Efficient Multiple Objective Optimization for Fair Misinformation Detection

Eric Enouen[1*], Katja Mathesius[2*], Sean Wang[3*], Arielle Carr[4], Sihong Xie[4]
Ohio State University[1], Drake University[2], Cornell University[3], Lehigh University[4] * Equal contributions

***Abstract*—Multiple-objective optimization (MOO) aims to simultaneously optimize multiple conflicting objectives and has found important applications in machine learning, such as simultaneously minimizing classification and fairness losses. At an optimum, further optimizing one objective will necessarily increase at least another objective, and decision-makers need to comprehensively explore multiple optima to pin-point one final solution. We address the efficiency of exploring the Pareto front that contains all optima. First, stochastic multi-gradient descent (SMGD) takes time to converge to the Pareto front with large neural networks and datasets. Instead, we explore the Pareto front as a manifold from a few initial optima, based on a predictor-corrector method. Second, for each exploration step, the predictor iteratively solves a large-scale linear system that scales quadratically in the number of model parameters, and requires one backpropagation to evaluate a second-order Hessian-vector product per iteration of the solver. We propose a Gauss-Newton approximation that scales linearly, and that requires only first-order inner-product per iteration. Third, we explore different linear system solvers, including the MINRES and conjugate gradient methods for approximately solving the linear systems. The innovations make predictor-corrector efficient for large networks and datasets. Experiments on a fair misinformation detection task show that 1) the predictor-corrector method can find Pareto fronts better than or similar to SMGD with less time, and 2) the proposed first-order method does not harm the quality of the Pareto front identified by the second-order method, while further reducing running time.**

I. Introduction

Multi-objective optimization aims to find optimal solutions for multiple objective functions and has been an important tool for machine learning. For example, in multi-task learning, each learning task has an objective function to be optimized and tasks can be optimized jointly; in a recommendation system, content relevance and personalization are two important goals for the system to achieve simultaneously; and in fair machine learning, fairness and classification accuracy are two important objectives. Usually the multiple objectives are conflicting and it is impossible to find a solution that is optimal for all individual objective functions. Rather, trade-offs among the objectives are necessary and optimality in MOO can be characterized by the Pareto optimality: a Pareto optimum is a solution where improving any one objective function necessarily harms at least one other objective function, and jointly improving all objectives is impossible.

Previous work [3] aims to find a single Pareto optimum without controlling the trade-off among the objectives. This is undesirable since an objective function may not be sufficiently minimized, while the users cannot access and compare multiple trade-offs. To address this, the authors of [22], [12] proposed multi-gradient descent methods, which maintain a set of current best trade-offs and push the solutions towards the Pareto front. However, they still cannot control which trade-offs to reach during the optimization and the solutions can occupy only a small region of the front. In [11], [14], further constraints are added to the gradient-based optimization so that preferences over the trade-offs can be specified and the solutions are better spread across the front.

One common drawback of this prior work is the computational efficiency in comprehensively traversing the Pareto front for practitioners to pinpoint the desired trade-off. The above optimization algorithms start from arbitrary initial points that can be far away from the fronts, demanding many gradient descent steps to converge. Recovering the fronts requires running these algorithm multiple times starting from some set of initial random solutions. As each descent iteration can be costly with large datasets and neural networks, such methods are not feasible for comprehensively recovering the Pareto fronts (see, e.g., Figure 3).

The predictor-corrector method addresses this issue. Intuitively, the optimal solutions are assumed to form a low dimensional manifold, which can be approximated by a tangent plane at an optimal solution. Therefore, moving from one optimal solution to a neighboring optimal solution can be done by moving along the plane (via the predictor step) and pushing the approximated solution back to the Pareto front manifold (via the corrector step) to find the next optimal solution. Since two Pareto optima are closed on the manifold, it is expected that less computation is needed to explore the manifold locally.

However, in the predictor step, we still face two challenges centering around solving a large-scale linear system of the form $H\mathbf{v} = \mathbf{b}$. Here, H is an $n \times n$ symmetric matrix and $\mathbf{v} \in \mathbb{R}^n$ is a vector of exploration direction that moves the current Pareto optimum $\mathbf{x}$ to $\mathbf{x} + \mathbf{v}$, and $\mathbf{b} = \sum_{i=1}^{m} \beta_i \nabla f_i(\mathbf{x})$ is a linear combination of the gradients of individual objective functions f_i, $i = 1, \ldots, m$, for some coefficients $\beta_i > 0$. First, the system $H\mathbf{v} = \mathbf{b}$ requires an iterative solver, such as the Krylov methods conjugate gradient (CG) or MINRES [23], [19]. As the solver must solve a new system per predictor iteration, the associated cost is a substantial portion of the total cost of exploring a Pareto front. In particular, the matrix H is a linear combination of the Hessian matrices of the objective functions [16], [13]. Direct evaluation of the Hessian matrices can be costly or even infeasible for deep learning models. One can apply the Pearlmutter trick [18] and use

978-1-6654-8045-1/22/$31.00 ©2022 IEEE

a forward and backward propagation implemented by auto-differentiation to evaluate the matrix-vector product, $H\mathbf{v}$. This is still quite costly for large networks and datasets since one forward-backward propagation is needed per iteration of the solver, which is invoked per iteration of the predictor. Second, without knowing the properties of the system with a specific multi-objective optimization problem, currently no study about the efficiency of the solvers is available. Since we do not explicitly store the Hessian, H, the use of Krylov methods is a natural choice as such solvers require only a linear operator (or matrix-vector product) to construct the Krylov subspace. In the present study, we consider CG and MINRES since both can be employed when the coefficient matrix (e.g., H) is symmetric. CG, however, requires that the matrix be symmetric *positive definite* (PSD) and H is not guaranteed to be PSD.

We propose a different approach based on the predictor-corrector method to approximate the Pareto front. We address the efficiency of the Pearlmutter trick by avoiding the forward and backward propogation required to compute $H\mathbf{v}$ during each iteration of the linear system solver. We adopt the Gauss-Newton method to approximate the Hessian matrices, and the resulting approximation is low-rank, cheaper-to-compute, and PSD. The Gauss-Newton approximation uses the sum of outer products of gradient vectors of objective functions to approximate the second-order Hessian. As a result, only the gradients of the objective functions are needed and those can be computed as calculating the right-hand side $\mathbf{b}$ vector. No additional forward and backward propagations are needed for computing $H\mathbf{v}$. Since the outer products are PSD, either MINRES and CG methods can be used to solve the corresponding linear systems. As a by-product, we are able to benchmark the cost of MINRES and CG within the predictor-corrector method.

The paper is organized as follows. In Section II, we review multi-objective optimization (MOO). In Section III, we describe the predictor-corrector algorithm and the Gauss-Newton approximation for MOO. In Section IV, we formulate an MOO problem for fair misinformation detection and validate our claims through empirical experiments on three datasets for misinformation detection.

II. Preliminaries

In this section we review the basics of multi-objective optimization, predictor-corrector methods, and iterative methods for solving linear systems. The notation is in Table I.

A. MOO and Multi-gradient descent

We consider an MOO problem that has m objective functions $f_i(\mathbf{x})$, $i = 1, \ldots, m$, where $\mathbf{x} \in \mathbb{R}^n$ is the parameter of the functions. For example, in fair machine learning, the goal is to optimize a predictive model's parameter $\mathbf{x}$ so as to minimize classification loss (measured by f_1) while reducing the discrepancy (measured by f_2) between the treatment of different populations. We let $\mathbf{f} = [f_1, \ldots, f_m]^\top : \mathbb{R}^n \to \mathbb{R}^m$ be the vector of the m objective functions. We want to find an optimal $\mathbf{x}^*$ that minimizes all m objectives simultaneously.

TABLE I: Notations and definitions.

Notations	Definitions
f_i	The i-th objective function to minimize
$\mathbf{f}$	The vector of the m objectives
$\mathbf{x}$	Vector of parameters of $\mathbf{f}$
J	The Jacobian of $\mathbf{f}$ w.r.t. $\mathbf{x}$
$J^\top$	The transpose of a matrix, J
H_i	The Hessian matrix of f_i w.r.t. $\mathbf{x}$
H^i	The i^{th} power of H
$\mathbf{v}^{(i)}$	The approximate solution at the i^{th} iteration

However, since the objectives can be conflicting and no single solution $\mathbf{x}$ can attain all the minima of individual objectives, we must accept some trade-offs among the objectives. To characterize the optimality with multiple conflicting objectives, a Pareto optimum is a solution where simultaneously reducing all m objectives is impossible, and reducing one objective will necessarily increase at least another objective. We say that the solution $\mathbf{x}$ dominates another solution $\mathbf{x}'$ if $f_i(\mathbf{x}) \leq f_i(\mathbf{x}')$ for all $i = 1, \ldots, m$ and at least one strict inequality holds. A Pareto optimum is optimal in the sense that it is not dominated by any other solutions.

There can be multiple Pareto optima and the image of the set of Pareto optima under the mapping $\mathbf{f}$ in the space $\mathbb{R}^m$ is called the Pareto front. We aim to generate a Pareto front for $\mathbf{f}$ so that a user can select the optimal solution with the accepted trade-offs. For example, in fair machine learning, we want to find a predictive model with accuracy higher than a given threshold while minimizing unfairness. Without searching for a Pareto front, the user may not be able to find solution with the desired levels of accuracy and fairness.

To find a Pareto front, the multi-gradient descent algorithm [4], [3] starts from a random initial solution and calculates a descent direction that can jointly reduce all objective functions in each iteration. The iterations continue until a Pareto optimum is reached where such a descent direction is no longer possible. If the current solution $\mathbf{x}$ is not on the Pareto front, we can optimize the weights (λ_i, $i = 1, \ldots, m$) of the gradients of the objectives so that the weighted sum of the gradients is a descent direction for all objectives as

$$\max_{\lambda} -\frac{1}{2}\left\|\sum_{i=1}^{m}\lambda_i(\nabla f_i(\mathbf{x}))\right\|^2 \tag{1}$$

$$\text{s.t. } \sum_{i=1}^{m}\lambda_i = 1, \lambda_i \geq 0, i = 1, ..., m, \tag{2}$$

where $\nabla f_i(\mathbf{x})$ is the gradient of the i-th objective function at the current solution $\mathbf{x}$. This can be extended to find multiple Pareto solutions on the front [12]. Since a Pareto solution is found based on a random initial solution, the method can still take many steps to move to the Pareto front. Thus, to comprehensively recover a Pareto front, a large set of solutions needs to be maintained and optimized, leading to high cost.

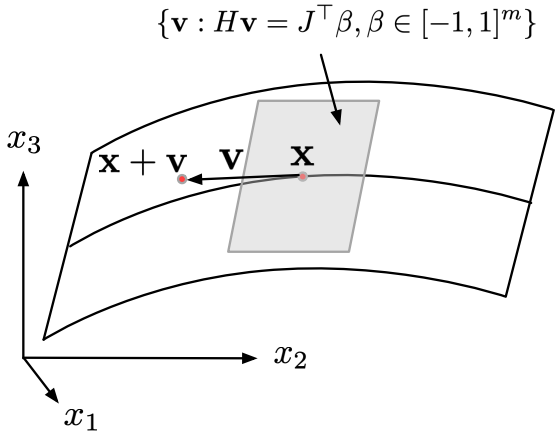


Fig. 1: Pareto optima constitute a 2-dimensional manifold in $\mathbb{R}^3$. The predictor uses the tangent plane to predict the direction $\mathbf{v}$ so that $\mathbf{x}+\mathbf{v}$ is close to the next Pareto optimum.

B. Predictor-corrector methods

We introduce the predictor-corrector method that can find the Pareto front more efficiently. Predictor-corrector methods were introduced to MOO in [16], [13] as a way to explore a Pareto front.A predictor-corrector method consists of two steps: a predictor step that approximately moves to a neighboring point of the current Pareto solution, and a corrector step that pushes the approximated neighboring point onto the Pareto front. Another round of prediction and correction can then be conducted to generate the next Pareto solution.

Predictor. This step is derived in [7]. For brevity, we present the main result below. Locally, at the current Pareto solution $\mathbf{x}$, the hyperplane in $\mathbb{R}^n$ that passes through $\mathbf{x}$ and is tangent to the manifold can be described by the linear system

$$H(\mathbf{x})\mathbf{v} = J^T\beta. \tag{3}$$

Here, $H(\mathbf{x}) = \sum_{i=1}^m \alpha_i H_i(\mathbf{x})$ is a linear combination of the Hessians of individual objectives, $\mathbf{v}$ represents the exploration direction so that $\mathbf{x}+\mathbf{v}$ is on the tangent hyperplane, $J \in \mathbb{R}^{m\times n}$ is the Jacobian matrix of the m objective functions in $\mathbf{f}$, and β is a weighting vector chosen from $[-1,1]^m$ to determine which objectives to increase/decrease by moving along $\mathbf{v}$. See Fig. 1 for a demonstration in $\mathbb{R}^3$.

Corrector. The corrector step can employ any multi-objective optimization method, such as the above-mentioned multi-gradient descent method. The idea is to descend towards the Pareto front starting from $\mathbf{x}+\mathbf{v}$ where $\mathbf{v}$ is the exploration direction generated by the predictor. This step is necessary since the predictor step approximates the manifold using a linear system and so $\mathbf{x}+\mathbf{v}$ can be off the manifold.

C. Krylov Subspace Methods

A Krylov subspace method iteratively computes an approximate solution $\mathbf{v}^{(j)}$ at iteration j when solving the linear system $H\mathbf{v} = \mathbf{b}$ by updating the initial solution $\mathbf{v}^{(0)}$ as $\mathbf{v}^{(j)} = \mathbf{v}^{(0)} + \mathbf{z}^{(j)}$, with $\mathbf{z}^{(j)} \in \mathcal{K}^j(H,\mathbf{u})$. Here,

$$\mathcal{K}^j(H,\mathbf{u}) = \text{span}\{\mathbf{u}, H\mathbf{u}, H^2\mathbf{u}, \ldots, H^{j-1}\mathbf{u}\} \tag{4}$$

is the Krylov space of dimension j, for H an $n\times n$ square matrix and $\mathbf{u}$ a length n column vector.[1] The update $\mathbf{z}^{(j)}$ comes from a projection onto the Krylov space, and its computation, as well as the choice of $\mathbf{u}$, is specific to the method. We note that $\mathcal{K}^j$ is never explicitly computed as in (4). For the methods employed in this paper, MINRES implicitly builds an orthogonal basis for the Krylov space and iterates such that the residual $\|\mathbf{r}^{(j)}\| = \|\mathbf{b} - H\mathbf{v}^{(j)}\|$ is minimized. CG does so such that the residual is orthogonal to $\mathcal{K}^j$. Other approaches, such as biconjugate gradients and error minimizing methods also exist (see e.g., [23]). More details on the Krylov methods specifically employed in this paper are given in Section III-A, and we refer the reader to [23], [19] for further details on Krylov methods in general.

III. Proposed method

The key bottleneck of the predictor-corrector method is to solve the large-scale linear system in Eq. (3). There exist critical challenges to this: 1) The matrix H is too large and dense to be computed and stored explicitly. 2) Inverting H is very expensive (i.e., $O(n^3)$ time complexity) rendering iterative methods (Section III-A) necessary. 3) Though an iterative method using the Pearlmutter trick can avoid the direct evaluation and inversion of H, it still requires one backpropagation for every iteration of the iterative methods to compute $H\mathbf{v}$. With large datasets, especially those without the I.I.D. assumption to facilitate stochastic gradient estimation, estimation of the matrix-vector product requires going through the datasets once for each backpropagation. Lastly, 4) the matrix H is not necessarily PSD, and so the use of CG is not guaranteed.

A. CG and MINRES

In order to solve for $\mathbf{v}$ in Eq. (3), we want to avoid explicitly taking the inverse of the Hessian (i.e., we avoid *directly* solving the system) since this is generally far too expensive for our purposes and particularly as the number of parameters becomes very large. Instead, we employ computationally-less expensive iterative methods for approximately solving linear systems, and specifically the Krylov methods MINRES and CG, which can be implemented in a matrix-free fashion. In other words, we do not need to explicitly store the Hessian matrices in memory; rather, we simply need to define the matrix-vector product, or linear operator. The residual is guaranteed to monotonically decrease when employing MINRES, enabling early termination of the method. A major contribution of previous work was introducing the use of the iterative solver, MINRES [13]. In the present study, we aim to expand the user's choice to include CG and we provide a novel comparative analysis of these two iterative solvers when performing multiple-objective optimization.

CG and MINRES are iterative solvers for symmetric linear equations $H\mathbf{v} = \mathbf{b}$. CG [6] was developed as a more computationally efficient variant of the gradient descent method

[1]Notationally, we let H be our coefficient matrix throughout, but note that this need not be a Hessian matrix to use these Krylov methods.

and requires that H also be positive definite. Both methods operate by finding the gradient at the current solution (in our context, the $\mathbf{v}$ vector) and then moving in the H-orthogonal direction. That is, at iteration k, given a point $\mathbf{v}^{(k)}$ and a direction $\mathbf{p}^{(k)}$, CG performs a line search to find the value α and updates the solution as $\mathbf{v}^{(k+1)} = \mathbf{v}^{(k)} + \alpha\mathbf{p}^{(k)}$. Then, a new direction that is conjugate to $\mathbf{p}^{(k)}$ is computed such that at each iteration the residual is perpendicular to the Krylov space, that is, $\mathbf{r}^{(k)} \perp \mathcal{K}^k$. Theoretically, this method is guaranteed to converge in at most n iterations, but it is well-known that CG often reaches an acceptable tolerance in far fewer iterations. For ease of reference, we provide the CG method following that in [19] in Algorithm 1.[2]

The MINRES method solves the system $H\mathbf{v} = \mathbf{b}$ by choosing the update to the approximate solution $\mathbf{v}^{(k)}$ such that $\|\mathbf{r}^{(k)}\| = \|\mathbf{b} - H\mathbf{v}^{(k)}\|$ is minimized. One of the key features of this method is due to the symmetry of H: MINRES saves significant memory costs, requiring the storage of only the two previously computed basis vectors from the Krylov space. This is performed via the Lanczos algorithm using what is referred to as a three-term recurrence; we omit the details here but refer the reader to Section 6.6 in [19]. Since the practical implementation of MINRES can become quite complicated, we provide the algorithm for the algebraically equivalent conjugate residual (CR) method [19] for brevity. The major computational steps (and costs) of MINRES for our purposes can be easily highlighted in the CR algorithm. In both Algorithms 1 and 2, we let tol denote the user-defined convergence tolerance, $maxIter$ represent the max-allowable iterations, and $\mathbf{v}^{(0)}$ be the initial guess.

Algorithm 1 Conjugate Gradient Method for $H\mathbf{v} = \mathbf{b}$

1: $\mathbf{r}^{(0)} = \mathbf{b} - H\mathbf{v}^{(0)}$.
2: $\mathbf{p}^{(0)} = \mathbf{r}^{(0)}$.
3: $i = 0$.
4: **while** $i < maxIter$ and $\|\mathbf{r}^{(i)}\| > tol$ **do**
5: $\quad \alpha_i = \frac{\|\mathbf{r}^{(i)}\|^2}{(\mathbf{p}^{(i)})^\top H\mathbf{p}^{(i)}}$.
6: $\quad \mathbf{x}^{(i+1)} = \mathbf{x}^{(i)} + \alpha_i\mathbf{p}^{(i)}$.
7: $\quad \mathbf{r}^{(i+1)} = \mathbf{r}^{(i)} - \alpha_i H\mathbf{p}^{(i)}$.
8: $\quad \beta_i = \frac{\|\mathbf{r}^{(i+1)}\|^2}{\|\mathbf{r}^{(i)}\|^2}$.
9: $\quad \mathbf{p}^{(i+1)} = \mathbf{r}^{(i+1)} + \beta_i\mathbf{p}^{(i)}$.
10: $\quad i = i + 1$.
11: **end while**

CG is used for PSD matrices, while MINRES is reserved for symmetric indefinite matrices. While we provide a comparative analysis of these methods in the present study, the development of a specific recipe for when to choose one method over the other for these applications is part of ongoing work. Our results demonstrate flexibility in the choice of solver without sacrificing computational time.

We note that simple optimizations can be immediately taken advantage of in both algorithms. Since both require the initial residual, $\mathbf{r}^{(0)}$, using an initial guess of $\mathbf{v}^{(0)} = [0\ 0\ \cdots 0]^T$ allows us to avoid the matrix-vector product in line 1 of each algorithm and we simply let $\mathbf{r}^{(0)} = \mathbf{b}$. Further, and as we will demonstrate in our experiments, basing termination of CG and MINRES on maximum iterations alone is sufficient for generating an accurate Pareto front, allowing us to remove the test for convergence in future implementations (i.e., the computation of $\|\mathbf{r}^{(i)}\|$ at every iteration). While this is not a significant cost alone, when solving a large number of linear systems, the accumulated cost may no longer be negligible. Ongoing work focuses on continued improvements to our implementations to take advantage of these, and other, opportunities for speed-up in the convergence of these iterative methods.

[2]Note that in the iterative method literature, H is often used to denote an upper Hessenberg matrix, especially for methods like GMRES [20]. For consistency, we use H to denote a general coefficient for our application.

Algorithm 2 Conjugate Residual Method for $H\mathbf{v} = \mathbf{b}$

1: $\mathbf{r}^{(0)} = \mathbf{b} - H\mathbf{v}^{(0)}$.
2: $\mathbf{p}^{(0)} = \mathbf{r}^{(0)}$.
3: $i = 0$.
4: **while** $i < maxIter$ and $\|\mathbf{r}^{(i)}\| > tol$ **do**
5: $\quad \alpha_i = \frac{(H\mathbf{r}^{(i)})^\top\mathbf{r}^{(i)}}{\|H\mathbf{p}^{(i)}\|^2}$.
6: $\quad \mathbf{x}^{(i+1)} = \mathbf{x}^{(i)} + \alpha_i\mathbf{p}^{(i)}$.
7: $\quad \mathbf{r}^{(i+1)} = \mathbf{r}^{(i)} - \alpha_i H\mathbf{p}^{(i)}$.
8: $\quad \beta_i = \frac{(H\mathbf{r}^{(i+1)})^\top\mathbf{r}^{(i+1)}}{(H\mathbf{r}^{(i)})^\top\mathbf{r}^{(i)}}$.
9: $\quad \mathbf{p}^{(i+1)} = \mathbf{r}^{(i+1)} + \beta_i\mathbf{p}^{(i)}$.
10: $\quad i = i + 1$.
11: **end while**

We note that in our results (Section IV), we provide data from experiments using CG with the Hessian matrix *for comparison purposes only*. Because we cannot guarantee that the Hessian matrix is PSD, we would naturally choose MINRES as our iterative solver. As we later show, we still generate a similar Pareto front (in terms of quality) using CG with the Hessian, suggesting that some of these matrices may be (close to) PSD or that CG does not suffer as dramatically as the theory suggests for (slightly) indefinite matrices. Though, we do highlight that, in many cases, CG with the Hessian results in the largest overall runtime.

B. Gauss-Newton Approximation

We propose computing an approximation of H to avoid evaluating the Hessian matrix (and thus the backpropagation) per iteration of the solvers in Section III-A required to evaluate a single Hessian-vector product. The innovative techniques will rely on the Gauss-Newton approximation [21], which uses a low-rank positive semi-definite matrix to approximate H for optimizing a single scalar function $f(\mathbf{x})$. Here, we are minimizing a classification loss $\ell = \ell(h(\mathbf{x}))$, where h is the machine learning model that outputs a real value for regression or a class probability distribution for classification, and ℓ is a convex loss function, such as square error for linear regression or negative log-likelihood for classification that takes $h(\mathbf{x})$ as input. The Gauss-Newton approximation is based on

$$\begin{aligned} \ell' &= \ell'(h(\mathbf{x}))\nabla_{\mathbf{x}}h(\mathbf{x}), \\ \ell'' &= \nabla_{\mathbf{x}}h(\mathbf{x})\ell''(h(\mathbf{x}))(\nabla_{\mathbf{x}}h(\mathbf{x}))^\top + \ell'(h(\mathbf{x}))\nabla^2_{\mathbf{x}}h(\mathbf{x}), \end{aligned}$$

where the second line follows from the chain rule of derivatives. The Gauss-Newton approximation replaces the Hessian, H, with $\nabla_{\mathbf{x}} h(\mathbf{x}) \ell''(h(\mathbf{x}))(\nabla_{\mathbf{x}} h(\mathbf{x}))^\top$. This is reasonable when $\ell'(h(\mathbf{x}))$ is near 0 or $h(\mathbf{x})$ is near linear around $\mathbf{x}$ [21].

Inspired by the above analysis, we exploit the Gauss-Newton approximation to speed up solving Eq. (3) within the predictor step. However, the objective functions that we adopt are loss functions that may not be convex with respect to the output of the model, $h(\mathbf{x})$, and the term $\ell''(h(\mathbf{x}))$ may not be positive to ensure that the Gauss-Newton approximation is positive definite. For example, there are many fairness loss functions that are non-convex [24]. Another issue with the second-order term is when the loss is piece-wise linear so that the term (and the Gauss-Newton approximation) becomes zero in some regions, leading to a singular linear system in Eq. (3).

We propose the following Gauss-Newton approximation for the i-th objective function f_i:

$$\text{GN}_i = \nabla_{\mathbf{x}} f_i(\mathbf{x}) \nabla_{\mathbf{x}} f_i(\mathbf{x})^\top = J_i^\top J_i, \tag{5}$$

where the Jacobian J_i is the column vector of partial derivative of $f_i(\mathbf{x})$ with respect to $\mathbf{x}$. Then, the matrix $H(\mathbf{x}) = \sum_{i=1}^n \alpha_i H_i(\mathbf{x})$ in Eq. (3) is replaced with $\text{GN}(\mathbf{x}) = \sum_{i=1}^n \alpha_i \text{GN}_i(\mathbf{x})$. This approximation is guaranteed to be positive definite so long as not all gradients are zero, facilitating the use of CG.

Regarding the computational complexity, we are no longer required to use backpropagation that goes through the training data once to evaluate $H(\mathbf{x})\mathbf{v}$ per iteration of the solvers (as even when $\mathbf{x}$ remains the same, $\mathbf{v}$ changes during a predictor step). Instead, we can go through training data just once to evaluate and cache the Jacobian matrix, which scales only linearly in n (the dimension of $\mathbf{x}$) and in m (the number of objectives). Per iteration of the linear system solver, only m vector inner products (evaluating $J_i\mathbf{v}$) and m scalar-vector product (evaluating $J_i^\top(J_i\mathbf{v})$) are needed for fixed J_i, $i = 1, \ldots, m$. The Jacobian can be updated at the next iteration of the predictor-corrector method, where we reach the next Pareto optimum.

C. Predictor-Corrector using Gauss-Newton approximation

Algorithms 3 and 4 describe how we generate a Pareto front. After training an initial Pareto optimal network with parameter $\mathbf{x}$, we use Algorithm 3 to explore the Pareto front in a breadth-first style. Algorithm 4 describes the computation of the Hessian-vector product using the Gauss-Newton approximation. Algorithm 4 is used at each iteration of MINRES or CG in the predictor step of the predictor-corrector algorithm. We take advantage of the fact that we compute a weighted gradient of the objective functions, so J and J^T can be represented as vectors. We can then use the Gauss-Newton approximation to approximate the product $H(\mathbf{x})\mathbf{v}$ using the fixed J matrix, followed by m inner products and scalar-vector product in an iteration of MINRES or CG. Note that in our method, we use $K = 1, \alpha = 0.1$, and predetermined values for β. However, one could also store and reuse the computed weighted gradient and instead randomly sample β to generate more than one child network from a single parent.

TABLE II: Dataset and model sizes

Datasets	$\lvert\mathcal{V}^P\rvert$	$\lvert\mathcal{V}^R\rvert$	$\lvert\mathcal{V}^U\rvert$	Model size
YelpChi	201	67395	38063	1234
YelpNYC	923	358911	160220	1234
YelpZip	5044	608598	260277	1234

Algorithm 3 PC-GN-CG/MINRES

1: $\mathbf{x}$ = An initial Pareto optimal network.
2: N = Total number of networks to generate in each direction.
3: K = Number of children to generate per network.
4: β: Directions for Pareto front exploration.
5: $count = 0$.
6: Initialize queue q and list $output$.
7: Add $\mathbf{x}$ to q and $output$.
8: **while** $count < N$ **do**
9: Pop a network $parent$ from q.
10: $numChildren = 0$.
11: **while** $numChildren < K$ **do**
12: Evaluate J.
13: Iteratively solve $H(\mathbf{x})\mathbf{v} = J^T\beta$ ($H(\mathbf{x})\mathbf{v}$ using Alg (4).
14: Predictor: $\mathbf{x} = \mathbf{x} + \alpha\mathbf{v}$.
15: Corrector: correct $\mathbf{x}$ with one step of SMGD.
16: $child = x$.
17: optimize $child$ using a single training epoch .
18: add $child$ to q and $output$.
19: $count = count + 1$.
20: $numChildren = numChildren + 1$.
21: **end while**
22: **end while**
23: Repeat lines 8-21 for $\beta = (0, 1)^T$.
24: Remove dominated points from output.

Algorithm 4 HVP computation using GN

1: Input: $\mathbf{v}$, a length n vector.
2: Use automatic differentiation to compute the gradients of the objective functions.
3: J = weighted sum of gradients .
4: κ = inner product $\langle J, \mathbf{v}\rangle$
5: **return** $H(\mathbf{x})\mathbf{v} \approx \kappa J$.

IV. Experiments

A. MOO tasks and datasets

We use multiple MOO tasks and datasets to demonstrate the efficiency and effectiveness of the proposed GN-PC-MOO method. Table II shows the sizes of the datasets and the number of parameters in the corresponding neural networks.

Fair fake review detection with GNN. Reviews on e-commerce, such as Amazon and Yelp, are important to customers and business owners, and there are many misleading fake reviews that need to be detected to ensure the trustworthiness of the reviews. The review data can be represented as a review-graph defined as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_1, \ldots, v_N\}$ denotes the set of nodes and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ represents the set of undirected edges. There are three types of nodes in $\mathcal{G}$: user,

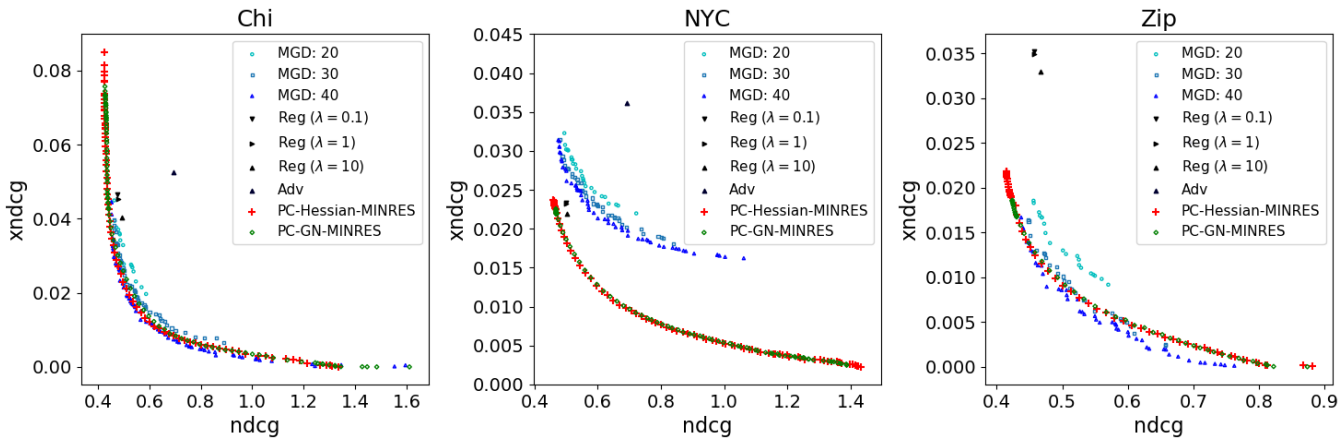


Fig. 2: Pareto fronts found by various methods. The two axes represent f_1 and f_2 objective function values on the training set. From left to right: comparison on YelpChi, YelpNYC, and YelpZip datasets. Within each figure: MGD is the SMGD method running for 20, 30, and 40 iterations; Reg (λ) means the fairness regularization method with a specific regularization parameter λ; Adv refers to the adversarial training method for fair machine learning; Predictor-Corrector uses the Hessian matrix with the CG solver, while the proposed method PC-GN-MINRES uses the Gauss-Newton approximation with the MINRES solver.

review, and product, respectively, and each node can be of only one of the three types. We denote the subsets of nodes of the three types as $\mathcal{V}^U, \mathcal{V}^R, \mathcal{V}^P \subset \mathcal{V}$, respectively. Each node $v_i \in \mathcal{V}$ has a feature vector $\mathbf{x}_i$, where the subscript is the node index. The neighbor of the node v_i is denoted by $\mathcal{N}(i) = \{v_j \in \mathcal{V} | e_{i,j} \in \mathcal{E}\}$.

GNN [10] is the state-of-the-art method for node prediction. For each node, a GNN model summarizes a node's neighborhood via message passing to predict if a review node is fake (positive) or genuine (negative). We let $\hat{y}_i = h(v_i; \mathbf{x})$ be the predicted probability of node v_i being fake, and train the model by minimizing the first objective, the cross-entropy loss on the set of labeled training reviews $\mathcal{V}^{\text{Tr}}$:

$$f_1(\mathbf{x}) = -\frac{1}{l} \sum_{i=1}^{l} \left[y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i) \right], \tag{6}$$

where $y_i \in \{0, 1\}$ is the label of a labeled review node, $v_i \in \mathcal{V}^{\text{Tr}} \cap \mathcal{V}^{\text{R}}$, and l is the total number of such labeled review nodes. We use the normalized discounted cumulative gain (NDCG) loss to measure the overall detection accuracy across all reviews:

$$\frac{1}{Z} \sum_{i=1}^{u} \mathbb{1}[y_i = 1] \frac{1}{\log(r_i + 1)}, \tag{7}$$

where r_i is the ranking position of the i-th unlabeled review node among all u unlabeled review nodes and Z is the maximal possible NDCG score as a normalization factor.

We also aim to reduce unfair treatment in the detection as the second objective. In particular, there are two groups of nodes, the favored group, indicated by $A = 0$ and consisting of reviewers, each of which posts more reviews. Those remaining reviewers posting fewer reviews are in the protected group, indicated by $A = 1$. On the training data, the protected group are labeled as spammers more often, biasing the trained GNN model to have a higher false positive rate over the protected group and leading to unfair detection. To measure the discrepancy in the detection accuracies over the two groups, we measure the NDCG on the two groups separately and take the absolute difference between the two NDCG scores as the second objective function f_2. Overall, we aim to optimize the GNN model $h(\mathbf{x})$ to find Pareto fronts of classification loss versus detection discrepancy tradeoffs.

B. Baselines and settings

We have two regular fair machine learning baselines that can find a solution that seeks to minimize both objectives. Fairness regularization [25], [26], [9] minimizes $f_1(\mathbf{x}) + \lambda f_2(\mathbf{x})$, where $\lambda f_2(\mathbf{x})$ represents the regularization and $\lambda > 0$ is the regularization strength. Adversarial training [2] trains an adversary that classifies data into two groups and our goal is to minimize f_1 while maximizing the adversary's classification error. We also include the following MOO methods as baselines.

- SMGD: the baseline proposed in [1]. It starts from several random initial solutions and then steps towards a Pareto front. It can be more time-consuming than the predictor-corrector methods as it can take many iterations to reach the front despite that it uses only first-order derivatives.
- PC-Hessian-CG: use the Pearlmutter trick [18] to evaluate the Hessian-vector product and use CG to solve the linear system for finding an exploration step in the predictor. If the Hessian matrices are PSD, then CG can be used. Though there is no guarantee that the Hessian matrices will be PSD, we include this baseline for a comprehensive comparison. The corrector uses multiple SMGD steps.
- PC-Hessian-MINRES: same as PC-Hessian-CG, except that the CG method is replaced with a MINRES solver.

We compare these baselines to two variants of the proposed method, PC-GN-CG and PC-GN-MINRES, that use Gauss-Newton approximation of the Hessian matrices. It is expected

that the Gauss-Newton approximation will reduce the running time of methods that rely on Hessian-vector products. We use different solvers to study whether the properties of the linear systems can influence the number of iterations, the running time, and the Pareto front quality.

Hyperparameter setting. We vary the λ in $f_1(\mathbf{x}) + \lambda f_2(\mathbf{x})$ for the fairness regularization method. For the PC methods, we generate one initial solution using 75 optimization steps of SMGD, followed by 100 predictor-corrector steps in directions $\beta = [-1, 1]^\top$ and $\beta = [1, -1]^\top$. For the predictor we used a step size of 0.1 and 50 max iterations for the solvers, following the ablation studies explored in [13]. For the corrector we used a step size of 0.01. We ran SMGD [12] for an increasing number of iterations (20, 30, and 40 epochs). The best step size was found to be 0.005 for the descent step.

C. Pareto front Quality

The closer a Pareto front is to the minimal values of individual objectives, the better. From the results provided in Figure 2, we make the following observations.

- On the YelpChi dataset, the two PC methods (PC-Hessian-MINRES and PC-GN-MINRES) are comparable in the quality of fronts produced by SMGD running for 40 epochs (which requires notably more time, as we discuss in Section IV-D). On the YelpNYC dataset, the front produced by the predictor-corrector methods completely dominates the fronts produced by the SMGD methods, even with 40 epochs. On the YelpZip dataset, the fronts produced by the PC methods are slightly dominated by SMGD that runs for 40 epochs. However, SMGD has a much longer running time.
- The spread of the fronts generated by SMGD with 20 epochs is much farther away from the origin than those from the predictor-corrector methods, which require a similar amount of time as the SMGD method. Overall, our method is a more efficient Pareto front generator and is able to find quality solutions at a faster rate than the multi-gradient descent algorithm.
- Both the regularization and adversarial training methods are dominated by SMGD and the PC methods. Furthermore, each baseline can generate just one solution and thus does not offer users a choice in the desired final solution.

D. Algorithm Speed

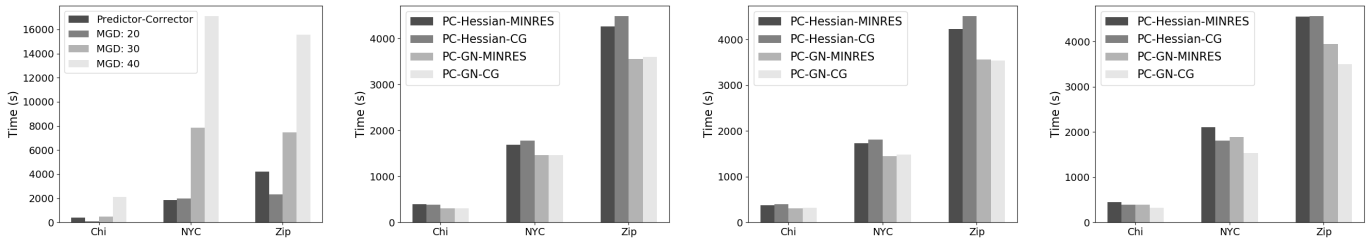
We demonstrate how the proposed Gauss-Newton approximation reduces the running time of the PC methods. We control the quality of the generated fronts and measure the time needed for each method to reach the fronts within an a certain proximity. Based on Figure 2, we regard the fronts found by SMGD as the target front and measure how long it takes for other methods to reach it. As shown in Figure 3a, the method PC-Hessian-CG has substantial speed ups over the multi-gradient descent method with over a 5-fold of speedup on YelpChi, a 9-fold of speedup on YelpNYC, and over a 3-fold speedup on YelpZip.

We further compare the running time of PC-GN-CG and PC-GN-MINRES that use Gauss-Newton approximation with PC-Hessian-CG and PC-Hessian-MINRES that use the Hessian matrices. From Figures 3b-3d with varying $maxIter$ for CG or MINRES to solve the linear system Eq. (3), we can see the the running time is further reduced from that of PC-Hessian-CG. This indicates that the Gauss-Newton approximation can reach the same Pareto fronts significantly faster. On YelpZip, the largest dataset, CG runs slightly faster than MINRES when the Gauss-Newton approximation is used. This is due to the fact that it requires fewer backpropagations. On the two smaller datasets, YelpChi and YelpNYC, CG can be slower than MINRES when the Hessian is used, since the reduction in Jacobian evaluations is less significant compared to the potentially slower convergence of CG when the Hessians are not guaranteed to be PSD. Overall, Gauss-Newton requires less running time than Hessian and running CG on the Gauss-Newton approximation is recommended.

E. Ablation and sensitivity studies

There are several components and hyperparameters of the proposed methods, and we study their influence on performance (i.e., quality of the Pareto fronts and running time).

- **Comparing different solvers.** In Figure 3, we show the total running time when using PC-Hessian-MINRES, PC-Hessian-CG, PC-GN-MINRES, and PC-GN-CG for varying maximum iterations, and in Figure 4 we show the quality of the Pareto fronts with maximum iterations set to 10. We observe that, in general, the Gauss-Newton approximation results in an overall faster runtime, but that there is no significant difference when using CG or MINRES. We also see that the running time does not change meaningfully when we reduce the number of maximum iterations for CG and MINRES (for all tests). This indicates that (1) the Gauss-Newton approximation is quite accurate while significantly reducing running time of the solvers, and (2) that the user can safely choose either CG or MINRES depending on preference (or availability) without affecting these same metrics.
- **The quality of the Pareto fronts with different maximum iterations for the solvers.** We compare the quality of the Pareto fronts found by our methods with a varied number of steps of the solvers that find the exploration direction $\mathbf{v}$ as in Eq. (3). Each row in Figures 5 and 6 show that for a fixed combination of linear system and solver, and for a varying number of the maximum iterations for CG and MINRES on the three datasets, we generate more or less the same Pareto fronts. This allows us to set a very modest number of maximum iterations for the iterative solver without affecting the quality of the Pareto fronts.
- **The effect of the corrector step.** We investigate whether the corrector is necessary after each predictor step, or if the predictor's approximation of the tangent direction is good enough and the corrector is unnecessary. We run PC-Hessian-CG with and without the corrector. The results can be seen in Figure 7. Without the corrector step, the



(a) Running time of PC-Hessian-CG and SMGD

(b) Running time of PC methods with 10 iterations for the solvers.

(c) Running time of PC methods with 25 iterations for the solvers.

(d) Running time of PC methods with 50 iterations for the solvers.

Fig. 3: (a): Running time of Predictor-Corrector (PC) and SMGD on the three datasets for fake review detection. (b)-(d): Running time of Predictor-Corrector (PC) methods with Hessian and Gauss-Newton approximation and CG/MINRES solvers, with $maxIter = 10$, 25, and 50.

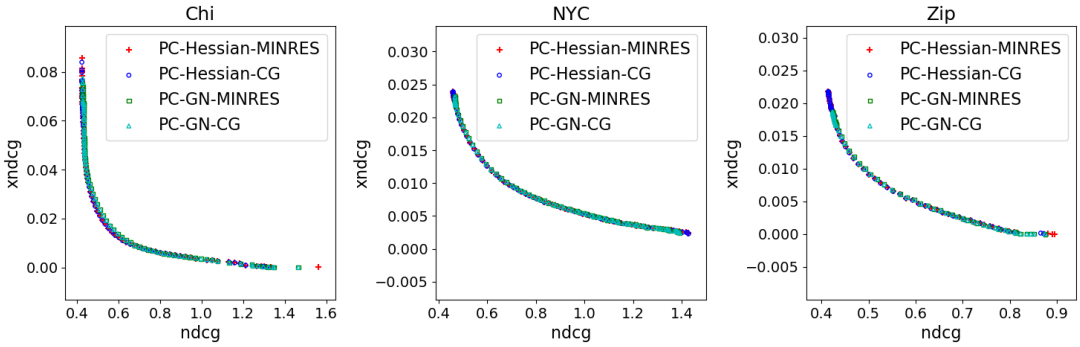
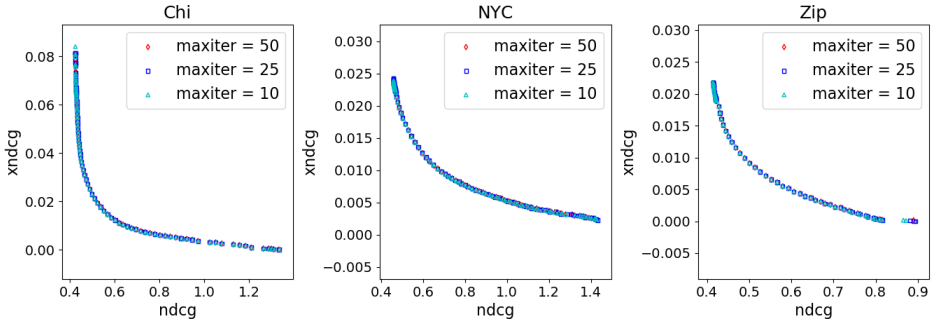
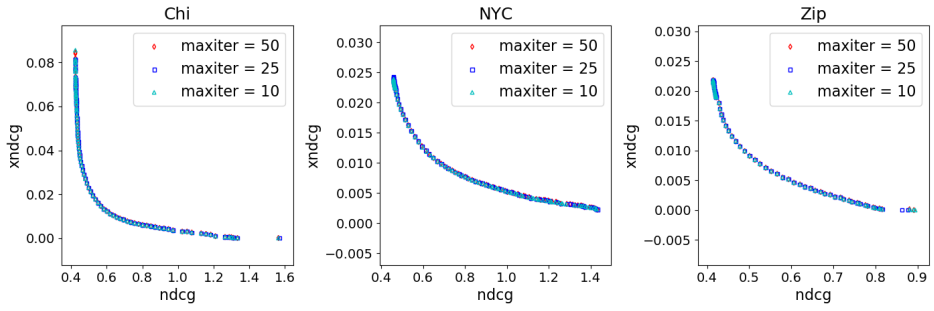


Fig. 4: Comparison of the quality of the Pareto front when using Hessian and Gauss-Newton matrices when solving the linear system in Eq. (3) using MINRES and CG.

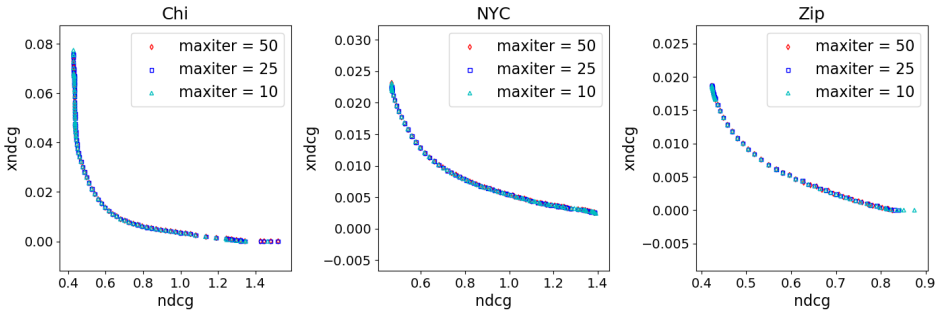


(a) PC-Hessian-CG

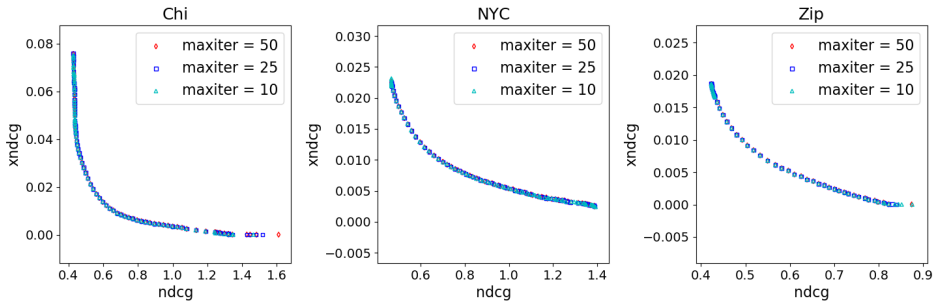


(b) PC-Hessian-MINRES

Fig. 5: Pareto front quality when varying the number of maximum iterations set for the linear solvers with Hessian matrices.



(a) PC-GN-CG



(b) PC-GN-MINRES

Fig. 6: Pareto front quality when varying the number of maximum iterations set for the linear solvers with Gauss-Newton approximations.

algorithm produces fronts that are slightly dominated by those generated with a corrector. However, the corrector also seems to shrink the Pareto fronts (shown in red) towards the bottom left corner. More Pareto optima can be produced, extending the fronts produced by the PC-Hessian-CG. This indicates that the predictor can indeed explore the manifold of Pareto optima to find new optimal solutions. In summary, the addition of a corrector step seems to trade off some spread of the produced solutions for more reduction in the objective function values where the objectives have the most competition. Therefore, we choose to utilize the corrector in all of the experiments.

V. Related Work

Multi-objective optimization has been studied extensively in numerical optimization [4], [17], [8], [3], [16], [12] and been applied to machine learning problems, such as multi-task learning and fair machine learning [22], [11], [14], [13]. We focus on the predictor-corrector method for its efficiency. Different from [13], [16] that directly use CG and the Pearlmutter trick to solve $H\mathbf{v} = \mathbf{b}$, we improve efficiency by exploiting low-rank approximation of the Hessian matrix H and the more advanced iterative method MINRES.

Solving a linear system for a descent direction $\mathbf{v}$ to optimize a machine learning model is commonly found in second-order numerical optimization. Second-order methods can exploit the local curvature information of an objective function, $\mathbf{f}$, to properly scale the gradient vector for a more robust descent direction. First-order methods need to fine-tune the step size in the direction of the negative gradient and this can be quite time-consuming in practice. The Newton method is a second-order method that solves the linear system $H\mathbf{v} = \mathbf{b}$, using iterative methods such as conjugate gradient. With many parameters to update, as is typical for neural networks, the Hessian matrix cannot be computed and stored explicitly, and the Pearlmutter trick is proposed to rely on backpropagation for fast Hessian-vector product evaluation to solve for $\mathbf{v}$. Another issue with using H is that it is not guaranteed to be positive definite and the CG method can have difficulty in convergence.

Gauss-Newton and Fisher information matrices are low-rank PSD matrices approximating the Hessian H. These approximating matrices are not only PSD, but also make the matrix-vector with $\mathbf{v}$ much cheaper to compute without requiring backpropagation for each iteration of CG that evaluates $H\mathbf{v}$. Rather, the approximating matrices are in the form of a sum of outer products of vectors that can be computed and stored explicitly. Besides solving $H\mathbf{v} = \mathbf{b}$ iteratively, in the work [15], block diagonal matrices approximation of H are derived so that direct inverses of the block diagonal matrices are less expensive to compute. All the previous work optimizes a single objective and did not aim to speed up the predictor-corrector framework for MOO.

Other empirical and theoretical comparative analyses of MINRES and CG have been performed (see e.g., [5]). Previous work has also demonstrated the effectiveness of MINRES for MOO (see e.g., [13]) as it is a matrix-free solver that guarantees a monotonic decrease in the norm of the residual (of the approximate solution for (3)). However, to our knowledge, the present study will be the first analysis of

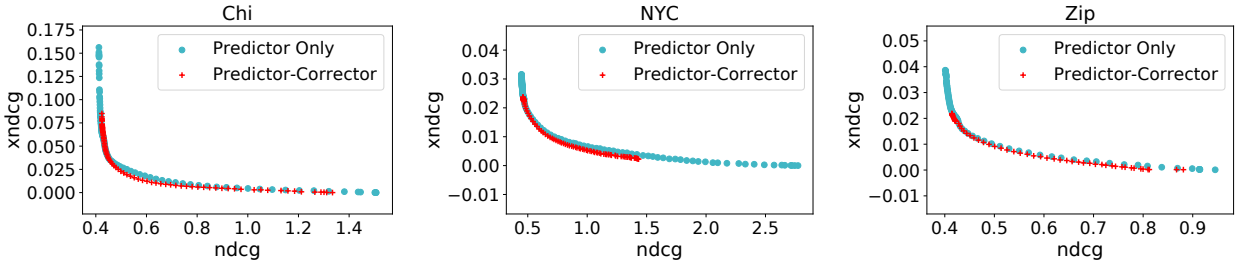


Fig. 7: Ablation study performed to analyze the impact of the corrector step.

the computational time and quality of the Pareto front when using CG and MINRES for applications arising in multiple-objective optimization. In [13], a comparative analysis is performed when varying the maximum number of iterations of the iterative solver. However, in the present study, we show that we can achieve an accurate Pareto front with even fewer maximum iterations than those considered in [13].

VI. Conclusion

We aim to find Pareto fronts that represent multiple trade-offs between multiple objective functions. Previous MOO methods start from a random initial solution or rely on second-order derivatives, thus they are inefficient. We propose a first-order approach using the Gauss-Newton approximation to remove the need for second-order derivatives, and embed the approach in the predictor-corrector method that generates Pareto fronts by exploring the manifold with high efficiency. We apply the method to a fake review detection task on three datasets, and demonstrate the proposed methods can find high-quality Pareto fronts using less time.

Acknowledgement

Sihong Xie is supported in part by the National Science Foundation under Grants NSF IIS-1909879, NSF CNS-1931042, NSF IIS-2008155, and NSF IIS-2145922. Sean Wang is supported by the National Science Foundation under Grants NSF CNS-2051037. Any opinions, findings, conclusions, or recommendations expressed in this document are those of the author(s) and should not be interpreted as the views of the National Science Foundation.

References

[1] Kai Burkholder, Kenny Kwock, Jiaxin Liu, and Sihong Xie. Certification and trade-off of multiple fairness criteria in graph-based spam detection. In *CIKM 2021*, 2021.

[2] Enyan Dai and Suhang Wang. Say No to the Discrimination: Learning Fair Graph Neural Networks with Limited Sensitive Attribute Information. In *WSDM*, 2021.

[3] J. Fliege, A.I.F Vaz, and L N Vicente. Complexity of gradient descent for multiobjective optimization. Technical report, 2018.

[4] Jörg Fliege and Benar Fux Svaiter. Steepest descent methods for multicriteria optimization. *Mathematical Methods of Operations Research*, 51(3):479–494, 2000.

[5] David Chin-Lung Fong and Michael Saunders. Cg versus minres: An empirical comparison. *SQU Journal for Science*, 2012.

[6] Magnus R. Hestenes and Eduard Stiefel. Methods of conjugate gradients for solving linear systems. *Journal of research of the National Bureau of Standards*, 49:409–435, 1952.

[7] C Hillermeier. Generalized Homotopy Approach to Multiobjective Optimization. *Journal of Optimization Theory and Applications*, 110(3):557–583, 2001.

[8] Miettinen Kaisa. *Nonlinear Multiobjective Optimization*, volume 12 of *International Series in Operations Research & Management Science*. Kluwer Academic Publishers, Boston, USA, 1999.

[9] Toshihiro Kamishima, Shotaro Akaho, Hideki Asoh, and Jun Sakuma. Fairness-Aware Classifier with Prejudice Remover Regularizer. In *Machine Learning and Knowledge Discovery in Databases*, pages 35–50, 2012.

[10] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[11] Xi Lin, Hui-Ling Zhen, Zhenhua Li, Qing-Fu Zhang, and Sam Kwong. Pareto Multi-Task Learning. In *NeurIPS*, volume 32, 2019.

[12] Suyun Liu and L N Vicente. The stochastic multi-gradient algorithm for multi-objective optimization and its application to supervised machine learning. Technical report, 2019.

[13] Pingchuan Ma, Tao Du, and Wojciech Matusik. Efficient Continuous Pareto Exploration in Multi-Task Learning. In *ICML*, 2020.

[14] Debabrata Mahapatra and Vaibhav Rajan. Multi-Task Learning with User Preferences: Gradient Descent with Controlled Ascent in Pareto Optimization. In *ICML*, 2020.

[15] James Martens and Roger Grosse. Optimizing Neural Networks with Kronecker-Factored Approximate Curvature. ICML, 2015.

[16] Adanay Martín and Oliver Schütze. Pareto Tracer: a predictor–corrector method for multi-objective optimization problems. *Engineering Optimization*, 2018.

[17] Quentin Mercier, Fabrice Poirion, and Jean-Antoine Désidéri. A stochastic multiple gradient descent algorithm. *European Journal of Operational Research*, 271(3):808–817, 2018.

[18] Barak A Pearlmutter. Fast Exact Multiplication by the Hessian. *Neural Comput.*, 6(1):147–160, 1994.

[19] Yousef Saad. *Iterative Methods for Sparse Linear Systems, 2nd Ed.* SIAM, 2003.

[20] Yousef Saad and Martin H. Schultz. GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 7(3):856–869, 1986.

[21] Levent Sagun, Utku Evci, V Ugur Guney, Yann Dauphin, and Leon Bottou. Empirical analysis of the hessian of over-parametrized neural networks. *arXiv preprint arXiv:1706.04454*, 2017.

[22] Ozan Sener and Vladlen Koltun. Multi-Task Learning as Multi-Objective Optimization. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

[23] Henk A. van der Vorst. *GMRES and MINRES*, page 65–94. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, 2003.

[24] Yongkai Wu, Lu Zhang, and Xintao Wu. On Convexity and Bounds of Fairness-Aware Classification. In *The World Wide Web Conference*, WWW '19, 2019.

[25] Muhammad Bilal Zafar, Isabel Valera, Manuel Gomez Rodriguez, and Krishna P Gummadi. Fairness Beyond Disparate Treatment and Disparate Impact: Learning Classification Without Disparate Mistreatment. In *WWW*, 2017.

[26] Muhammad Bilal Zafar, Isabel Valera, Manuel Gomez-Rodriguez, and Krishna P. Gummadi. Fairness constraints: A flexible approach for fair classification. *Journal of Machine Learning Research*, 2019.