constant c. To determine c, we take the trace of both sides, and use linearity of expectation to move the trace inside the expectation:

$$\text{tr}(\mathbb{E}[\mathbf{P}]) = \text{tr}(c\mathbf{I}_D)$$
$$\mathbb{E}[\text{tr}(\mathbf{V}(\mathbf{V}^\top\mathbf{V})^{-1}\mathbf{V}^\top)] = cD$$

Then, using the identity $\text{tr}(\mathbf{A}\mathbf{B}) = \text{tr}(\mathbf{B}\mathbf{A})$ for $\mathbf{A} \in \mathbb{R}^{D\times K}$ and $\mathbf{B} \in \mathbb{R}^{K\times D}$:

$$\mathbb{E}[\text{tr}(\mathbf{V}^\top\mathbf{V}(\mathbf{V}^\top\mathbf{V})^{-1})] = cD$$
$$\mathbb{E}[\text{tr}(I_K)] = cD$$
$$K = cD \implies c = K/D$$

Thus,

$$\mathbb{E}[\mathbf{P}] = \frac{K}{D}\mathbf{I}_D$$

■

B.4. Variance Analysis of Forward Gradient Descent

If we start with the definition from [1] of the forward gradient:

$$g_i(\boldsymbol{\theta}) = \frac{\partial f}{\partial \theta_i} v_i^2 + \sum_{j \neq i} \frac{\partial f}{\partial \theta_j} v_i v_j,$$

The expectation is given by $\mathbb{E}\left[g_i(\boldsymbol{\theta})\right] = \frac{\partial f}{\partial \theta_i}$. It is of interest as to how the variance of this estimate behaves, i.e. $\mathrm{Var}\left(g_i(\boldsymbol{\theta})\right)$. Using the definition, $\mathrm{Var}\left(X\right) = \mathbb{E}[X^2] - \mathbb{E}^2[X]$, we can derive each component as:

$$\mathbb{E}^2[g_i(\boldsymbol{\theta})] = \left[\frac{\partial f}{\partial \theta_i}\right]^2,$$

and

$$\mathbb{E}[g_i(\boldsymbol{\theta})^2] = \mathbb{E}\left[\left[\frac{\partial f}{\partial \theta_i}\right]^2 v_i^4 + 2\frac{\partial f}{\partial \theta_i}\sum_{j \neq i}\frac{\partial f}{\partial \theta_j} v_i^3 v_j + \left(\sum_{j \neq i}\frac{\partial f}{\partial \theta_j} v_i v_j\right)^2\right].$$

We can expand the last term as:

$$\left(\sum_{j \neq i}\frac{\partial f}{\partial \theta_j} v_i v_j\right)^2 = v_i^2\left(\sum_{j \neq i}\left[\frac{\partial f}{\partial \theta_j}\right]^2 v_j^2 + 2\sum_{j \neq i}\sum_{k \neq i, k > j}\frac{\partial f}{\partial \theta_j}\frac{\partial f}{\partial \theta_k} v_j v_k\right),$$

then moving the expectation inside the square brackets and using the identities, $\mathbb{E}[X^2] = 1, \mathbb{E}[X^4] = 3, \mathbb{E}[X^3 Y] = 0, \mathbb{E}[X^2 Y^2] = 1$, and $\mathbb{E}[X^2 Y Z] = 0$ for the normal distribution gives:

$$\begin{aligned}\mathbb{E}[g_i(\boldsymbol{\theta})^2] &= \left[\frac{\partial f}{\partial \theta_i}\right]^2 [3] + 2\frac{\partial f}{\partial \theta_i}\sum_{j \neq i}\frac{\partial f}{\partial \theta_j}[0] + \sum_{j \neq i}\left[\frac{\partial f}{\partial \theta_j}\right]^2 [1] + 2\sum_{j \neq i}\sum_{k \neq i, k > j}\frac{\partial f}{\partial \theta_j}\frac{\partial f}{\partial \theta_k}[0] \\ &= 3\left[\frac{\partial f}{\partial \theta_i}\right]^2 + \sum_{j \neq i}\left[\frac{\partial f}{\partial \theta_j}\right]^2.\end{aligned}$$

Finally, the variance is given by

$$\mathrm{Var}\left(g_i(\boldsymbol{\theta})\right) = 2\left[\frac{\partial f}{\partial \theta_i}\right]^2 + \sum_{j \neq i}\left[\frac{\partial f}{\partial \theta_j}\right]^2.$$

B.5. Convergence Analysis of FGD

Using the following assumptions:

- $f : \mathbb{R}^D \to \mathbb{R}$ is a convex function with minimum $f(\boldsymbol{\theta}^*)$.
- The update step (minimization) is: $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \mathbf{g}(\boldsymbol{\theta})$.
- $\mathbb{E}\left[\mathbf{g}(\boldsymbol{\theta})\right] = \nabla f(\boldsymbol{\theta}), \mathbb{E}\left[\mathbf{g}(\boldsymbol{\theta})^2\right] \leq G^2$ for some constant G. We derived this form above.
- f has an L-Lipschitz continuous gradient: $||\nabla f(\mathbf{x}) - \nabla f(\mathbf{y})|| \leq L||\mathbf{x} - \mathbf{y}||$.

We start with a first-order Taylor series expansion and input the update rule:

$$f(\boldsymbol{\theta}_{t+1}) \approx f(\boldsymbol{\theta}_t) + \nabla f(\boldsymbol{\theta}_t) \cdot (\boldsymbol{\theta}_{t+1} - \boldsymbol{\theta}_t) + \frac{1}{2}(\boldsymbol{\theta}_{t+1} - \boldsymbol{\theta}_t)^\top \nabla^2 f(\boldsymbol{\theta}_t)(\boldsymbol{\theta}_{t+1} - \boldsymbol{\theta}_t)$$
$$\approx f(\boldsymbol{\theta}_t) - \eta \nabla f(\boldsymbol{\theta}_t) \cdot \mathbf{g}(\boldsymbol{\theta}_t) + \frac{1}{2}\eta^2 \mathbf{g}(\boldsymbol{\theta}_t)^\top \nabla^2 f(\boldsymbol{\theta}_t)\mathbf{g}(\boldsymbol{\theta}_t).$$

Since the rate of change of the gradient is bounded due to the L-Lipschitz continuous gradient assumption, the curvature along any direction follows $\mathbf{g}(\boldsymbol{\theta}_t)^\top \nabla^2 f(\boldsymbol{\theta}_t)\mathbf{g}(\boldsymbol{\theta}_t) \leq L\|\mathbf{g}(\boldsymbol{\theta}_t)\|^2$. The next step is to combine this upper bound on the curvature with expectation operator to give the expected decrease in the function value:

$$\mathbb{E}\left[f(\boldsymbol{\theta}_{t+1})|\boldsymbol{\theta}_t\right] \leq \mathbb{E}\left[f(\boldsymbol{\theta}_t) - \eta \nabla f(\boldsymbol{\theta}_t) \cdot \mathbf{g}(\boldsymbol{\theta}_t) + \frac{L\eta^2}{2}\|\mathbf{g}(\boldsymbol{\theta}_t)\|^2 \Big| \boldsymbol{\theta}_t\right]$$
$$\mathbb{E}\left[f(\boldsymbol{\theta}_{t+1}) - f(\boldsymbol{\theta}_t)|\boldsymbol{\theta}_t\right] \leq -\eta\|\nabla f(\boldsymbol{\theta}_t)\|^2 + \frac{G^2 L\eta^2}{2}.$$

To infer the convergence rate, we want to see the rate in which the square of the gradient norm tends to zero since a magnitude of zero means the update rule has approaches a critical point, which will be the minimum, $f(\boldsymbol{\theta}^*)$, due to the convexity assumption. Following the outlined approach of [15], we usually want to know how many iterations we need to get to $\|\nabla f(\boldsymbol{\theta}_t)\|^2 \leq \epsilon$. Therefore we can rearrange the above to bound $\|\nabla f(\boldsymbol{\theta}_t)\|^2$ as

$$\eta\|\nabla f(\boldsymbol{\theta}_t)\|^2 \leq f(\boldsymbol{\theta}_t) - \mathbb{E}\left[f(\boldsymbol{\theta}_{t+1})|\boldsymbol{\theta}_t\right] + \frac{G^2 L\eta^2}{2}.$$

Introducing the summation over T steps of the squared norms while using the fact that the first two terms on the right make up a telescoping series, such that $\sum_{t=1}^{T} \mathbb{E}\left[f(\boldsymbol{\theta}_t) - f(\boldsymbol{\theta}_{t+1})|\boldsymbol{\theta}_t\right] = \mathbb{E}\left[f(\boldsymbol{\theta}_1) - f(\boldsymbol{\theta}_{T+1})|\boldsymbol{\theta}_1\right]$, and assuming $f(\boldsymbol{\theta}_{T+1}) \geq f(\boldsymbol{\theta}^*)$, gives

$$\sum_{t=1}^{T} \eta\|\nabla f(\boldsymbol{\theta}_t)\|^2 \leq f(\boldsymbol{\theta}_1) - f(\boldsymbol{\theta}^*) + \sum_{t=1}^{T} \frac{G^2 L\eta^2}{2}.$$

Finally, we know that $\sum_{t=1}^{T} \|\nabla f(\boldsymbol{\theta}_t)\|^2 \geq \min_{t\in\{1,\ldots,T\}} \left\{\|\nabla f(\boldsymbol{\theta}_t)\|^2\right\}$, giving

$$\min_{t\in\{1,\ldots,T\}} \left\{\|\nabla f(\boldsymbol{\theta}_t)\|^2\right\} \sum_{t=1}^{T} \eta \leq f(\boldsymbol{\theta}_1) - f(\boldsymbol{\theta}^*) + \sum_{t=1}^{T} \frac{G^2 L\eta^2}{2}$$
$$\min_{t\in\{1,\ldots,T\}} \left\{\|\nabla f(\boldsymbol{\theta}_t)\|^2\right\} \leq \frac{f(\boldsymbol{\theta}_1) - f(\boldsymbol{\theta}^*)}{\sum_{t=1}^{T} \eta} + \frac{G^2 L}{2} \frac{\sum_{t=1}^{T} \eta^2}{\sum_{t=1}^{T} \eta}.$$

Since we have implicitly assumed a constant step size, we get the following bound:

$$\min_{t\in\{1,\ldots,T\}} \left\{\|\nabla f(\boldsymbol{\theta}_t)\|^2\right\} \leq \frac{f(\boldsymbol{\theta}_1) - f(\boldsymbol{\theta}^*)}{T\eta} + \frac{G^2 L\eta}{2}.$$

If we want the error to be below ϵ, then

$$\frac{f(\boldsymbol{\theta}_1) - f(\boldsymbol{\theta}^*)}{T\eta} + \frac{G^2 L\eta}{2} \leq \epsilon,$$

Therefore error at T is $O(1/T) + O(\eta)$, meaning that we get sub-linear convergence up to some constant that depends on the learning rate. If the learning rate is set to be step dependent, then we can remove the constant and get different learning rates.

Appendix C. Detailed Results

C.1. Rosenbrock Function: Additional Results

As an initial illustration of the behavior of each forward-mode approach, we show how a single step looks from a randomly chosen starting point in Figure 1 for FGD, FoMoH-1D, and FoMoH-BP, where FoMoH-BP corresponds to FoMoH-1D with the direction being the exact gradient. We plot the expected (average) step across 10,000 samples for all approaches in the 2D Rosenbrock function. These steps are shown with solid lines. For each approach, we also plot the sampled steps by superimposing a scatter plot in the corresponding approach's color. All methods are compared to the Newton step, $(\nabla^2 f(\boldsymbol{\theta}))^{-1} \nabla f(\boldsymbol{\theta})$ shown in red. For FGD, we see that the expected descent direction is the same as the gradient at that point, hence the alignment with FoMoH-BP that directly calculates the gradient. This plot highlights the reliance on a well-chosen learning rate for FGD, whereas FoMoH-BP's step size is automatically normalized by the local curvature along the gradient. FoMoH-1D (green), on the overhand, has an expected descent direction that differs from the gradient and is governed by the distribution of samples that fall on the ellipse defined by the Hessian, $\nabla^2 f(\boldsymbol{\theta})$. For this point, the Newton step is the descent direction that falls on this ellipse and corresponds to the local minimum of the quadratic approximation. Another insight gained from this figure is that the variance of FGD's descent direction is less constrained than FoMoH's descent direction (blue), where the sample direction is controlled by the local Hessian. FoMoH-2D directly aligns with the Newton step.

In Figure 3, we now compare the performance of the competing optimization routines for the 2D Rosenbrock function. These results are shown for the same 10 randomly sampled starting locations for all approaches initialized with different random seeds. We highlight with the thicker line the median performing run. Both axes are on the log scale and show the significant advantage of FoMoH-KD, with $K = 2$. We also note the advantage (at least for this 2D example) of all FoMoH approaches that use second-order information. Additionally, the two forward-mode only approaches actually outperform the optimization routines that include backpropagation.

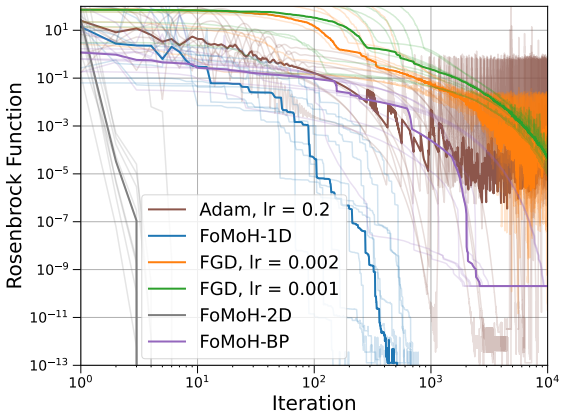

Figure 3: Comparison of the stochastic approaches in minimization of the Rosenbrock function. Average performance (median) is shown over 10 random initial conditions. FoMoH outperforms all first-order approaches, with FoMoH-2D converging in orders of magnitude faster than all methods.

10D Rosenbrock Function. We now focus on the performance of FoMoH-KD as we increase K from 2 to the input dimension of the function. Figure 2 shows this comparison for the 10D Rosenbrock function, where we use 10 random initializations for the different K. The median performance for each K is then shown, where we see a perfect ordering of performance that aligns with the dimension of the hyperplane. The best performing FoMoH-KD is for $K = D$, with the worst corresponding to the lowest dimension implemented, $K = 2$. Overall this figure highlights how FoMoH-KD trends towards Newton's method as K tends to D, where we actually see the median performances of FoMoH-10D and Newton's method aligned.

C.2. Multinomial Logistic Regression

Table 1 displays mean and standard deviation performance for each approach, where the forward-mode-only approaches are highlighted separately from the methods that include reverse-mode steps. Table 2 includes the final hyperparameter selection. We used [3] to perform a grid search with 100 iterations, where the batch size choice was between [128, 512, 1024, 2048] (see Table 2 for the final hyperparameter selection). For the learning rate scheduler, we reduced the learning rate at the epoch where the NLL starts to increase. For FoMoH-3D we multiplied the learning rate by 0.8, whereas for the other approaches we multiplied the learning rate by 0.1. Figures 4 and 5 display the training and validation curves, where we include a separate plot to focus on the forward-mode-only approaches.

Table 1: Multinomial Logistic regression results for MNIST. When comparing the forward-mode-only approaches in the upper section of the table, we see improvement in performance with increasing hyperplane dimension for FoMoH-KD. For this logistic regression example, FoMoH-3D and FoMoH-2D with learning rate schedulers, are competitive with FGD. However we see this result change with a larger dimensional problem in Table §3 for the CNN results. Both reverse-mode approaches in the lower section of the table have similar performance, and are included for reference.

APPROACH	TRAINING LOSS	VALIDATION LOSS	TRAINING ACCURACY	VALIDATION ACCURACY
FGD	0.2976 ± 0.0007	$\mathbf{0.2949 \pm 0.0017}$	0.9154 ± 0.0003	0.9163 ± 0.0019
FOMOH-1D	0.3223 ± 0.0013	0.3186 ± 0.0019	0.9073 ± 0.0011	0.9110 ± 0.0021
FOMOH-1D (LR-SCH.)	0.3192 ± 0.0012	0.3160 ± 0.0025	0.9085 ± 0.0011	0.9118 ± 0.0021
FOMOH-2D	0.3010 ± 0.0018	0.3015 ± 0.0031	0.9144 ± 0.0009	0.9149 ± 0.0015
FOMOH-2D (LR-SCH.)	0.2921 ± 0.0014	0.2951 ± 0.0027	0.9174 ± 0.0005	$\mathbf{0.9170 \pm 0.0010}$
FOMOH-3D	0.3449 ± 0.0017	0.3343 ± 0.0034	0.8999 ± 0.0008	0.9036 ± 0.0023
FOMOH-3D (LR-SCH.)	$\mathbf{0.2893 \pm 0.0017}$	0.3054 ± 0.0034	$\mathbf{0.9195 \pm 0.0007}$	0.9153 ± 0.0010
FOMOH-BP	0.2312 ± 0.0001	0.2679 ± 0.0003	0.9366 ± 0.0001	0.9267 ± 0.0002
BACKPROPAGATION	0.2265 ± 0.0000	0.2710 ± 0.0001	0.9381 ± 0.0001	0.9267 ± 0.0003

C.3. CNN

We now move from a model with $7{,}850$ parameters to a convolutional neural network (CNN) with $431{,}080$ parameters. As before we use the MNIST dataset and we now perform hyperparameter optimization using Bayesian optimization. Both Table 3 and Figure 6 highlight the advantage of FoMoH-KD over FGD. For the larger parameter space FGD requires more epochs to converge compared to all FoMoH-KD variants. The learning rate scheduler further improves FoMoH-KD by helping to avoid getting stuck in low performance regions. Here, we see the clear trend that

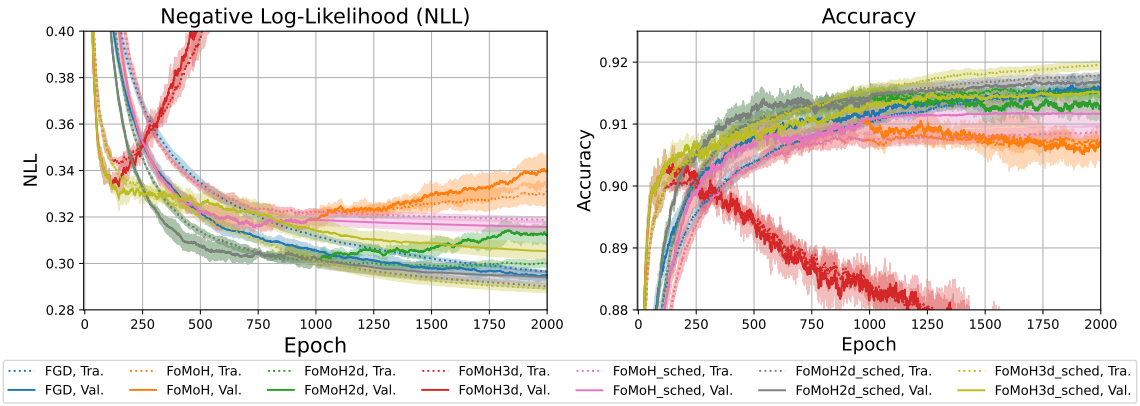

Figure 4: **Forward-Mode:** Training and validation curves for logistic regression model on the MNIST dataset. Average and standard deviation is shown for five random initializations.

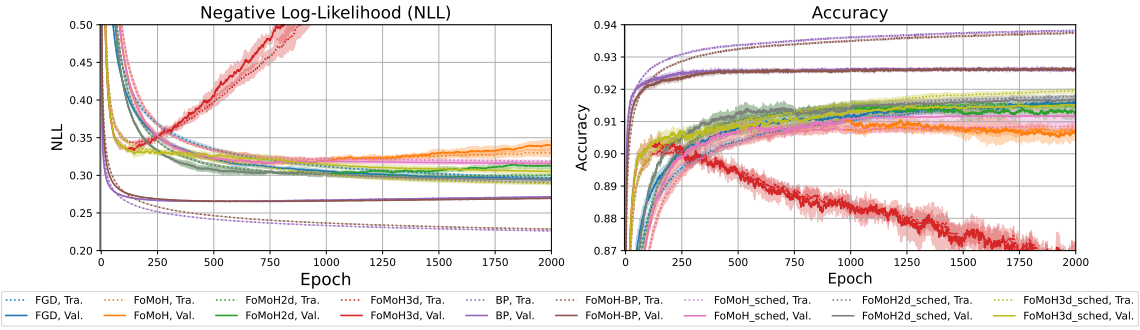

Figure 5: **Forward-Mode and Reverse-Mode:** Training and validation curves for logistic regression model on the MNIST dataset. Average and standard deviation is shown for five random initializations.

Table 2: Hyperparameter Optimization for Logistic Regression.

APPROACH	LEARNING RATE	LEARNING RATE BOUNDS	BATCH SIZE
FGD	0.00006497	[0.00001, 0.1]	128
FOMOH-1D	0.1362	[0.001, 1.0]	1024
FOMOH-1D (LR-SCH.)	0.1362	[0.001, 1.0]	1024
FOMOH-2D	0.04221	[0.001, 1.0]	512
FOMOH-2D (LR-SCH.)	0.04221	[0.001, 1.0]	512
FOMOH-3D	0.1	[0.001, 1.0]	512
FOMOH-3D (LR-SCH.)	0.1	[0.001, 1.0]	512
FOMOH-BP	0.04688	[0.01, 1.0]	2048
BACKPROPAGATION	0.03561	[0.01, 0.5]	2048

the best performing forward-mode-only approach comes from the largest K, which was $K = 3$ for this experiment. As expected, both optimizers FoMoH-BP and Backpropagation outperform the

forward-mode-only approaches. Overall, these results highlight that second-order information helps scale the performance of forward-mode optimization to larger dimensions.

Table 3: CNN results for MNIST. The forward-mode-only approaches in the upper section of the table show that FoMoH's performance improves with the dimension of K, especially when used with the learning rate scheduler. FoMoH-3D outperforms FoMoH-2D, FoMoH, and FGD. The reverse-mode approaches in the lower section outperform forward-mode, with BP slightly better than FoMoH-BP.

APPROACH	TRAINING LOSS	VALIDATION LOSS	TRAINING ACCURACY	VALIDATION ACCURACY
FGD	0.1211 ± 0.0097	0.1104 ± 0.0086	0.9641 ± 0.0038	0.9677 ± 0.0030
FOMOH-1D	0.1663 ± 0.0069	0.1571 ± 0.0092	0.9518 ± 0.0011	0.9550 ± 0.0002
FOMOH-1D (LR-SCH.)	0.1617 ± 0.0119	0.1575 ± 0.0158	0.9515 ± 0.0035	0.9539 ± 0.0034
FOMOH-2D	0.1015 ± 0.0041	0.1016 ± 0.0066	0.9691 ± 0.0011	0.9693 ± 0.0020
FOMOH-2D (LR-SCH.)	0.0900 ± 0.0070	0.0913 ± 0.0041	0.9731 ± 0.0022	0.9718 ± 0.0014
FOMOH-3D	0.1085 ± 0.0105	0.1073 ± 0.0133	0.9674 ± 0.0022	0.9686 ± 0.0028
FOMOH-3D (LR-SCH.)	$\mathbf{0.0809 \pm 0.0061}$	$\mathbf{0.0923 \pm 0.0132}$	$\mathbf{0.9759 \pm 0.0014}$	$\mathbf{0.9734 \pm 0.0022}$
FOMOH-BP	0.0093 ± 0.0016	0.0310 ± 0.0006	0.9981 ± 0.0005	0.9903 ± 0.0003
BACKPROPAGATION	0.0053 ± 0.0034	0.0329 ± 0.0032	0.9990 ± 0.0009	0.9909 ± 0.0004

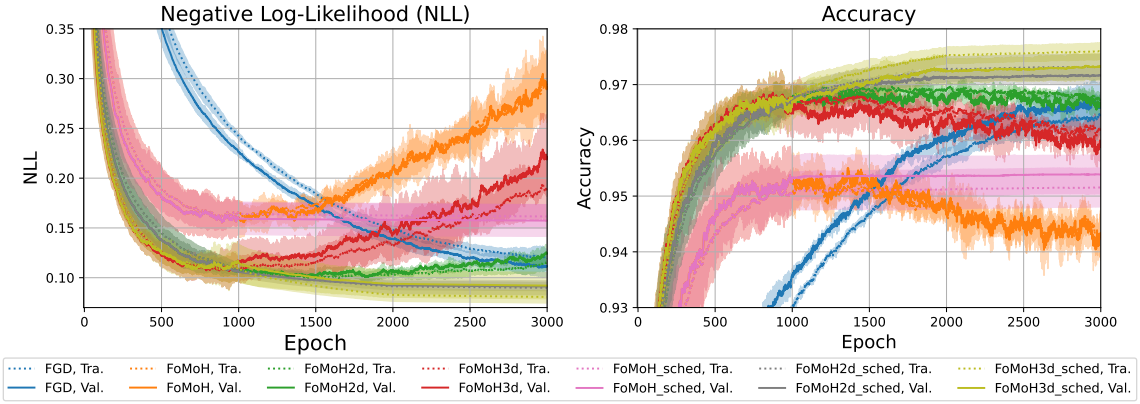

Figure 6: Forward-mode training and validation curves for the CNN on the MNIST dataset. Average and standard deviation is shown for three random initializations. Note how FGD (blue) is much slower to converge, with FoMoH-KD improving in performance with increasing K.

Table 4 includes the final hyperparameter selection for the experimental results. We used [3] to perform Bayesian optimization with 100 iterations, where the batch size choice was fixed to 2048. For FoMoH-3D, we used the same hyperparameters as for FoMoH-2D as this gave sufficient performance (and still outperformed the other forward-mode approaches). All learning rate schedulers reduced the learning rate by 10 every 1000 epochs.

Figure 7 includes the reverse-mode training and validation curves for Backpropagation and FoMoH-BP in addition to the curves shown in Figure 6.

Table 4: Hyperparameter Optimization for Logistic Regression.

APPROACH	LEARNING RATE	LEARNING RATE BOUNDS	BATCH SIZE
FGD	0.0001376	[0.00001, 0.1]	2048
FOMOH-1D	0.542	[0.001, 1.0]	2048
FOMOH-1D (LR-SCH.)	0.542	[0.001, 1.0]	2048
FOMOH-2D	0.3032	[0.001, 1.0]	2048
FOMOH-2D (LR-SCH.)	0.3032	[0.001, 1.0]	2048
FOMOH-3D	0.3032	[0.001, 1.0]	512
FOMOH-3D (LR-SCH.)	0.3032	[0.001, 1.0]	2048
FOMOH-BP	0.04688	[0.01, 1.0]	2048
BACKPROPAGATION	0.03561	[0.005, 0.2]	2048

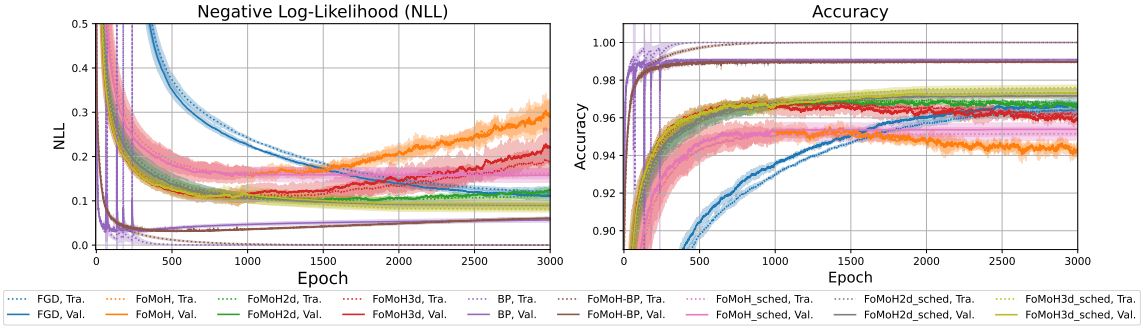

Figure 7: Training and validation curves for CNN on the MNIST dataset. Average and standard deviation is shown for three random initializations.

C.4. Computational Resources

All experiments are run on a NVIDIA RTX 6000 GPU. The main compute cost came from both the grid search and the Bayesian optimization that we ran over the six different optimization routines for the different experiments.