HyperASPO: Fusion of Model and Hyper Parameter Optimization for Multi-objective Machine Learning

Aswin Kannan
IBM Research
India
aswkanna@in.ibm.com

Anamitra Roy Choudhury
IBM Research
India
anamchou@in.ibm.com

Vaibhav Saxena
IBM Research
India
vaibhavsaxena@in.ibm.com

Saurabh Raje
University of Utah
United States
saurabh.mraje@gmail.com

Parikshit Ram
IBM Research
United States
parikshit.ram@ibm.com

Ashish Verma
IBM Research
United States
ashish.verma1@ibm.com

Yogish Sabharwal
IBM Research
India
ysabharwal@in.ibm.com

***Abstract*—Current state of the art methods for generating Pareto-optimal solutions for multi-objective optimization problems mostly rely on optimizing the hyper-parameters of the models (HPO - hyper-parameter Optimization). Few recent, less studied methods focus on optimizing over the space of model parameters, leveraging the problem specific knowledge. We present a generic first-of-a-kind method, referred to as HyperASPO, that combines optimization over the spaces of both hyper-parameters and model parameters for multi-objective optimization of learning problems. HyperASPO consists of two stages. First, we perform a coarse HPO to determine a set of favorable hyper-parameter configurations. In the second step, for each of these configurations, we solve a sequence of weighted single objective optimization problems for estimating Pareto-optimal solutions. We generate the weights in the second step using an adaptive mesh constructed iteratively based on the metrics of interest, resulting in further refinement of Pareto frontier efficiently. We consider the widely used XGBoost (Gradient Boosted Trees) model and validate our method on multiple classification datasets. Our proposed method shows up to 20% improvement over the hypervolumes of Pareto fronts obtained through state of the art HPO based methods with up to $2\times$ reduction in computational time.**

***Index Terms*—Hyperparameter optimization, Model parameters, XGBoost, HyperASPO, Pareto Optimization**

I. Introduction

Machine learning problems in various domains are often inherently multi-objective [1], requiring a model to simultaneously perform well with respect to multiple objectives $f_1, \ldots, f_m$. In classification problems with imbalanced classes (common in finance domains), we would want the model to have high accuracy as well as high precision, recall and the per-class F1-scores [2], [3]. Computer vision applications often require both high accuracy and (adversarial) robustness [4]. Machine translation problems often desire models that simultaneously possess high BLEU [5], METEOR [6] and RIBES [7] scores. This necessitates multi-objective optimization (MOO) of relevant variables $z \in \mathcal{Z}$ of the learning problem defined as:

$$\min_{z \in \mathcal{Z} \subset \mathbb{R}^d} \quad \mathbf{F}(z) = \{f_1(z), \ldots, f_m(z)\}. \tag{1}$$

These optimization problems do not generally have a single solution, but a family of solutions, usually denoted by a Pareto-efficient frontier (or *Pareto front*) in the space of objectives $\mathcal{S} \subset \mathbb{R}^m$ that corresponds to the best trade-offs between the objectives. The Pareto front consists of a set of *non-dominated points* called Pareto solutions – for any two Pareto solutions z_1^* and z_2^*, there exist $i, j \in [m] = \{1, \ldots, m\}$ such that $f_i(z_1^*) < f_i(z_2^*)$ and $f_j(z_1^*) > f_j(z_2^*)$. The Pareto front provides the user with useful insights into the problem, and lets her select different solutions on the Pareto frontier based on the current requirements at hand. Pareto fronts are quantitatively evaluated using the *hypervolume* (HV) metric [8], [9].

A. Related Work

Multi-objective learning of model parameters (MP), denoted by $\theta \in \Theta$, is often handled by *scalarizing* the multiple objectives into a weighted single objective $f = \sum_{i=1}^m w_i f_i$ for some weights $\{w_i, i \in [m]\} \in \Delta^m$, the standard simplex [10]. These weights could be chosen with domain expertise [11] or by optimizing for the worst-case [12]. However, such techniques do not generate a Pareto front, and at best obtain a single point on it. A dense Pareto front can be naively obtained by considering a uniform mesh over Δ^m and optimizing the scalarized objective with weights $w_i, i \in [m]$ corresponding to points on the mesh. However, this is computationally impractical especially for $m > 2$ objectives. Pareto optimization with an adaptive mesh over Δ^m is able to scale to larger number of objectives by starting with a coarse grid over Δ^m and refining it adaptively to ensure uniform density on the Pareto front in the objective space $\mathcal{S} \subset \mathbb{R}^m$ [13]. However, this relies on the ability to map points in the objective space $\mathcal{S}$ to the space of the model parameters $\mathcal{Z}$ – for any set of weights $\{w_i, i \in [m]\} \in \Delta^m$, we can easily obtain $z^* = \arg\min_{z \in \mathcal{Z}} \sum_{i=1}^m w_i f_i(z)$ and obtain the corresponding point $F(z^*) \in \mathcal{S}$; for a given

978-1-6654-3902-2/21/$31.00 ©2021 IEEE

$\{w_i^\star, i \in [m]\} \in \Delta^m$ and $F^\star \in \mathcal{S}$, the above scheme heavily relies on the ability to find some $z^\star \in \mathcal{Z}$ such that $F(z^\star) = F^\star$. In learning problems, we usually optimize MP with respect to some objective; it is usually *not possible to learn MP to get a specified objective value*. For stochastic regimes, Gaussian methods [10] successfully approximate the Pareto frontier for biobjective ($m = 2$) problems, but become computationally expensive for higher number of objectives since they require multiple first and second-order gradients in each optimization step. Hence, applying such methods to large scale learning problems is quite challenging.

Multi-objective learning also manifests in *multi-task learning* (MTL) [14], [15] with the goal of simultaneously learning the model parameters θ with data from multiple related tasks to improve performance across all tasks, motivating the use of Pareto-optimization [16]. However, MTL usually has two structural assumptions: (i) the base objectives $f_i, i \in [m]$ represent the application of a single objective function to distinct data sets, and (ii) the model for each task has a task-specific set of MP and a set of MP shared across all tasks, effectively resulting in a unique model per objective. In general multi-objective learning, the objectives will have different forms (accuracy vs adversarial robustness), and the same model is to be optimized across all the objectives.

General objectives can be effectively handled via *black-box optimization* techniques such as genetic algorithms or Bayesian optimization (BO). These techniques rely solely on the ability to evaluate the objectives at any specific value of the optimized variables. Genetic algorithms [17], [18] use Pareto dominance to update a population of solution candidates. Other related search based schemes [19] operate iteratively by adopting descent directions along coordinates and simplexes. Multi-objective Bayesian Optimization (MOBO) methods [20]–[22] build surrogate models for objective functions and iteratively select a single or batch of solution candidates that are expected to most improve the Pareto front by either directly trying to maximize the expected HV improvement or by maximizing information gain about the Pareto front.

These black-box optimization schemes are very general in terms of the forms of objectives and have been shown to be very useful in various MOO applications outside of multi-objective learning. However, these black-box techniques are usually applicable to problems with a small number of variables d (usually $d \sim 10$). Hence, in the context of multi-objective learning, they are not suitable for optimization of the larger set of model parameters $\theta \in \Theta$ and have instead been used for the *optimization of the smaller set of hyper-parameters* (HP) denoted by $\lambda \in \Lambda$. For any given HP, the model parameter optimization (MPO) is performed with only a single objective which is usually a standard loss function such as log-loss or mean squared error but can also be any other custom objective function. However, the MPO does not consider all the objectives of interest. We further elaborate the optimization based on HP in section II.

B. Motivation & Contributions

As the preceding discussion highlights, in the most general form, the optimized variables z in multi-objective learning must include the HPs λ and the corresponding MP θ_λ, which is a challenging task. Hence, multi-objective hyper-parameter optimization (HPO) just focuses on λ and optimizes for θ_λ in a single-objective manner. For learning with $m > 2$ general objectives, HPO is currently the most appropriate tool. However, HPO can be computationally quite expensive for multi-objective learning for two main related reasons: (i) Each HP corresponds to a single point in the objective space $\mathcal{S} \subset \mathbb{R}^m$; generating a Pareto front with high HV in a multi-dimensional $\mathcal{S}$ will require a large number of evaluations of the objectives, each corresponding to a single HP, requiring as many MPO from scratch. (ii) When evaluating a large number of HPs, it is quite possible that very computationally expensive HP configurations are selected (for example, very deep decision trees or very deep & wide neural networks).

However, learning problems have more structure that we can leverage for improved efficiency. For example, learning within a single model class (corresponding to a fixed HP $\lambda \in \Lambda$) or MPO for $\theta_\lambda \in \Theta_\lambda$ is often performed with gradient-based optimization which are significantly more efficient and have better convergence than black-box optimization. Moreover, in most learning methods, given a fixed HP λ, the computational cost of the MPO is often agnostic to the objective, allowing better control over the computational cost of the multi-objective learning; consider construction of a decision tree of depth 10 or training of a 4-layer neural network for 10 epochs. Furthermore, operating in a single model class allows us to further improve computational efficiency by leveraging techniques such as warm-starts when appropriate.

This motivates us to focus on the following questions in this paper: (i) *Can we generate high quality Pareto fronts for $m \geq 2$ general objectives leveraging gradient-based MPO within a single model class (that is, for a fixed HP)?* (ii) *Given that HP provide an additional degree of freedom for learning algorithms, can we leverage the advantages of multi-objective HPO to further improve the generated Pareto fronts?* To address the above questions, we make the following contributions:

- For a fixed HP, we present a novel algorithmic framework `ASPO` (**A**daptively **S**calarized **P**areto **O**ptimization) that can leverage *gradient-based MPO* and *adaptive multi-dimensional mesh refinement* to efficiently generate Pareto fronts with high HV for $m \geq 2$ general objectives with *any machine learning method*. Theoretically, we show that this Pareto front has the maximal HV among all scalarized schemes under certain regularity conditions.
- We present a novel general method `HyperASPO` that combines, for the first time, HPO and MPO in multi-objective optimization, where *both HPO and MPO explicitly utilize MOO*. `HyperASPO` first performs a coarse multi-objective HPO and then employs `ASPO` on computationally cheap and promising HPs to efficiently generate high quality Pareto fronts.

- We focus on the specific application of ASPO and HyperASPO using Gradient Boosted Decision Trees (specifically XGBoost [23]), which is a widely used and very popular machine learning model for tabular data. We note that our scheme extends to any machine learning model and is not tied to XGBoost alone. We demonstrate the empirical advantage of ASPO and HyperASPO over existing multi-objective HPO schemes on multiple classification problems and objectives, demonstrating up to 20% HV improvement and $\sim 2\times$ speedup over the most recent state-of-the-art [20], [22].

The paper is organized as follows: We present the precise optimization problem, the multi-objective learning setup, and our novel algorithmic frameworks ASPO and HyperASPO in Section II. We extend our framework to multiple dimensions (greater than three) in Section III. In Section IV, we quantify the utility of our novel frameworks using XGBoost and multiple classification data sets. We finally conclude in section V.

II. Algorithm Description

We consider multi-objective learning with m objectives $f_j, j \in [m]$ where the optimization variables z correspond to the hyper-parameters (HP) of the learner $\lambda \in \Lambda$ **and** the model parameters (MP) $\theta_\lambda \in \Theta_\lambda$; Θ_λ is the model class defined by HP λ. We focus on supervised learning with a set S of examples, with examples $(x, y) \in S$ drawn IID from some unknown distribution $\mathcal{D}$. Each objective $f_j, j \in [m]$ corresponds to the expected loss with respect to a per-sample loss ℓ_j, defined as $f_j(z) \stackrel{\text{def}}{=} \mathbb{E}_{(x,y)\sim\mathcal{D}}\, \ell_j(\theta_\lambda; (x, y))$. The per-sample loss function can be objectives such as logistic loss (in classification), squared loss (in regression), as well as non-decomposable learning objectives such as area under the precision-recall curve and related metrics via decomposable approximations [24].

We denote with $\mathbf{w} = \{w_j, j \in [m]\} \in \Delta^m$ the objective weights on a simplex for scalarizing the multi-objective $\mathbf{F}(z)$. For the purposes of learning, we split the set S into a training set S_t, used to learn the MP $\theta_\lambda \in \Theta_\lambda$ in the model class specified by a HP λ, and a held-out validation set S_v, used for the HPO and for generating the final Pareto front in the objective space $\mathcal{S}$. Let $z_j^\star = \arg\min_{z\in\mathcal{Z}} f_j(z)$ be the per-objective solutions. Then $\mathbf{F}^{\mathrm{U}} = \{\mathbf{F}(z_j^\star), j \in [m]\} \subset \mathcal{S}$ is the set of *Utopia points* in the objective space $\mathcal{S}$.

Given a set of examples S split into S_t and S_v, we wish to optimize for multiple objectives simultaneously with respect to the HP $\lambda \in \Lambda$ and the corresponding MP $\theta_\lambda \in \Theta_\lambda$. Let $\widehat{\mathcal{D}}_S$ denote the empirical distribution over a set S. In current multi-objective HPO, for a selected HP λ, the MP are learned using some loss function ℓ_0 (it could be one of the $\{\ell_j, j \in [m]\}$), that is:

$$\text{HPO:} \qquad \theta_\lambda \leftarrow \arg\min_{\theta\in\Theta_\lambda} \mathop{\mathbb{E}}_{(x,y)\sim\widehat{\mathcal{D}}_{S_t}} \ell_0(\theta; (x, y)).$$

Given θ_λ, the point $\mathbf{F}(\{\lambda; \theta_\lambda\}) \in \mathcal{S}$ is generated as $\{\mathbb{E}_{(x,y)\sim\widehat{\mathcal{D}}_{S_v}} \ell_j(\theta_\lambda; (x, y)), j \in [m]\}$. This highlights that a single HP generates a single point in the objective space $\mathcal{S}$ in current multi-objective HPO. As we motivated in Section I, we wish to generate multiple points in $\mathcal{S}$ from the same model class Θ_λ (corresponding to the HP λ). We achieve that by generating multiple objective weights $\mathbf{w}^{(i)} \in \Delta^m, i \in [k]$ and learning MP as follows.

$$\text{MPO:} \qquad \theta_\lambda^{\mathbf{w}^{(i)}} \leftarrow \arg\min_{\theta\in\Theta_\lambda} \mathop{\mathbb{E}}_{(x,y)\sim\widehat{\mathcal{D}}_{S_t}} \sum_{j=1}^{m} w_j^{(i)} \ell_j(\theta; (x, y)).$$

Then $\{\theta_\lambda^{\mathbf{w}^{(1)}}, \ldots, \theta_\lambda^{\mathbf{w}^{(k)}}\}$ generates multiple (in this case, k) corresponding points in the objective space $\mathcal{S}$ from the same model class Θ_λ. One of our key contributions is a scheme to generate these sequence of weights $\mathbf{w}^{(1)}, \ldots, \mathbf{w}^{(k)} \in \Delta^m$ in a way that maximizes the HV of the consequent Pareto front in $\mathcal{S}$, building upon [13]. In the remainder of this section, (i) we first present the algorithmic framework of ASPO that, for a fixed HP λ, adaptively generates a sequence of objective weights and performs a sequence of MPO to generate a high quality Pareto front with a single model class Θ_λ, and (ii) we then explain how ASPO can be used in conjunction with any multi-objective HPO scheme to develop the HyperASPO framework for simultaneously optimizing for HP λ and corresponding MP θ_λ, and efficiently determining Pareto fronts. As an example, in the case of XGBoost, hyper-parameters can refer to maximum number of nodes, maximum tree depth, number of rounds, etc. The model parameters are learned during the training and include branching thresholds associated with nodes of the trees (for example, feature$_1$ $>$ 15). Note that the final model prediction is obtained by combining the predictions of individual trees used in the ensemble.

A. ASPO *Framework*

The algorithm takes as input the total budget specified in terms of the number of Pareto solutions to consider (evaluate), denoted by N. Our approach comprises of four main steps as described below.

- *Initialization:* Compute an initial mesh of quadrilaterals in the objective space.
- *Step 1:* Refine the patches to generate a mesh of new points. We follow [13] to define these meshes in the space of objectives (not model parameters).
- *Step 2:* Map the mesh points from the objective to input space (i.e., machine learning model parameters). This is done by using the weights to scalarize the multi-objective problem formulation and then using this scalarized formulation to learn the model parameters. This set of model parameters yields a candidate Pareto solution.
- *Step 3:* Identify and drop points that are very close to each other.

The initialization step is executed once to generate an initial mesh in the objective space. The next three steps are executed iteratively for T iterations, where T ($1 \le T \le N$) is a hyperparameter of our procedure. For the ease of exposition, we analyze a running example with three objectives. We discuss generalization to higher dimensions (> 3 objectives) in section III.

Initialization Step: Compute an initial mesh. We start by computing the Utopia points. These Utopia points form a primitive Pareto surface. For the case of three objective functions, there are three Utopia points and thus this surface is a triangle. We split this surface into several independent quadrilaterals that form our initial mesh of patches. An example of such an initial mesh generation is illustrated in Figure 1a. We partition the triangle into independent quadrilaterals: Q_1, Q_2 and Q_3 as shown in the figure[1].

Step 1: Patch Selection and Mesh Refinement. In this step, we examine all the patches (quadrilaterals) generated in the previous iteration and refine them further introducing more points in the objective space. We introduce a total of $n_t = N/T$ points in each iteration, where N and T are the input budget (number of evaluations) and number of iterations respectively. Recall that each patch is defined by four vertices and each vertex refers to a set of weights in the objective space. Consider a patch generated in the previous iteration – let it be defined by the four vertices $f^{(a)}, f^{(b)}, f^{(c)}$ and $f^{(d)}$ (in cyclic order). The patch is refined by introducing n_v points in the form of a uniform two dimensional mesh. Here, n_v is computed as the ratio of the area of the patch, A_{abcd}, to the area of the initial surface defined by the utopia points, A_{total}; this way we generate more points in the larger patches resulting in better uniformity of the mesh at the end of each iteration. We choose two interpolation interval parameters, α_h for the horizontal and α_v for the vertical. As an example in figure 1a, in quadrilateral Q_1, α_h corresponds to the interval along the edge defined by vertices U_2 and P_4 and α_v refers to the interval along the edge defined by vertices U_2 and P_7. The patch is refined by introducing $n_v = ((1/\alpha_h)+1)\cdot((1/\alpha_v)+1)$ points in the form of a uniform two dimensional mesh; for $0 \le i \le (1/\alpha_h)$ and $0 \le k \le (1/\alpha_v)$, the $(i,k)^{th}$ point $\widehat{f}^{(i,k)}$ is determined from its dimension-wise components:

$$\widehat{f}_j^{(i,k)} = (1-i\alpha_h)\left((1-k\alpha_v)f_j^{(a)} + (k\alpha_v)f_j^{(d)}\right) + (i\alpha_h)\left((1-k\alpha_v)f_j^{(b)} + (k\alpha_v)f_j^{(c)}\right),$$

$$\alpha_v = \alpha_h = \frac{1}{\sqrt{n_v}-1},\ n_v = \max\left(\left(\left\lceil\left\{\sqrt{n_t\cdot\frac{A_{\text{abcd}}}{A_{\text{total}}}}\right\}\right\rceil\right)^2, 4\right).$$

Note that four of these points map to the patch vertices and are known without entailing further computation. In this work, we simply state a symmetric case of the individual patch budget being a square number and with $\alpha_h = \alpha_v$; ideally, the meshing can vary along different edges and the generalization accommodates for the same. Figure 1b demonstrates this mesh refinement. The mesh is kept slightly coarse just for illustration and readability. As an example, in quadrilateral Q_1, for $\alpha_h = 0.5$, P_9 corresponds to the interpolated point between vertices U_2 and P_7. The vertices U_2 and P_7 refer to the vertex pair $\{a, d\}$ in our notation and P_9 corresponds to one point j with interpolated value $\widehat{f}_j^{(i,k)}$. Note that $i = 0$ and $k = 1$ for the above case. Let weight w_j correspond to the point denoted by function value f_j and let weights $\widehat{w}_j^{(i,k)}$ denote the weights corresponding to interpolated points $\widehat{f}_j^{(i,k)}$. The weights for these interpolated points are determined from the weights of the vertices similarly as:

$$\widehat{w}_j^{(i,k)} = (1-i\alpha_h)\left((1-k\alpha_v)w_j^{(a)} + (k\alpha_v)w_j^{(d)}\right) + (i\alpha_h)\left((1-k\alpha_v)w_j^{(b)} + (k\alpha_v)w_j^{(c)}\right).$$

Step 2: Map from Objective space to Input space. While the previous step interpolated values within the quadrilaterals in the objective space, in this step we actually map these points back to find out the exact underlying models that map to these values. It is noted that finding a machine learning model that exactly matches an interpolated point (function value) is not straightforward. In several cases, this may not even present a feasible solution, leave alone a 1-1 correspondence. To alleviate this problem, we allow for a slight deviation and minimize an approximate function to obtain a feasible solution (model). This is done by defining the following modified interpolation problem:

$$\begin{aligned} \min_{z\in\mathcal{Z},\epsilon\ge 0} \quad & \rho\epsilon + \sum_{j=1}^{m} w_j f_j(z) \\ \text{subject to} \quad & \left|\eta f_j(z) - \widehat{f}_j\right| \le \epsilon \forall j \in [m]. \end{aligned} \tag{2}$$

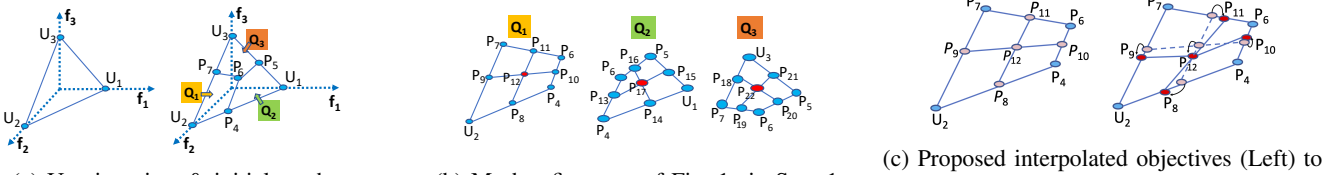
Note that η and ϵ allow for deviation from the interpolated function values. Note that $\widehat{f}$ denotes an interpolated point that we are trying to map back and $f(z)$ denotes the actual solution for model parameters z. While η is a hard multiplicative relaxation parameter allowing for very little deviation from a scaled $\widehat{f}$, ϵ is more soft. We try to minimize the error parameter ϵ by including it in the objective with a suitable scaling parameter ρ. Using the interioricity properties [25] of the above constraint in f_j, we solve equivalent penalized or augmented Lagrangian versions of problem (2).

$$\min_{z\in\mathcal{Z}} \sum_{j=1}^{m}\left(w_j f_j(z) + \rho(\eta f_j(z) - \hat{f}_j)^2\right).$$

The difference between the actual solutions (f) and interpolated points ($\widehat{f}$) is illustrated in figure 1c. In practice, $\eta < 1$ is kept between 0.9 to 0.95 in the initial stages and close to 1 in the later stages of the algorithm (finer meshes). Intuitively this helps in exploring the domain better by allowing for slightly larger error in the initial stages.

Step 3: Remove redundant solutions. Though relatively rare, different weight choices can lead to similar or same solutions. To eliminate such points, we use Euclidean distance as a measure of separation. If $\|\mathbf{F^2} - \mathbf{F^1}\| \le \delta$ for a suitable threshold δ where $\mathbf{F}^2$ is the solution from the current iteration and $\mathbf{F^1}$ is a previously computed solution, we drop $\mathbf{F^2}$ from further consideration. We note that each solution computed by solving our subproblems can be theoretically guaranteed to be optimal only under some assumptions. At the same time, we have the advantage of using closed form expressions of the

[1]This can be done in many different ways - any scheme should work. In this case, we introduce four additional points P_4, P_5, P_6, and P_7 by interpolating between the Utopia points to obtain the three independent quadrilaterals: Q_1, Q_2 and Q_3. P_4, P_5, and P_7 are obtained by taking pairwise means of the triangle vertices and P_6 is obtained by taking the mean of all the 3 vertices.



(a) Utopia points & initial mesh

(b) Mesh refinement of Fig. 1a in Step 1

(c) Proposed interpolated objectives (Left) to actual objectives from solving (2) in Step 2 (Right)

Fig. 1: The ASPO progression

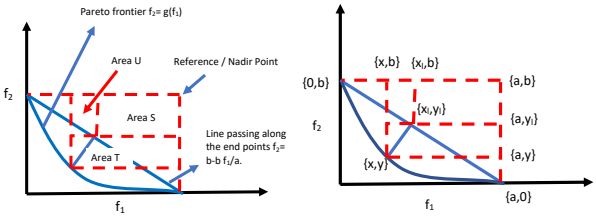


Fig. 2: Graphical view of the biobjective optimization subproblem in ASPO.

custom metrics of interest and their gradients, thus making the empirical performance good.

The mapping of the solution back from the objective space to input space is not straightforward with other adaptive models like [13]. These methods [13] further entail some assumptions on the closed form expressions of the objectives of interest and impose hard constraints. In the context of training machine learning models, direct application of these schemes is intractable. While learning based problems present complex functionals, for proof of concept, we provide some theoretical backing on the intuition behind our method by looking at a biobjective case as follows.

Theorem 1. *Consider a biobjective optimization problem* $\min_x F(x) = \{f_1(x), f_2(x)\}$ *with continuously differentiable and bounded* $F(x)$*. Let the underlying Pareto frontier be convex. Then, our adaptive scheme that interpolates along the midpoints in space of* `function values` *generates a family of solutions with maximal hyperarea in comparison to any other uniformly weighted scheme.*

The case is depicted succinctly in figure 2. For the ease of notation, we refer to the coordinates of f_1 and f_2 by $\{x, y\}$ and the the Pareto frontier by $y = g(x)$. Let the end points be denoted by intercepts a and b (and connected by a line segment) as shown in the figure. We note that the reference point is the Nadir (worst case) point $\{a, b\}$ and our problem pertains to minimization. We additionally note that the point returned by the adaptive scheme (interpolated) is $\{x, y\}$ and the one closest to the same along the line is $\{x_l, y_l\}$. The hyperarea is hence a sum of areas of three rectangles, $\Delta H = S + U + T$. Here, S denotes the rectangle formed between points $\{x_l, y_l\}$ and a, b, U, and T refer to the areas obtained along $\{x, y\}$ and $\{a, y_l\}$ and $\{x_l, y_l\}$ and $\{x, b\}$ respectively. Hence,

$$S = (b - y_l)(a - x_l) = \frac{b}{a}x(a - x).$$

The above expression is minimized at $x_l = \frac{a}{2}$. It hence suffices to show that the mid-point obtained from interpolation also minimizes areas U and T.

$$\begin{aligned} U + T &= (y_l - y)(a - x) - (b - y_l)(b/a)(y - y_l) \\ &= \left(b - \frac{b}{a}x_l - g(x)\right)\left(a^2 - ax + \frac{b^2}{a}x_l\right). \end{aligned}$$

Note that the above follows by observing that the line normal to L_1 (the line between intercepts) intersects the curve $y = g(x)$ at $\{x, y\}$. It is easy to observe that the equation of the normal line is $b(y - y_l) = a(x - x_l), \quad ay_l = ab - bx_l.$

We note that g(x) is a decreasing convex function such that $g'(x) \leq 0$ and $g''(x) \geq 0$, for all x. This helps in observing that $U + T$ increases from $x = 0$ till $x = a/2$ and subsequently decreases till $x = a$. Noting the concavity of $U + T$ also in the same space, we ascertain that $x_l = \frac{a}{2}$ maximizes $U + T$. Note that the proof follows for any general number of mesh points $m = 2^k$ by a recursive argument on bisection.

B. `HyperASPO` *Framework*

In this section, we present the `HyperASPO` framework for simultaneously optimizing for HP λ & corresponding MP θ_λ. The `ASPO` performance is determined by the quality of the Pareto front returned by the algorithm, as well as the total time/number of evaluations to compute the Pareto front. Both these metrics depend on the choice of the hyper-parameters, thus choosing the appropriate hyper-parameter configuration is critical. Moreover, given a particular budget on time/ number of evaluations, it may be beneficial to run `ASPO` for multiple HP configurations (where each configuration is run for a few number of evaluations) and combining the respective Pareto fronts rather than running `ASPO` for a large number of evaluations for a single HP configuration.

Our `HyperASPO` mechanism is effected by combining the `ASPO` procedure with any of the state-of-the-art HPO methods (for e.g., the MOBO methods). We run the HPO method for some number `hp_eval` of evaluations. This shall perform a coarse HPO to determine a set of favorable HP configurations. Next we sort the above obtained HP configurations in increasing order of the training time using the HPO scheme. Other possible

Algorithm 1 `HyperASPO` Procedure.

1: Input: `ASPO` procedure $\mathcal{A}$, Hyper-parameter optimization scheme $\mathcal{H}$, maximum number `hp_eval` of evaluations/ hyper-parameter configurations of $\mathcal{H}$, time budget T for combined optimization.
2: Output: Pareto front for the MOO problem.
3: Perform a coarse hyper-parameter optimization using $\mathcal{H}$ to determine a set of `hp_eval` favorable hyper-parameter configurations.
4: Sort the above obtained configurations in increasing order of the training time using the HPO scheme.
5: Set $i \leftarrow 1$. Set $t \leftarrow T_{\mathcal{H}}$, where $T_{\mathcal{H}}$ is the time to run $\mathcal{H}$.
6: **while** $t < (T - T_{\mathcal{H}})$ **do**
7: Run $\mathcal{A}$ for configuration i in the sorted list.
8: $t \leftarrow t + t_i$, where t_i denotes the training time of $\mathcal{A}$ for configuration i.
9: $i \leftarrow i + 1$.
10: **end while**
11: Fusion: Combine the i MPO frontiers with the HPO frontier and choose the non-dominated points to return the Pareto front.

ways of ordering can be based on objective values returned by the HPO scheme for the HP configurations, or a naïve way of simply ordering the HP configurations randomly. We choose the top configurations from this sorted list, and return the fusion of the respective Pareto frontiers obtained from the `ASPO` runs for each of them. Note that the performance of `HyperASPO` will depend on the choice of `hp_eval`, as well as the metric used in ordering the HP configurations obtained from HPO. We empirically demonstrate that setting `hp_eval` = 100 and choosing the top HP configurations with minimum training time perform best for `HyperASPO`. The pseudo-code of `HyperASPO` mechanism is presented in Algorithm 1. To the best of our knowledge, this is the first approach that combines optimization over the spaces of both hyper-parameters and model parameters (HPO plus MPO) for multi-objective optimization.

III. Multidimensional Extensions of ASPO

In this section, we illustrate one way to generalize adaptive meshes to more than three dimensions (for handling more than 3 objectives). We note that there could be other possible ways to generate such meshes. The case of uniform meshes is well studied in literature. However, the task becomes non-trivial in the case of adaptive settings, particularly in multiobjective optimization applications. In adaptive instances, the main non-obvious difficulty arises in dividing the entire region of weights into sub-spaces based on volumes corresponding to function values. Also, as a corollary, when these subspaces (usually polygons) are generated arbitrarily, they are not guaranteed to cover the entire space of weights.

For simplicity, we begin with the case of a biobjective problem (figure 3, left). The line is given by end points A and B with respective weights $\{0, 1\}$ and $\{1, 0\}$. The point C in the space of weights corresponds to $\{0.5, 0.5\}$. However in the function space, $F^c = 0.5F^A + 0.5F^B$. The distances 1 and 2 are computed based on the relative distances $F^c - F^a$ and $F^c - F^b$. This further can be refined by looking at line segments AC and BC. Let's assume, point A marks the function values $\{0.2, 0.3\}$ and point B marks $\{0.1, 0.7\}$. Then, point C marks the interpolated point $\{0.15, 0.5\}$. A similar philosophy works

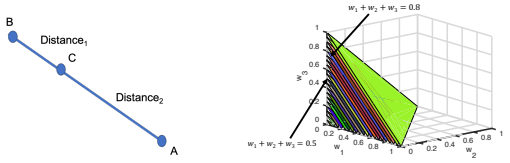


Fig. 3: Non-uniform 2D meshes (Left) and 4D Meshes (Right).

for higher dimensions. A two dimensional version of `ASPO` has also been created based on this methodology. Visualization of the space of weights in dimensions greater than three is harder, since they are defined by a volume. In four dimensions, $\sum_{i=1}^{4} w_i = 1$, denotes a prism with four vertices and is given by figure 3 (right). This in turn translates to $w_1 + w_2 + w_3 \leq 1$ and the volumetric region can also be viewed by a fusion of multiple slices as shown. For our subsequent analysis, we define the convex hull of weights w as $W = \text{conv}(w_1, \ldots, w_m)$, where $m \geq 3$. Let e refer to a vector of ones and e_i point to a coordinate vector with the i^{th} entry equal to 1 and the rest 0. Additionally, let $\bar{v} = \frac{e}{n}$, $v_i = e_i$. Then,

$$W_{-i} = \left\{ w \in W : w \in \text{conv}\left(v_1, \ldots, v_{i-1}, \underbrace{\bar{v}}_{\text{i}}, v_{i+1}, \ldots, v_n \right) \right\}.$$

Note that conv(.) refers to the convex hull. The following theorem acts as a generalization and states that when using all but one coordinate vertices and the centroid, we span the entire surface of weights. We discuss about the volume computation portion post the proof of the theorem.

Theorem 2. *Let W and W_{-i} be as defined above. Then, $\cup_{i=1}^{n} W_{-i} = W$.*

Proof. By definitions of the respective convex hulls, any $w \in W$ and $\bar{w} \in \cup_{i=1}^{n} W_{-i}$ can be represented as follows.

$$w = \sum_{i=1}^{n} \alpha_i v_i, \quad \bar{w} = \sum_{i=1}^{n} \beta_i z_{-i}, \quad \sum_{i=1}^{n} \alpha_i = 1, \quad \sum_{i=1}^{n} \beta_i = 1$$
$$z_{-i} = \sum_{j \neq i} \gamma_{ij} v_j + (1 - \sum_{j \neq i} \gamma_{ij})\bar{v}, \quad \alpha_i, \beta_i, \gamma_i \geq 0.$$

As a contradiction let us assume that $\bar{w} \notin W$. Noting that $\bar{v} = \frac{1}{n} \sum_{j=1}^{n} v_j$ and defining $\delta_{ij} = \gamma_{ij} + \frac{1}{n}(1 - \sum_{j \neq i} \gamma_{ij})$, for $j \neq i$, we have

$$z_{-i} = \sum_{j \neq i} \delta_{ij} v_j + \delta_{ii} v_i.$$

Note that $\delta_{ii} = \frac{1}{n}(1 - \sum_{j \neq i} \gamma_{ij})$. This further leads to the claim that there exists no α such that the following holds for a fixed β and δ:

$$\sum_{i=1}^{n} \alpha_i v_i = \sum_{i=1}^{n} \beta_i \sum_{j=1}^{n} \delta_{ij} v_j, \Longrightarrow \sum_{j=1}^{n} \delta_{ij} \neq 1, \quad \forall i,$$

where the latter follows from the specification of β. Evaluating the expression on the left, we have the following.

$$\begin{aligned} LHS &= \sum_{j=1}^{n} \delta_{ij} = \sum_{j\neq i} \left(\gamma_{ij} + \frac{1}{n}(1 - \sum_{j\neq i} \gamma_{ij}) \right) + \frac{1}{n}(1 - \sum_{j\neq i} \gamma_{ij}) \\ &= \sum_{j\neq i} \gamma_{ij} + \frac{n-1}{n} - \frac{1}{n} \sum_{j\neq i} \sum_{j\neq i} \gamma_{ij} + \frac{1}{n} - \frac{1}{n} \sum_{j\neq i} \gamma_{ij} \\ &= 1 + \sum_{j\neq i} \gamma_{ij} - \frac{n-1}{n} \sum_{j\neq i} \gamma_{ij} - \frac{1}{n} \sum_{j\neq i} \gamma_{ij} = 1. \end{aligned}$$

This disproves the above claim that $\sum_{j=1}^{n} \delta_{ij} \neq 1$. It hence follows that each point z_{-i} is generated from a simplex with all vertices v_i. This in turn leads to the claim that $\bar{w}$ also falls within the set W. Our claim follows and the proof for the theorem is complete. □

Mesh Generation Algorithm: The core of our method lies on how we effectively use the regions with higher concentration of the pareto frontier. We generalize the above adaptive concept of weights to function values. Here, F_{-i} takes the place of W_{-i} and is defined as follows.

$$F_{-i} \in \text{conv} \left(F_1, \ldots, \underbrace{\bar{F}}_{i}, \ldots, F_n \right), \; F_i = \begin{pmatrix} f_1(v_i) \\ \vdots \\ f_n(v_i) \end{pmatrix}.$$

Note that $\bar{F} = 1/n \sum_{i=1}^{n} f_i(v)$. To further refine meshes, we compute the volume of each polytope marked by vertices F_{-i}. Computational budget is accordingly allocated to each of these polytopes based on the ratios of these volumes.

$$V_{-i} = \text{Vol}(F_{-i}) = \frac{\det \left| \begin{pmatrix} F_1 \ldots F_n \\ 1 \ldots 1 \end{pmatrix} \right|}{n!}, \; N_{-i} = \frac{N V_{-i}}{\sum_{i=1}^{m} V_{-i}}.$$

For each volume object F_{-i} (or W_{-i}), we further refine the mesh in weights as follows.

$$v_{-i}^{k} = \sum_{j=1}^{n} v_j \underbrace{\Delta m_j^k}_{\alpha_j}, \; v_j \in W_{-i}, \; \Delta = \frac{1}{m}, \; 0 \geq m_j^k \in \mathbb{Z}. \quad (3)$$

The value of m decides the spacing between any two points and Δ marks the interval. As an example, say if $m = 5$, $\Delta = 0.2$, and m_j takes values between 0 to 5. We specifically note that m_j^k can be explicitly defined as follows.

$$\sum_{j=1}^{n} m_j^k = m, \quad 0 \leq m_j \leq m, \quad m_j^k \in \mathbb{Z}, \quad \forall k \in \{1, \ldots, K\}.$$

It is easy to note that K refers to the number of possible choices of picking $n-1$ bins from $m+n-1$ possibilities. K is also related to the computational budget (denoted by N_{-i}) and can hence be stated as follows.

$$K = \{m + n - 1\} \; C_{n-1}, \quad K \leq N_{-i}.$$

It can be noted that the above complexity result follows the same order of that of uniform meshes (exponential). We can also easily observe that for arbitrary sets W_t of the form $W = \cup_{t=1}^{T} W_t$, Theorem 2 holds. The following remark, justifies the use of our meshing approach (with quadrilaterals) in the previous section's running example.

Remark: Let $n = 3$. If, $W_1 = \{(1/3)e, \; e_1, \; 0.5(e_1 + e_2), \; 0.5(e_1 + e_3)\}$, $W_2 = \{(1/3)e, \; e_2, 0.5(e_1 + e_2), \; 0.5(e_2 + e_3)\}$, and $W_3 = \{(1/3)e, \; e_3, \; 0.5(e_1 + e_3), \; 0.5(e_2 + e_3)\}$. Then, the larger space is divided into 3 quadrilaterals and the theory in section II holds. If, $W_1 = \{(1/3)e, \; e_2, \; e_3\}$, $W_2 = \{(1/3)e, \; e_1, \; e_3\}$, and $W_3 = \{(1/3)e, \; e_2, \; e_1\}$, the space is divided into 3 triangles instead of quadrilaterals.

IV. Experimental Results

We consider multiple learning problems related to binary classification using the popular `XGBoost` model. Our first set of experiments considers problems with tri-objective instances. Table Ia presents the details of the datasets used for these experiments. All the datasets are from the Kaggle benchmark, and can also be found in other sources like UCI Machine Learning repository. For these problems, we consider minimization of three different objectives (metrics), namely Classification Error (1-Accuracy), False Positive Rate (FPR), and Normalized MCC (Mathews Correlation Coefficient) defined as $0.5 \times (1 - MCC)$. The datasets are divided into training, validation and test sets based on stratified partitioning; the percentage split across the three sets are mentioned in the last column of Table Ia for different datasets. Our chosen datasets include very large, large, medium, small, balanced, and (positively and negatively) imbalanced datasets (refer to the second, third and fourth columns of Table Ia); this demonstrates efficacy of our approach across diverse datasets. Additionally, noting the highly imbalanced nature of the HCDR dataset with close to 93 percent negative samples, (see the fourth column of Table Ia), we consider a biobjective version of the HCDR problem to show some interesting behavior for conflicting objectives - the objectives chosen for this setup are precision error and recall error.

Our other set of experiments focus on the study of fairness for machine learning models. Table Ib presents the details of the datasets used for these experiments. These datasets are obtained from the *AI Fairness 360* toolkit [35], and are also quite diverse in terms of number of features and records. Here for each dataset, the training, validation and test set is obtained by a fair split of 64% -16% -20%. The fair split strategy considers both target labels and fairness (sensitive) feature values when performing stratified partitioning. The objectives chosen are Balanced Accuracy Error [36], [37] and a fairness metric referred to as *Demographic Parity Difference* (DPD) [38] or *Calders-Verwer discrimination score* (CV score) [39]. More details on the metrics used in our experimental setup are provided below.

Objectives used in our experiments. For the tri-objective problems, let the three metrics FPR (False Positive Rate), Classification Error, and Scaled Mathews Correlation Coefficient be respectively denoted by `fpr`, `err`, and `mcc`. The Classification Error `err` can also be stated as "1-Accuracy". Note that in place of MCC, we use the Pearson's coefficient and operate in

TABLE I: Datasets for our experiments.

Datasets	No. of Features	No. of Records	Negative/ Positive Samples	Train-Val-Test Split
Donors [26]	11	130K	0.18	70K-30K-30K
German Credit [27]	58	1K	0.43	64%-16%-20%
Amazon [28]	9	32K	0.06	64%-16%-20%
Higgs Boson [29]	28	100K	0.89	64%-16%-20%
HCDR [30]	24	10K	13.45	80%-20%-

(a) Datasets for problems with 3 objectives of Classification Error, FPR and MCC. HCDR is also considered for bi-objective problem with precision and recall as objectives. Separate test data of 10K records is used with HCDR.

Datasets	No. of Features	No. of Records	Fairness Feature
Compas [31]	11	6K	African-American
Adult [32]	98	45K	Sex (Gender)
Law [33]	4	23K	Race
Bank [34]	16	45K	Marital Status

(b) Datasets for Fairness problems with objectives of Balanced Accuracy Error and DPD score. The train-val-test split is a fair split of 64% -16% -20% for all the datasets.

continuously differentiable domains for algorithmic functioning (they are equivalent when the labels are discrete 0/1). For the bi-objective version of the HCDR problem, let the metrics of precision error and recall error be respectively denoted as `prec` and `rec`. Denoting the actual and predicted labels of a particular dataset with m test samples by $\mathbf{y^a}$ and $\mathbf{y^p}$ respectively, the metrics are defined as follows.

$$\texttt{fpr}(\mathbf{y^a}, \mathbf{y^p}) = \frac{\sum_{i=1}^{m}(1-y_i^a)y_i^p}{\sum_{i=1}^{m}(1-y_i^a)}, \quad \texttt{err}(\mathbf{y^a}, \mathbf{y^p}) = \|\mathbf{y^a} - \mathbf{y^p}\|_1.$$

$$\texttt{mcc}(\mathbf{y^a}, \mathbf{y^p}) = \frac{1}{2} \times \left(1 - \frac{\sum_{i=1}^{m}\left(y_i^a - \bar{y}_i^a\right)\left(y_i^p - \bar{y}_i^p\right)}{\sqrt{\sum_{i=1}^{m}\left(y_i^a - \bar{y}_i^a\right)^2}\sqrt{\sum_{i=1}^{m}\left(y_i^a - \bar{y}_i^a\right)^2}}\right).$$

$$\texttt{prec}(\mathbf{y^a}, \mathbf{y^p}) = \frac{\sum_{i=1}^{m}(1-y_i^a)y_i^p}{\sum_{i=1}^{m} y_i^p}, \quad \texttt{rec}(\mathbf{y^a}, \mathbf{y^p}) = \frac{\sum_{i=1}^{m}(1-y_i^p)y_i^a}{\sum_{i=1}^{m} y_i^a}.$$

For the fairness problems, the metric DPD score (or the CV Score) represents the gap between the probabilities of getting positive outcomes by a binary predictor $\hat{Y}$ for two different sensitive groups in the input data. It is defined as:

$$DPD := \left|Pr(\hat{Y} = +|A = 1) - Pr(\hat{Y} = +|A = 0)\right|$$

Here, **A** denotes a binary sensitive attribute or the fairness feature (e.g., gender or race) in the input data with A=1 and A=0 representing privileged and unprivileged groups respectively. Predicted outcome $\hat{Y}$ can be positive (favorable) or negative (unfavorable). A fair predictor will have similar proportions of positive outcomes within each sensitive group resulting in a lower DPD value. Noting that A marks the binary sensitive attribute, we define index sets I_A and J_A referring to the sample points (feature data) that correspond to 0's and 1's of the attribute. The objective takes the following form:

$$I_A = \{j | m \geq j \in Z, A_j = 1\}, \; J_A = \{j | m \geq j \in Z, A_j = 0\}$$

$$\texttt{DPD}(\mathbf{y^a}, \mathbf{y^p}) = \left|\frac{\sum_{i \in I_A} y_i^p}{|I_A|_c} - \frac{\sum_{i \in J_A} y_i^p}{|J_A|_c}\right|.$$

Note that the function $|.|_c$ represents the cardinality of a set, whereas $|.|$ denotes the absolute value function. Finally, the Balanced Accuracy Error metric (which is 1-Balanced Accuracy) is quite standard and is defined as:

$$\texttt{bal\_err}(\mathbf{y^a}, \mathbf{y^p}) = \frac{1}{2|P|_c}\sum_{i \in P}|1 - y_i^p| + \frac{1}{2|N|_c}\sum_{i \in N}|y_i^p|.$$

Also, P and N denote the set of points (labels) classified as positive (1's) ad negative (0's) respectively. We also finally note that the above custom losses are used as-is or deployed with their generic convex approximations (eg: logarithmic smoothing) during training with `ASPO`. HPO schemes train using standard losses like cross entropy and test / validate on the above custom losses (metrics).

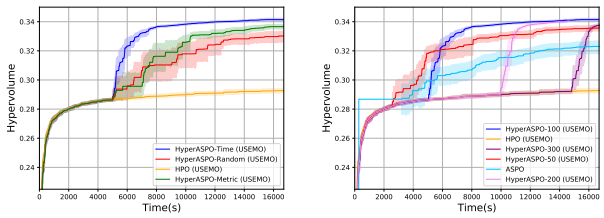
Baselines and Test Bed. We use well established solvers USEMO [20] (State of the art recent Bayesian solver) and NSGA-II [17] (Genetic Algorithm based solver) as our primary baselines. Our choice of USEMO as the baseline is guided by prior art which demonstrate the efficiency of USEMO and Bayesian Optimization schemes in general [20], [40]. Under limited evaluation budget, USEMO is shown to outperform other existing BO schemes, such as ParEGO [40]. For further validation, we also compare our method against another more recent MOBO method based on q-Expected Hypervolume Improvement (qEHVI) [22]. This is available as part of open-source library BoTorch.[2]. Note that our approach is complementary to the underlying HPO scheme, and can be applied in conjunction with any HPO based method. To ensure consistency and repeatability, we run `HyperASPO`, USEMO, NSGA-II, and qEHVI for 15 trials and report the respective averages (and deviations). We considered seven hyper-parameters for XGBoost for our evaluation [3]. Since the methods train on different classes of functions, we deploy the widely accepted hypervolume indicator as our yardstick for comparison. We use unit vector as reference point for hypervolume computation [8], [9] as all the objective values range between 0 and 1 (0-best,1-worst value) for our multi-objective minimization problems. All our reported results correspond to the *unseen test data*. Our implementation of `HyperASPO` is in Python and the computation was performed on Intel Xeon system having processing speed of 3.5GHz.

A. Performance Evaluation

Fusion Choices: Prior to our tests, a preliminary investigation was done on different types of fusion to find the one that is empirically most efficient. Recall that for `HyperASPO` the initial `hp_eval` HP configurations obtained from HPO are sorted based on their training time in the HPO scheme. For this study, we also experimented with other types of sorting, namely the "metric" and "random' sorting. For the "metric" sorting, the HP configurations are sorted in increasing order of

[2]We modified the qEHVI example code based on BoTorch available at https://botorch.org/tutorials/multi_objective_bo using q=4

[3]hyper-parameters tuned for XGBoost are tree depth, maximum leaves, number of rounds, learning rate, min child weight, max delta step and lambda.



(a) Varying sorting scheme. (b) Varying value of `hp_eval`.

Fig. 4: Fusion Types on Donors dataset. `HyperASPO` with around 100 HPO evaluations followed by `ASPO` on the HP configurations with minimum training time performs best.

the highest (worst) objective value, i.e., $\max(\texttt{fpr}, \texttt{err}, \texttt{mcc})$ for the case of the tri-objective problem. The "random" sorting just sorts the `hp_eval` configurations randomly (Uniform Distribution). Figure 4a compares the HV vs. Time plot for the three types of fusion for the Donors dataset. Here `hp_eval` is set to 100; thus all schemes switch from HPO to `ASPO` after 100 evaluations of HPO. It is easy to note that the time based fusion performs best (similar observation is found for other datasets/fairness problems), and we hence proceed with this as our default fusion choice for the subsequent results. On a similar note, we also investigated the impact of varying `hp_eval`, i.e., the switch point from HPO to `HyperASPO`. Figure 4b shows the performance of the fusion scheme for the Donors dataset with the switch happening at 50, 100 (default), 200, and 300 evaluations of HPO. It is easy to observe that the best switch point is the one with 100 evaluations of HPO. It achieves same hypervolume in lesser time compared to higher number of evaluations. Therefore, we keep this as default switch point for our subsequent results. We also show the results with pure ASPO (without fusion) in Figure 4b, where configurations are generated randomly. This corresponds to the extreme case where `hp_eval`=0. We observe that `HyperASPO` is the best of both worlds, with significantly better performance than any one of HPO or Pure ASPO.

Tri-objective Problems: Figure 5a shows results for `HyperASPO`, USEMO, and NSGA-II for Donors dataset for the tri-objective problem. We observe that our fusion based `HyperASPO` scheme performs better than both HPO schemes that saturate beyond a time point. A relative improvement of 20% in hypervolume (HV) is seen with a $2\times$ speedup. Speedup indicates the factor by which `HyperASPO` reduces the time to attain the max HV attained by HPO scheme. Figure 5b, Figure 5c and Figure 5d show the performance for Higgs, Amazon and German Credit datasets respectively. The improvement in HV is close to 20% in the case of both Higgs and German datasets, whereas for the Amazon dataset it is between 8 to 10 percent. A speeedup of atleast $2\times$ is noticed in all cases. This behavior is seen to be consistent with both the HPO schemes. All the datasets exhibit a sudden increase in the HV values for the `HyperASPO` curve; this point actually corresponds to the switch point from HPO to `HyperASPO`.

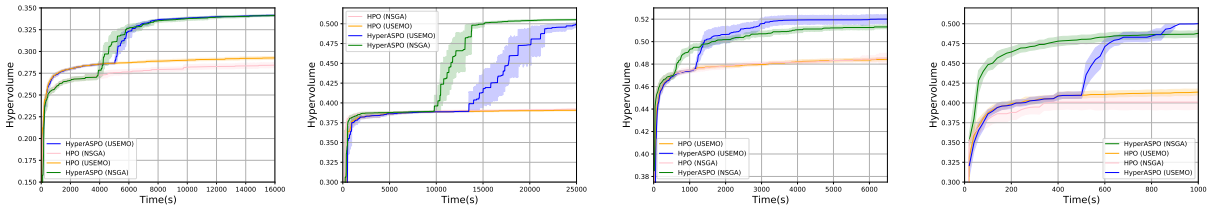
Fairness Problems: The fairness problems are bi-objective as mentioned earlier. Figures 6a and 6b show the results with Bank and Law datasets respectively. While a 20% HV improvement is noticed with the Bank dataset, the Law dataset shows an extraordinary improvement in HV that is more than 100%. For Compas and Adult datasets, we notice an improvement of around 5-10 percent with `HyperASPO`. The incremental improvement in performance for Compas dataset is attributed to the fact that it is a comparatively easier problem and the dataset is also small. The question on speedup requires less attention here since the HPO schemes saturate quite early.

Additional Results: We show the performance of the schemes on the triobjective version of the HCDR problem in Figure 7a. It can be noticed that all schemes saturate at a very low HV. A marginal 1% improvement can however be noticed with `HyperASPO`. The large imbalance in the HCDR dataset (negatives to positives ratio of the order of 14) makes it ill-posed along the space of the three metrics (Error, FPR and MCC turn out to be redundant due to trivial solutions). We therefore consider a bi-objective version of the HCDR problem with two objectives being precision error and recall error. Figure 7b shows a performance improvement of close to 10% in HV and a speedup close to $2\times$ with `HyperASPO` for the bi-objective HCDR problem. To further validate the effectiveness of `HyperASPO` approach, we considered another more recent HPO scheme based on q-Expected Hypervolume Improvement (qEHVI) in `HyperASPO`, and demonstrate the corresponding performance for the Donors and Higgs datasets in Figures 7c and 7d. We observe more than 15% improvement in hypervolume with `HyperASPO` over the qEHVI scheme. Use of `HyperASPO` also provides $1.5\times$ speedup in time over the HPO scheme. These results show that `HyperASPO` can be effectively used with any HPO based method to improve its performance, in terms of improved hypervolume and/or reduction in computation time.

We also analyze the diversity of solutions obtained through `HyperASPO` by studying the spread of its Pareto front, as shown in Figures 7e- 7g. We note that although HPO based approaches lead to good quality solutions, they are mostly concentrated towards one portion of the Pareto frontier as seen in the figure. `HyperASPO` on the other hand covers a larger region by generating solutions close to even the extreme points. This also offers an advantage to designers, where more choices lead to improved feasible designs.

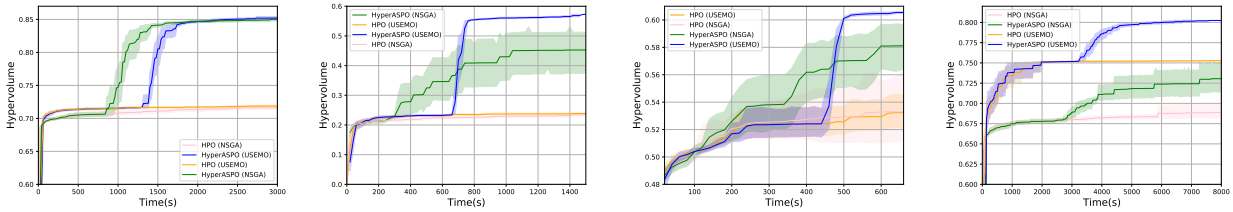
V. Conclusion and Future Work

Traditionally, Bayesian optimization, Search, and Genetic algorithms have been applied to multi-objective problems in machine learning by operating over the hyper-parameter (HP) space. Our work is the first one to apply a fusion of both hyper-parameter and model-parameter optimization over multiple objectives of interest. We leverage the existing HPO methods to obtain a set of favorable hyper-parameters (HPs). For each HP configuration in this set, we explore the space of model parameters. Empirical evaluation of our algorithm on real-world datasets shows an improved performance over current state



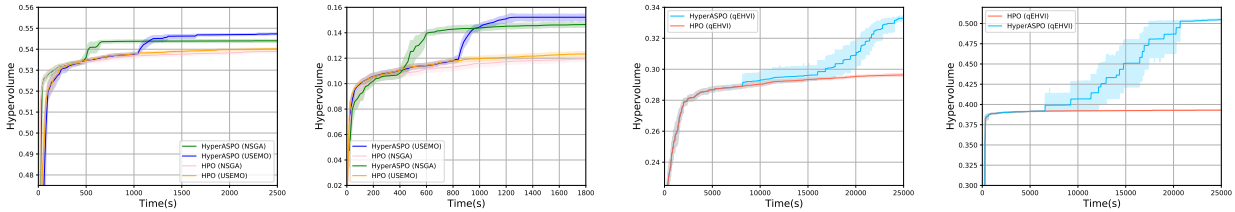
(a) Donors Dataset. (b) Higgs Boson Dataset. (c) Amazon Dataset. (d) German Credit Dataset.

Fig. 5: Results for Tri-objective Problems. HyperASPO (USEMO) (resp. HyperASPO (NSGA)) refers to the scenario when `HyperASPO` is applied on the hyper-parameters generated via USEMO (resp. NSGA-II). Relative improvement of 20% in hypervolume is obtained using `HyperASPO` for Donors, Higgs Boson and German Credit Datasets, and 10% for Amazon. The sudden increase in the HV values for the `HyperASPO` curves correspond to the switch points from HPO to `HyperASPO`. We note again that switch points are the same across all versions of `HyperASPO` and happens at `hp_eval` = 100.

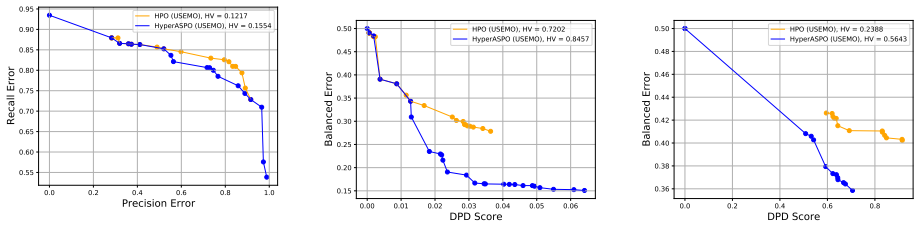


(a) Bank Dataset. (b) Law Dataset. (c) Compas Dataset. (d) Adult Dataset.

Fig. 6: Results for Fairness Problems. Similar observations as in Figure 5. Relative improvement of 5-10% in hypervolume is obtained using `HyperASPO` for Compas and Adult, while improvement of 20% and as high as 100% observed for Bank and Law datasets respectively.



(a) HCDR Tri-objective. (b) HCDR Bi-objective. (c) Donors with qEHVI. (d) Higgs with qEHVI.



(e) Spread Plot for HCDR Bi-obj. (f) Spread Plot for Bank. (g) Spread Plot for Law.

Fig. 7: Additional Results. (a) Large imbalance in HCDR makes it ill-posed for tri-objective case. `HyperASPO` provides (b) better HV for the HCDR Bi-objective version, (c)-(d) improvement over a more recent baseline of qEHVI, and (e)-(g) also shows better spread in the pareto-front compared to corresponding HPO scheme.

of the art HPO based methods. We perform comprehensive empirical evaluation considering many classification problems using highly popular and widely used Gradient Boosted Trees (XGBoost) method. As a good direction for future research, it would be worthwhile to evaluate our method on other learning models such as neural networks. It would also be worth exploring other types of learning problems such as regression and multi-label classification problems.

REFERENCES

[1] Y. Jin, **Multi-objective machine learning**. Springer Science & Business Media, 2006, vol. 16.

[2] G. Forman, "An extensive empirical study of feature selection metrics for text classification," **J. Mach. Learn. Res.**, vol. 3, pp. 1289–1305, 2003. [Online]. Available: http://portal.acm.org/citation.cfm?id=944919.944974

[3] A. Tharwat, "Classification assessment methods," **Applied Computing and Informatics**, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2210832718301546

[4] S. Garg, S. Jha, S. Mahloujifar, and M. Mahmoody, "Adversarially robust learning could leverage computational hardness," 2019.

[5] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: a method for automatic evaluation of machine translation," in **Proceedings of the 40th annual meeting of the Association for Computational Linguistics**, 2002, pp. 311–318.

[6] A. Lavie and A. Agarwal, "Meteor: An automatic metric for mt evaluation with high levels of correlation with human judgments," in **Proceedings of the second workshop on statistical machine translation**, 2007, pp. 228–231.

[7] H. Isozaki, T. Hirao, K. Duh, K. Sudoh, and H. Tsukada, "Automatic evaluation of translation quality for distant language pairs," in **Conference on Empirical Methods in Natural Language Processing**, 2010, pp. 944–952.

[8] C. Audet, J. Bigeon, D. Cartier, S. Le Digabel, and L. Salomon, "Performance indicators in multiobjective optimization," Les cahiers du GERAD, Tech. Rep. G-2018-90, 2018. [Online]. Available: http://www.optimization-online.org/DB_HTML/2018/10/6887.html

[9] L. While, L. Bradstreet, and L. Barone, "A fast way of calculating exact hypervolumes," **IEEE Trans. on Evolutionary Computation**, vol. 16, no. 1, pp. 86–95, 2012. [Online]. Available: http://kriging.sourceforge.net/htmldoc/function/stk_dominatedhv.html

[10] R. Marler and J. Arora, "Survey of multi-objective optimization methods for engineering," **Structural and Multidisciplinary Optimization**, vol. 26, no. 6, pp. 369–395, Apr 2004. [Online]. Available: https://doi.org/10.1007/s00158-003-0368-6

[11] M. Wistuba, A. Rawat, and T. Pedapati, "A survey on neural architecture search," 2019.

[12] C. Cortes, M. Mohri, J. Gonzalvo, and D. Storcheus, "Agnostic learning with multiple objectives," **NeurIPS**, vol. 33, 2020.

[13] I. Y. Kim and O. L. de Weck, "Adaptive weighted sum method for multiobjective optimization: a new method for pareto front generation," **Structural and Multidisciplinary Optimization**, vol. 31, no. 2, pp. 105–116, Feb 2006. [Online]. Available: https://doi.org/10.1007/s00158-005-0557-6

[14] Y. Zhang and Q. Yang, "A survey on multi-task learning," **arXiv preprint arXiv:1707.08114**, 2017.

[15] S. Ruder, "An overview of multi-task learning in deep neural networks," **arXiv preprint arXiv:1706.05098**, 2017.

[16] O. Sener and V. Koltun, "Multi-task learning as multi-objective optimization," in **NeurIPS**, 2018.

[17] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," vol. 6, no. 2, 2002. [Online]. Available: https://doi.org/10.1109/4235.996017

[18] A. Zerbinati, J.-A. Desideri, and R. Duvigneau, "Comparison between MGDA and PAES for Multi-Objective Optimization," INRIA, Research Report RR-7667, Jun. 2011. [Online]. Available: https://hal.inria.fr/inria-00605423

[19] S. Utyuzhnikov, "Multiobjective optimization: Quasi-even generation of pareto frontier and its local approximation," **Handbook of Optimization Theory: Decision Analysis and Application**, pp. 211–235, 01 2011.

[20] S. Belakaria, A. Deshwal, N. K. Jayakodi, and J. R. Doppa, "Uncertainty-aware search framework for multi-objective bayesian optimization," **AAAI Conference on Artificial Intelligence**, vol. 34, no. 06, 2020. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/6561

[21] M. Konakovic Lukovic, Y. Tian, and W. Matusik, "Diversity-guided multi-objective bayesian optimization with batch evaluations," **NeurIPS**, vol. 33, 2020.

[22] S. Daulton, M. Balandat, and E. Bakshy, "Differentiable expected hypervolume improvement for parallel multi-objective bayesian optimization," **arXiv preprint arXiv:2006.05078**, 2020.

[23] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in **Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining**, 2016, pp. 785–794.

[24] E. Eban, M. Schain, A. Mackey, A. Gordon, R. Rifkin, and G. Elidan, "Scalable learning of non-decomposable objectives," in **AISTATS**, 2016.

[25] J. Nocedal and S. Wright, **Numerical optimization**, 2nd ed., ser. Springer series in operations research and financial engineering. New York, NY: Springer, 2006. [Online]. Available: http://gso.gbv.de/DB=2.1/CMD?ACT=SRCHA&SRT=YOP&IKT=1016&TRM=ppn+502988711&sourceid=fbw_bibsonomy

[26] DonorsChoose.org, "The donors dataset," 2018. [Online]. Available: https://www.kaggle.com/donorschoose/io

[27] U. M. Learning, "The german credit dataset," 2017. [Online]. Available: https://www.kaggle.com/uciml/german-credit/version/1

[28] "Amazon.com employee access challenge," https://www.kaggle.com/c/amazon-employee-access-challenge.

[29] P. Baldi, P. Sadowski, and D. Whiteson, "Searching for exotic particles in high-energy physics with deep learning." **Nature communications**, vol. 5, p. 4308, 2014.

[30] H. C. Group, "Home credit dataset," 2018. [Online]. Available: https://www.kaggle.com/c/home-credit-default-risk/data

[31] J. Angwin, J. Larson, S. Mattu, and L. Kirchner, "Machine bias. there's software used across the country to predict future criminals. and it's biased against blacks." https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing, 2016.

[32] D. Dua and C. Graff, "UCI machine learning repository," http://archive.ics.uci.edu/ml, 2017. [Online]. Available: http://archive.ics.uci.edu/ml

[33] L. F. Wightman, "Lsac national longitudinal bar passage study. lsac research report series." 1998.

[34] S. Moro, P. Cortez, and P. Rita, "A data-driven approach to predict the success of bank telemarketing," **Decis. Support Syst.**, vol. 62, pp. 22–31, 2014.

[35] A. F. . toolkit, "Aif360 datasets," https://aif360.readthedocs.io/en/latest/modules/datasets.html#module-aif360.datasets, 2018.

[36] K. H. Brodersen, C. S. Ong, K. Stephan, and J. Buhmann, "The balanced accuracy and its posterior distribution," **2010 20th International Conference on Pattern Recognition**, pp. 3121–3124, 2010.

[37] J. D. Kelleher, B. M. Namee, and A. D'Arcy, **Fundamentals of Machine Learning for Predictive Data Analytics: Algorithms, Worked Examples, and Case Studies**. The MIT Press, 2015.

[38] P. Gajane and M. Pechenizkiy, "On formalizing fairness in prediction with machine learning," **CoRR**, vol. abs/1710.03184, 2017. [Online]. Available: http://arxiv.org/abs/1710.03184

[39] T. Calders and S. Verwer, "Three naive bayes approaches for discrimination-free classification," **Data Min. Knowl. Discov.**, vol. 21, pp. 277–292, 09 2010.

[40] J. Knowles, "Parego: A hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems," **IEEE Transactions on Evolutionary Computation**, vol. 10, no. 1, pp. 50–66, 2006.