Metric Learning to Accelerate Convergence of Operator Splitting Methods for Differentiable Parametric Programming

Ethan King, James Kotary, Ferdinando Fioretto, Ján Drgoňa

***Abstract*— Recent work has shown a variety of ways in which machine learning can be used to accelerate the solution of constrained optimization problems. Increasing demand for real-time decision-making capabilities in applications such as artificial intelligence and optimal control has led to a variety of approaches, based on distinct strategies. This work proposes a novel approach to learning optimization, in which the underlying metric space of a proximal operator splitting algorithm is learned so as to maximize its convergence rate. While prior works in optimization theory have derived optimal metrics for limited classes of problems, the results do not extend to many practical problem forms including general Quadratic Programming (QP). This paper shows how differentiable optimization can enable the end-to-end learning of proximal metrics, enhancing the convergence of proximal algorithms for QP problems beyond what is possible based on known theory. Additionally, the results illustrate a strong connection between the learned proximal metrics and active constraints at the optima, leading to an interpretation in which the learning of proximal metrics can be viewed as a form of active set learning.**

I. INTRODUCTION

A substantial literature has been dedicated to the use of machine learning to aid in the fast solution of constrained optimization problems. This interest is driven by an increasing need for real-time decision-making capabilities, in which decision processes modeled by optimization problems must be resolved faster than can be met by traditional optimization methods. Such capabilities are of interest in various application settings such as job scheduling in manufacturing [26], [25], power grid operation [14], and optimal control [34].

A prominent application of machine learning in accelerating optimization is to learn the parameters of a standard solution algorithm, such that iterations to convergence are minimized. Examples include gradient stepsizes [1], and initial solution estimates [34]. This paper proposes an alternative approach by parametrizing the underlying metric space of an optimization algorithm which relies on *proximal operators*. Proximal operators, which include projections, are based on a notion of distance within a metric space and employed in many practical optimization methods. While most methods that employ proximal algorithms are typically based on the standard Euclidean metric, it is well-known that many such methods are also guaranteed to converge for non-Euclidean metrics defined as general quadratic forms over the continuous space of positive definite matrices [4].

The possibility of accelerating convergence by selecting non-Euclidean metrics within that space has been noted [18], but no known method has shown to be effective over a general class of optimization problems. For limited classes of problems, *optimal* metric choices have been modeled as the solution to an auxiliary optimization problem. But for many problems including general quadratic programming (QP) problems, such models are yet unknown [19]. Theoretical insights have been used to suggest *heuristic* metric choices for QP problems [18], [19], but the potential for improvement over these heuristic rules has not been fully explored.

This paper proposes differentiable programming to both explore the potential, and to overcome the challenges of metric selection for more general classes of optimization problems. Specifically, we propose a system of end-to-end learning for proximal optimization, which trains machine learning models to predict metrics that empirically minimize solution error over a prescribed number of iterations on a given problem instance. Enhanced convergence of two proximal optimization methods is demonstrated on Quadratic Programming (QP) problems, including test cases where theoretically prescribed heuristic metric choices perform poorly.

We demonstrate that while prior heuristic models of optimal metric selection can *fail* in the presence of active constraints at the optimal solution, our learned metrics are *correlated* with the active constraints at optima, and can accelerate convergence by ignoring the inactive constraints. This leads to an interpretation of metric selection as a problem which incorporates active set prediction, whose difficulty may approach that of solving the optimization problem itself. The proposed integration optimization and learning thus shows advantages in both accuracy and efficiency over theoretical approaches to metric selection in proximal optimization.

II. RELATED WORK

This paper's topic is at the intersection of learning to accelerate optimization, and metric selection in proximal optimization. Before proceeding to the main contributions, related work is summarized with respect to both areas.

A. Learning to Accelerate Optimization

Various systems for learning fast solutions to optimization problems have been proposed. For example, several works have shown how to learn heuristics such as branching rules

Ethan King is with the Pacific Northwest National Laboratory, Richland, WA, USA ethan.king@pnnl.gov

James Kotary is with the University of Virginia, Charlottesville, Virginia, USA jk4pn@virginia.edu

Ferdinando Fioretto is with the University of Virginia, Charlottesville, Virginia, USA fioretto@virginia.edu

Ján Drgoňa is with the Pacific Northwest National Laboratory, Richland, WA, USA Jan.Drgona@pnnl.gov

[2], [20], [22] and cutting planes [33] in mixed-integer programming. An early survey [6] provides a comprehensive summary of machine learning in combinatorial optimization. Further surveys on learning to branch [28] and learning to cut [9] provide even more detail on the topic. To enhance the resolution of optimization problems with continuous variables, several works have also considered simplifying an optimization problem by first learning its active constraints [8], [29]. An altogether different paradigm aims to train deep neural networks to produce solutions to optimization problem directly. For example, several works consider end-to-end learning of solutions to combinatorial problems [5], [21], [24], [35]. Other works have shown how to learn solutions to problems with general nonlinear constraints, either by leveraging Lagrangian duality [15], [26], [31], differentiable constraint corrections [10], or reparametrization of the feasible space [23]. Another closely related direction [34] focuses on learning warm-starts to proximal algorithms for quadratic programming.

B. Metric Selection in Proximal Optimization

The potential for accelerating the convergence of a proximal algorithm by optimizing its underlying proximal metric has been theoretically demonstrated in previous works. The authors in [19] derived the optimal choice of metric for ADMM and Douglas-Rachford splitting algorithms on a limited class of problems. Based on this result, they also suggested heuristic methods for selecting an appropriate metric for problems outside of that class. Similar results were shown in [18] for a fast dual forward-backward splitting method. Unfortunately, these theoretical results do not extend to many problems of practical interest, including generic Quadratic Programming (QP) problems. Furthermore, when the optimal metric can be computed, it typically requires solution of a difficult semidefinite program, reducing its practical benefit in accelerating the solution of problems. This paper demonstrates the use of end-to-end machine learning to derive models whose predicted proximal metrics can outperform the heuristic theory-based models of [19] on several QP problems.

III. Preliminaries

Let $\mathcal{S}^n_{++}$ be the set of positive definite matrices. For $M \in \mathcal{S}^n_{++}$ let $\mathbb{R}^n_M$ be the Hilbert space on $\mathbb{R}^n$ with the corresponding inner product and norm defined respectively for all $x, y \in \mathbb{R}^n$ as

$$\langle x, y \rangle_M = x^T M y, \qquad \text{and} \qquad ||x||^2_M = x^T M x \,.$$

We will denote by $\Gamma(\mathbb{R}^n_M)$ the set of functions $f : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$ that are proper, closed and convex, where $\mathbb{R} \cup \{\infty\}$ represents the extended reals. For $f, g \in \Gamma(\mathbb{R}^n_M)$, *Douglas-Rachford* splitting (DR) considers optimization problems of the form

$$\min_{x \in \mathbb{R}^n} f(x) + g(x) \,, \tag{1}$$

and computes optimal solutions by following the iterations

$$y_k = \text{prox}_\gamma g(x_k) \,, \tag{2a}$$
$$z_k = \text{prox}_\gamma f(2y_k - x_k) \,, \tag{2b}$$
$$x_{k+1} = x_k + z_k - y_k \,, \tag{2c}$$

where the $\text{prox}_\gamma f$ is the *proximal operator* defined with respect to the space $\mathbb{R}^n_M$ as

$$\text{prox}_\gamma f(x) = \underset{z \in \mathbb{R}^n}{\arg\min}\, f(z) + \frac{1}{\gamma}||x - z||^2_M \,, \tag{3}$$

for $\gamma > 0$. It can be shown that if a solution of (1) exists then the DR iterations will converge, in particular the sequence $\text{prox}_\gamma g(x_k)$ will converge weakly to a solution and under additional mild assumptions will converge strongly. Various alternative formulations and relaxations of DR exist but in this paper we will primarily restrict consideration to the formulation (2); for proofs and additional details see for example [3].

This formulation naturally gives rise to the question of the selection of a positive-definite matrix M to define a metric in (3) for a given problem to improve convergence of the iterations (2). In [19] Giselsson and Boyd study the optimal metric choice to improve convergence rate for DR applied to the Fenchel dual of (1), which can be shown to be equivalent to employing the alternating direction method of multipliers (ADMM) on the primal problem [19], [16], [12]. In this case, any choice of M other than the identity is equivalent to the use of preconditioning in ADMM. A standard formulation for problems to be solved by ADMM is given by

$$\min_{x \in \mathbb{R}^n, y \in \mathbb{R}^m} \quad f(x) + g(y) \tag{4a}$$
$$\text{subject to:} \quad Ax + By = c \tag{4b}$$

for $f, g \in \Gamma(\mathbb{R}^n_M)$, $A \in \mathbb{R}^{m,n}$, $B \in \mathbb{R}^{m,m}$, and $c \in \mathbb{R}^m$. ADMM applied to the preconditioned primal problem

$$\min_{x \in \mathbb{R}^n, y \in \mathbb{R}^m} \quad f(x) + g(y) \tag{5a}$$
$$\text{subject to:} \quad MAx + MBy = Mc \,, \tag{5b}$$

is equivalent to DR applied to its Fenchel dual when utilizing the metric M. Thus dual DR using metric M can be implemented by the primal ADMM iterations

$$x_{k+1} = \underset{x \in \mathbb{R}^n}{\arg\min}\{f(x) + \gamma 2||M(Ax + By_k - c) + u_k||^2_2\} \,, \tag{6a}$$
$$y_{k+1} = \underset{y}{\arg\min}\{g(y) + \frac{\gamma}{2}||M(Ax_{k+1} + By - c) + u_k||^2_2\} \,, \tag{6b}$$
$$u_{k+1} = u_k + M(Ax_{k+1} + By_{k+1} - c) \,. \tag{6c}$$

In [19] the authors show how to calculate the matrix M which optimizes the convergence rate of ADMM applied to (5), under the additional assumptions that f is strongly convex and smooth, and that A has full row rank. In such cases, an optimal metric M can be modeled as the solution to a related semidefinite programming problem (SDP). However, those requisite assumptions exclude many practical

optimization problems, including general-form quadratic programming (QP) problems. For QP forms outside the scope of these assumptions, the authors of [19] suggest heuristic models of metric selection.

Significant work has been done to improve ADMM convergence using preconditioning, for instance in [17], [7]. As pointed out in [19], in the case of QP problems these methods ultimately amount to reconditioning the quadratic objective function. The same is true of the heuristic method they propose, which equates to selecting the optimal metric for a related *unconstrained* QP. In this paper, we explore the potential for empirically learning metrics to enhance convergence of proximal algorithms on QP problems which do not satisfy the assumptions required for metric optimization presented in [19]. We investigate solution of QP problems using learned metrics with DR applied to both the primal and dual problems, implementing ADMM for solution of the dual as given in (6).

IV. Learning Metrics to Accelerate Quadratic Programming

The proposed system for metric learning in proximal optimization leverages a reformulation of general QP problems which renders the metrics easier to learn. In this section, we first introduce the problem reformulation before describing details of the end-to-end learning approach. In brief, a neural network model is trained to predict positive definite matrices M as a function of the parameters which define an optimization problem instance. Solution error after a fixed number of iterations (2) or (6) is treated as a loss function and minimized, by backpropagation through the solver iterations in stochastic gradient descent training. While the system is general and can in principle be applied to any problem of the form (1), the scope of this paper is limited to demonstration on QP problems.

A. Problem Reformulation

Let $Q \in \mathbb{R}^{n,n}$ be a positive semi-definite matrix, $q \in \mathbb{R}^n$, $L \in \mathbb{R}^{m,n}$, $b \in \mathbb{R}^m$, $W \in \mathbb{R}^{k,k}$, and $c \in \mathbb{R}^k$. We consider QP problems of the form

$$\underset{x\in\mathbb{R}^n}{\arg\min} \ \frac{1}{2}x^TQx + q^Tx \tag{7a}$$

$$\textit{s.t.} \ \ Lx = b \tag{7b}$$

$$Wx + c \leq 0. \tag{7c}$$

For implementation with both primal DR and ADMM we introduce slack variables $s \in \mathbb{R}^m$ and reformulate the problem as

$$\underset{z\in\mathbb{R}^{n+k}}{\arg\min} \ \frac{1}{2}z^TI_x^TQI_xz + q^TI_xz \tag{8a}$$

$$\textit{s.t.} \ \ Rz + r = 0 \tag{8b}$$

$$I_sz \geq 0, \tag{8c}$$

where I^n is the $n \times n$ identity matrix, and

$$z = \begin{bmatrix} x \\ s \end{bmatrix}, \quad R = \begin{bmatrix} L & 0 \\ W & I^k \end{bmatrix}, \quad r = \begin{bmatrix} -b \\ c \end{bmatrix},$$
$$I_s = \begin{bmatrix} 0 & 0 \\ 0 & I^k \end{bmatrix}, \quad I_x = \begin{bmatrix} I^n & 0 \\ 0 & 0 \end{bmatrix}.$$

Note that even if the inequalities (7c) represent simple box constraints on the variables x (for example $x < 0$) we still introduce corresponding slack variables. This is done to construct a splitting for both primal DR and ADMM with the intent to increase the impact M can have on convergence, as will be described in the next section.

1) QP Splitting: For implementation of both primal DR and ADMM on problem (8) we use the splitting

$$f(z) = z^TI_x^TQI_xz + q^TI_xz + i_{\{z\in\mathbb{R}^{n+k} \ : \ Rz+r=0\}}(z) \tag{9a}$$

$$g(z) = i_{\{z\in\mathbb{R}^{n+k} \ : \ I_sz\geq 0\}}(z) \tag{9b}$$

where for a set $\mathcal{S}$ we define the indicator function on $\mathcal{S}$ to be

$$i_S(x) = \begin{cases} 0 & x \in S \\ \infty & x \notin S \end{cases}.$$

With this splitting we implement the primal DR iteration as given in (2), and implement ADMM as in (6), with $A = I$, $B = -I$, and $c = 0$. Minimization steps with respect to f can be accomplished with for example the corresponding Karush-Kuhn-Tucker conditions. Minimization steps with respect to g for both algorithms equate to projections onto the positive orthant. Slack variables are initialized at zero throughout.

For both ADMM and DR iterations using the splitting (9), the proximal operators as given in (2b) and (6a) equate to projections onto the relaxed constraint set (8b) with respect to the underlying metric. With the slack variables for each inequality constraint initialized at zero, a relatively large corresponding weight in the metric matrix M will bias the non-Euclidean projection to maintain those slacks near zero. Hence if M has relatively large weights for just the slacks corresponding to the active constraints of a problem, the projection can approximate projection onto the active set. Indeed, the learned metrics exhibit this expected behavior as illustrated in Figure 2.

B. End-to-End Learning Framework

As suggested by the results of Section (V), the optimal metric for solving of an instance of (8) can be closely related to the active constraints at its optimal solution. Thus, it is expected that learning the optimal metrics for solving a class of problems may be nearly as difficult as learning their optimal solutions. As is common in prior works on learning to solve optimization problems, the metric learning problem is formulated relative to a *parametric* optimization problem

$$x^\star(p) = \underset{x\in\mathbb{R}^n}{\arg\min} \ f_p(x) + g_p(x), \tag{10}$$

and we learn to predict metrics for parametric problem instances within a limited distribution. In the QP problem

(7), this corresponds to the elements Q, q, L, b, w, and c each being potential functions of p. The metric M which best solves problem (10) is then learned as a function of the problem's parameters $p \in \mathbb{R}^v$. This learned function takes the form of a neural network $\mathcal{N}_\omega : \mathbb{R}^v \to \mathcal{S}^n_{++}$ with weights ω, so that $M = \mathcal{N}_\omega(p)$. It is trained over a distribution of problem parameters $p \sim \mathcal{P}$, for which a finite dataset of instances $\{p_i\}_{i\in\mathcal{T}}$ are drawn. A target dataset $\{x^*(p_i)\}_{i\in\mathcal{T}}$ contains the corresponding optimal solutions, as per (10).

To define a loss function for training $\mathcal{N}_\omega$ on these data, first define the following function. Let $\mathcal{D}_k : \mathbb{R}^v \times \mathcal{S}^n_{++} \times \mathbb{R}^n \to \mathbb{R}^n$ denote the application of k iterations of DR or ADMM, on problem (10) with parameters p using metric M, starting from initial variable values x_0. It yields a solution estimate x_k; that is, $\mathcal{D}_k(p, M, x_0) = x_k$. The metric prediction model $\mathcal{N}_\omega$ is then trained to minimize the overall loss function

$$\min_{\omega} \frac{1}{|\mathcal{T}|} \sum_{i\in\mathcal{T}} ||\mathcal{D}_k(p_i, \mathcal{N}_\omega(p_i), \mathcal{E}_\theta(p_i)) - x^*(p_i)||^2 \qquad (11)$$

by stochastic gradient descent. This requires backpropagation of gradients through the solver iterations which constitute $\mathcal{D}_k$. In this work, backpropogation is performed by automatic differentiation in PyTorch [32].

In equation (11), the function $\mathcal{E}_\theta$ is an oracle which returns a starting point x_0 for any parameter vector p. As part of an overall mechanism for producing fast solutions to (7), it is a neural network trained to produce direct estimates of the optimal solution to (10) by mean square error regression:

$$\min_{\theta} \frac{1}{|\mathcal{T}|} \sum_{i\in\mathcal{T}} ||\mathcal{E}_\theta(p_i) - x^*(p_i)||^2 . \qquad (12)$$

C. Metric Representation

Finally, we describe the manner of representation used to predict the metrics M via the model $M = \mathcal{N}_\omega(p)$. As a neural network, $\mathcal{N}_\omega$ produces a vector of values $m \in \mathbb{R}^n$ which is then scaled between predefined upper and lower bounds $[m_{\min}, m_{\max}]$. Finally, a scalar parameter $\rho \in \mathbb{R}$ is predicted and also scaled to fit within predefined bounds $[\rho_{\min}, \rho_{\max}]$. The final metric is constructed as the diagonal matrix $M = diag(\rho \cdot m)$.

V. Numerical Results

For illustrative purposes, this section begins with a reductive two-dimensional problem on which some effects of metric learning are most easily observed. Then, we demonstrate the effect of metric learning on convergence for larger examples consisting of a portfolio optimization problem, and a model predictive control problem.

In the following experiments, predictive models $\mathcal{N}_\omega$ and $\mathcal{E}_\theta$ are fully connected neural networks with rectified linear unit (ReLU) activation functions. The values $\rho_{\min}, \rho_{\max}, m_{\min}, m_{\max}$ can be treated as hyperparameters; in practice, it is found that effective metrics can be learned by searching over $\rho_{\max}$ while the others remain fixed. All numerical test cases in this work are implemented using NeuroMANCER, an open source differentiable programming library built on top of Pytorch [11].

A. Active Set Prediction

This section illustrates how metric learning correlates with active set prediction in inequality-constrained problems, by assigning higher metric weights to the coordinates which correspond to slack variables on the problem's active constraints. As an illustrative example, we consider a simple QP problem:

$$\min_{x,y} \; x^2 + y^2 \qquad (13a)$$

subject to:

$$-x - y + p_1 \leq 0 \qquad (13b)$$
$$x + y - p_1 - 1 \leq 0 \qquad (13c)$$
$$x - y + p_2 - 1 \leq 0 \qquad (13d)$$
$$-x + y - p_2 \leq 0 \qquad (13e)$$

for parameters $p_1, p_2 \in [-2, 2]$. The constraint set defines a box which is translated around the origin according to the parameter choices as shown in Figure 2. After assigning slack variables s, the constraints become

$$-x - y + p_1 + s_1 = 0 \qquad (14a)$$
$$x + y - p_1 - 1 + s_2 = 0 \qquad (14b)$$
$$x - y + p_2 - 1 + s_3 = 0 \qquad (14c)$$
$$-x + y - p_2 + s_4 = 0 \qquad (14d)$$
$$s_1,\ s_2,\ s_3,\ s_4 \geq 0 \qquad (14e)$$

We train both ADMM and DR metrics over the parameter space. The neural network map $\mathcal{N}_\omega$ returns diagonal metrics with diagonals of the form

$$diag([w_y, w_x, w_1, w_2, w_3, w_4])$$

where the weights w_x and w_y correspond to the primal variables, and the weights $\{w_1, w_2, w_3, w_4\}$ correspond to the slack variables for each of the constraints.

a) Settings: The initial prediction model $\mathcal{E}_\theta$ uses hidden dimension 80. For $\mathcal{N}_\omega$ we use hidden dimension 20, with bounds $\rho_{\min} = 0.05$, $\rho_{\max} = 1.0$, $m_{\max} = 5.0$, and $m_{\min} = 0.2$. We sample 2000 parameters uniformly at random to construct a training set, and an additional 2000 parameters uniformly at random for a test set. The initial solution estimator $\mathcal{E}_\theta$ was trained for 200 epochs at learning rate of 0.001. Both ADMM and DR were run for 10 steps during training, and $\mathcal{N}_\omega$ was trained for 100 epochs at a learning rate of 0.001 for both.

b) Results: As shown in Figure 1, DR with a trained metric converges the fastest, while ADMM using the heuristic metric as presented in [19] converges most slowly. Computation of the heuristic metric effectively ignores the inequality constraints, and computes a metric that achieves the fastest convergence with respect to the objective, were no constraints present this metric would achieve the best possible convergence. However, the results show that in the presence of constraints it can be detrimental.

Conversely, the learned metrics appear to achieve faster convergence by incorporating information about the active

constraints. Figure 2 and Figure 3 highlight the close correspondence between the metric weights corresponding to slack variables and whether the associated constraints are active at the optimum for a problem.

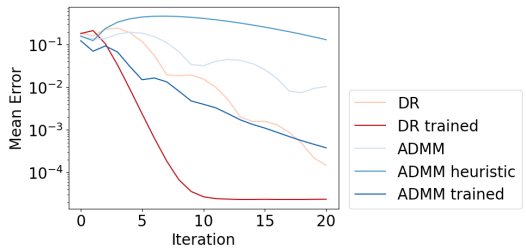


Fig. 1: Comparison of convergence of DR and ADMM using trained metrics versus not, as well as comparison to use of a heuristic metric choice. Reported values are the mean error at each iteration on 2000 test set problems.

B. Portfolio Optimization Problem

This experiment models the optimal allocation of assets in an investment portfolio as a quadratic programming problem. Given n investment assets, their future price differentials $p \in \mathbb{R}^n$ are treated as parameters in the following QP:

$$x^*(p) = \arg\min_x \; x^T \Sigma x - p^T x \tag{15a}$$

$$\text{subject to:} \quad 1^T x = 1 \tag{15b}$$

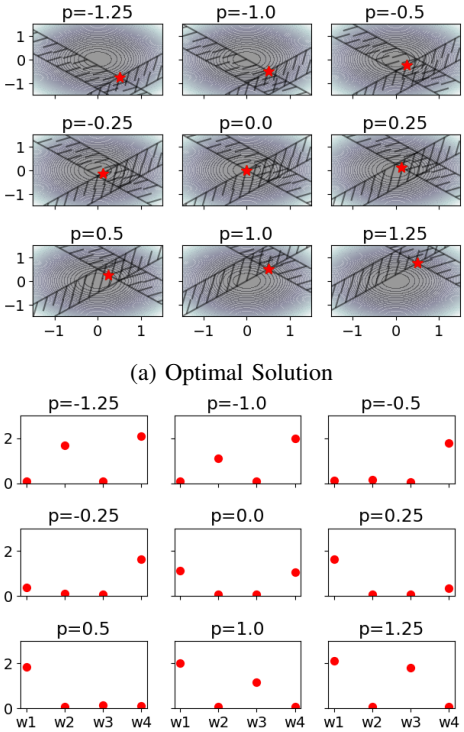
$$x \geq 0, \tag{15c}$$

where Σ represents a constant covariance matrix. The objective (15a) balances maximization of future profit with minimization of price covariance as a measure of risk. Constraints (15b,15c) define a valid proportional allocation.

a) Settings: Data on price action per asset are collected from the Nasdaq online database [30]. A training dataset of 5000 observations for the future price differential p are generated by adding Gaussian random noise to this data, plus an additional 500 each for the validation and testing sets. A 5-layer neural network with hidden layer size n is used to predict the elements of a diagonal metric matrix M as a function of p. The following parameters are fixed: $m_{\min} = 0.01$, $m_{\max} = 1.0$, $\rho_{\min} = 0.01$. A search over the upper bound $\rho_{\max} \in \{1.0, 5.0, 10.0, 50.0, 100.0, 500.0\}$ shows that the best results occur for $\rho_{\max} = 100.0$ but remain similar for higher values of $\rho_{\max}$.

b) Results: Figure 4 illustrates convergence of the DR and ADMM algorithms in solving (15), under metric prediction trained with k iterations in the loop, for $k \in \{5, 10, 15, 20, 25, 30\}$. At test time, 100 iterations of DR and 150 iterations ADMM are applied regardless of k. The plotted values represent mean relative solution error in the $L2$ norm. Dotted black curves correspond to a baseline in which the standard Euclidean metric is used.

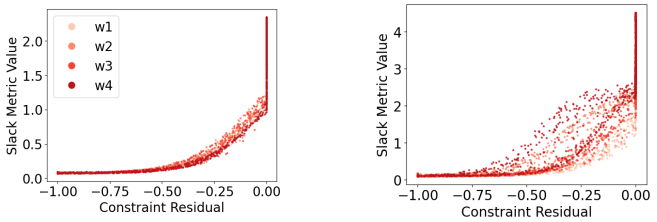
The following observations apply to both DR (at left) and ADMM (at right). The metric prediction model which is trained for k iterations always attains the best accuracy at exactly k iterations. Additionally, the models trained with



(a) Optimal Solution

(b) Metric weight for each slack variable

Fig. 2: Plot (a) shows the optimal solution of (13) for parameter choice $p_1 = p_2 = p$, for values of p ranging from -1.25 to 1.25. As the feasible set is translated around the origin the optimal solution marked by a red star can be seen to trace along the constraints. Correspondingly, plot (b) shows that the learned DR metric weights corresponding to the slack variables for the inequality constraints are near zero when the constraint is not active and larger when the constraint is active, showing a clear correspondence between the active set and metric weights on slack variables.



(a) DR

(b) ADMM

Fig. 3: Plots of the learned metric weight for slack variables and the corresponding constraint residual at the optimum on two thousand test problems for metrics learned using DR and ADMM. As constraint residuals becomes zero the constraints become active and the metric weight for the corresponding slack increases.

more iterations k perform better as more iterations are performed at test time. This implies that an accelerated DR or ADMM model intended for exactly k iterations should be trained using k iterations, while a model intended for iteration until convergence should train using large values of k.

Note additionally that in this particular experiment, training for small k comes at a cost of long-term convergence in the case of DR. In ADMM, models trained with larger k generally perform equally or better than with smaller k. An exception is observed for $k = 30$ in ADMM, in which error is minimized at iteration 30 at the cost of higher error in both earlier and later iterations. In all but one case, the learned metrics outperform the Euclidean metric at test time.

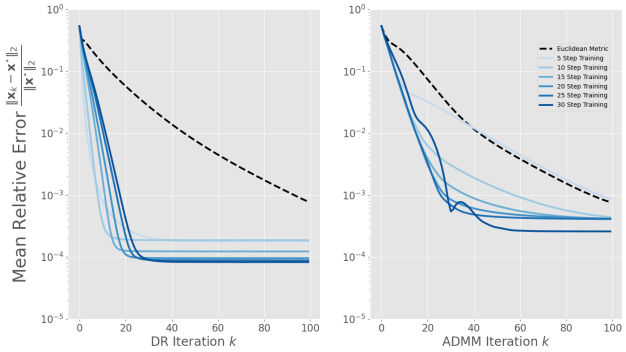


Fig. 4: Results of training proximal metrics for DR and ADMM on Portfolio Optimization, to minimize error at increments of 5 iterations.

c) Comparison with Heuristic Metric Selection: In order to compare the proposed metric learning for ADMM against the theoretically prescribed heuristic metric choice given in [19] we test the metric learning experiment on problem (15) with a reduced size of $n = 20$ assets. Problem size is reduced to allow a heuristic metric to be calculated efficiently.

To illustrate the effect of active constraints on the viability of the heuristic metric, results for ADMM are compared on two variants of problem (15). The first is as given in (15), and the second replaces the equality constraint (15b) with a scaled asset allocation budget $1^T x = 10$. This increase in budget has the effect of reducing the percentage of active constraints over problem parameters. Figure 5 shows the results due to each allocation budget, at left and right respectively. In the larger budget case few constraints are active on average over the test set. This approximates a problem setting with no constraints where the heuristic metric would be optimal, and it can be seen to improve convergence in comparison to the Euclidean metric. Further, the learned optimal metrics perform similarly when trained for more than $k = 5$ iterations. On the other hand, the smaller budget leads to more active constraints and the heuristic metric performs poorly. Conversely, the trained metric achieves a greater improvement in convergence rate with respect to the Euclidean metric than in the high budget case, highlighting the capacity for improvement by incorporating information about active constraints.

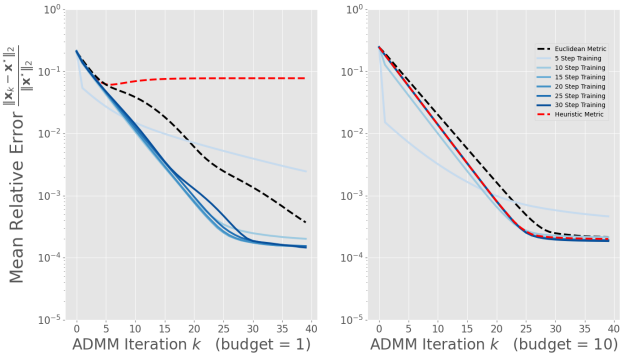


Fig. 5: Comparison of trained and heuristic metrics for ADMM for Portfolio Optimization. The left plot presents a case where problem constraints are routinely active over problem parameters, while the right shows a case in which problem constraints are more rarely active.

C. Quadcopter Control

We also test metric learning on a standard reference tracking model predictive control problem implemented for a linear discrete time dynamical model of a quadcopter. Let the dynamics be defined by $A \in \mathbb{R}^{n,n}$, $B \in \mathbb{R}^{n,m}$, and the cost function be defined by positive semi-definite matrices $Q \in \mathbb{R}^{n,n}$ and $R \in \mathbb{R}^{m,m}$. Here we take the parameter $p \in \mathbb{R}^n$ to be the initial state of the system and solve

$$u^*(p), x^*(p) = \arg\min_{x,u} \sum_{k=0}^{N} (x_{k+1} - r)^T Q (x_{k+1} - r) + u_k^T R u_k \tag{16a}$$

subject to:

$$x_1 = Ap + Bu_0\,, \tag{16b}$$
$$x_{k+1} = Ax_k + Bu_k\,, \tag{16c}$$
$$u_a \leq u_k \leq u_b\,, \tag{16d}$$
$$x_a \leq x_k \leq x_b\,, \tag{16e}$$
$$\forall k \in \{1, 2, \dots N\}\,. \tag{16f}$$

The optimal solution produces control actions $u_k \in \mathbb{R}^m$ that drive the state of the system $x_k \in \mathbb{R}^n$ from a given initial state p towards the reference point $r \in \mathbb{R}^n$ over a finite number of time steps $k \in \{1, 2, \dots N\}$. The control actions must also keep the state within the bounds $x_a, x_b \in \mathbb{R}^n$ while staying within the control bounds $u_a, u_b \in \mathbb{R}^m$. In practice the problem (16) is solved iteratively with the control implemented for just the initial time step, then the problem is re-solved from the new system state at the next time step. Thus the problem can be understood to be parameterized by the initial state and reference point.

Here we assume a fixed reference point at the origin and sample over a range of initial system states. The matrices describing the quadcopter model dynamics and objectives as well as state and control bounds are given in the Appendix VI-A, they were taken from a set of benchmark problems curated in [27], with data from the repository [13].

a) Settings: The quadcopter model has a 12 dimensional state and 4 dimensional control input. Only the first two state variable are constrained, and are restricted to the interval $[-\pi/6\ ,\ \pi/6]$. To generate training data we take the reference point r to be the origin and we generate initial states p uniformly at random with the first two variables sampled from $[-\pi/6\ ,\ \pi/6]$ and the rest from $[-0.8, 0.8]$. This choice was made such that generated problems were feasible, and state constraints were routinely active at solutions. We solve the problem (16) over a 10 step horizon, resulting in $n = 304$ variables with 132 inequality constraints, and 132 equality constraints. Diagonal elements of a metric matrix M are predicted on a per-instance basis using a 5-layer ReLU network of hidden layer size 400. As in Section V-B, the parameter choices: $m_{\min} = 0.01$, $m_{\max} = 1.0$, $\rho_{\min} = 0.01$ are fixed. A search over the upper bound $\rho_{\max} \in \{1.0, 5.0, 10.0, 50.0, 100.0, 500.0\}$ shows that the best results occur for $\rho_{\max} = 50.0$ and remain similar for higher values of $\rho_{\max}$.

b) Results: Convergence of both DR and ADMM due to the various trained metric prediction models are illustrated in Figure 6. Prediction models are trained using k steps of each algorithm for $\{5, 10, 15, 20, 25, 30, 35, 40\}$. Solution error over the full horizon needed for convergence is not shown, to make visible the effects of training up to the first 40 iterations. This is consistent with the intended application in real-time optimization, which demands solutions within stringent time constraints.

With regards to the effect of k, similar observations apply as in the portfolio optimization experiments. Models trained to minimize error at iteration k consistently perform best after exactly k iterations. Meanwhile, training with larger k generally benefits long-term convergence. Note that nearly all trained models reach a relative error of $1e-2$ in a fraction of the iterations required by the standard variant with a Euclidean metric.

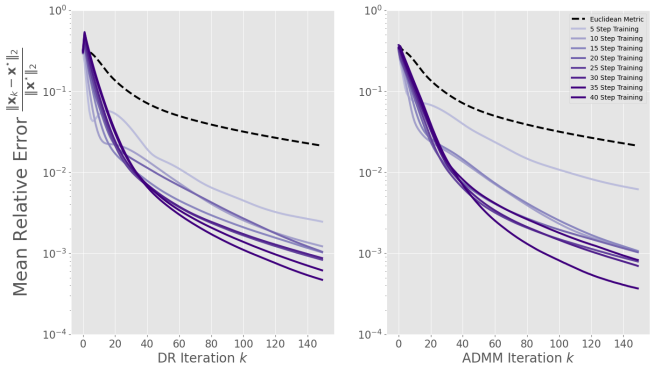


Fig. 6: Results of training proximal metrics for DR and ADMM on Quadcopter Control, to minimize error at increments of 5 iterations.

VI. CONCLUSION

Metric learning as presented here for parametric QP problems can consistently result in orders of magnitude improvements in solution accuracy at low a number of iterations. The most benefit is observed in problem settings with a significant number of inequality constraints relative to the problem size that are routinely active over parameters of interest. Notably, this is exactly the case in which the theoretically prescribed heuristic metrics are not guaranteed to be optimal. Future work is needed to understand the capacity for metric learning to reduce solution time on large-scale problems. By relying a problem reformulation with slack variables, the total number of variables is expanded resulting in a larger overall problem size. On the other hand, it allows for metric learning to potentially significantly reduce the number of iterations required to achieve a given accuracy as seen here. Because it is independent of other strategies for learning to accelerate optimization, it may have significant potential to be combined with previously proposed techniques. Combining the proposed metric learning with nonoverlapping strategies such as prediction of solution warmstarts may further reduce overall solution times.

APPENDIX

A. Quadcopter Model Details

$$
\begin{aligned}
Q &= diag([0, 0, 10, 10, 10, 10, 0, 0, 0, 5, 5, 5], \\
R &= diag([0.1, 0.1, 0.1, 0.1]), \\
u_a &= [9.6, 9.6, 9.6, 9.6] \ - \ 10.5916, \\
u_b &= \ [13., 13., 13., 13] \ - \ 10.5916
\end{aligned}
$$

$$
A = \begin{pmatrix}
1.0 & 0 & 0 & 0 & 0 & 0 & 0.1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1.0 & 0 & 0 & 0 & 0 & 0 & 0.1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1.0 & 0 & 0 & 0 & 0 & 0 & 0.1 & 0 & 0 & 0 \\
0.0488 & 0 & 0 & 1.0 & 0 & 0 & 0.0016 & 0 & 0 & 0.0992 & 0 & 0 \\
0 & -0.0488 & 0 & 0 & 1.0 & 0 & 0 & -0.0016 & 0 & 0 & 0.0992 & 0 \\
0 & 0 & 0 & 0 & 0 & 1.0 & 0 & 0 & 0 & 0 & 0 & 0.0992 \\
0 & 0 & 0 & 0 & 0 & 0 & 1.0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1.0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1.0 & 0 & 0 & 0 \\
0.9734 & 0 & 0 & 0 & 0 & 0 & 0.0488 & 0 & 0 & 0.9846 & 0 & 0 \\
0 & -0.9734 & 0 & 0 & 0 & 0 & 0 & -0.0488 & 0 & 0 & 0.9846 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.9846
\end{pmatrix}
$$

$$
B = \begin{pmatrix}
0 & -0.0726 & 0 & 0.0726 \\
-0.0726 & 0 & 0.0726 & 0 \\
-0.0152 & 0.0152 & -0.0152 & 0.0152 \\
-0 & -0.0006 & -0 & 0.0006 \\
0.0006 & 0 & -0.0006 & 0.0000 \\
0.0106 & 0.0106 & 0.0106 & 0.0106 \\
0 & -1.4512 & 0 & 1.4512 \\
-1.4512 & 0 & 1.4512 & 0 \\
-0.3049 & 0.3049 & -0.3049 & 0.3049 \\
-0 & -0.0236 & 0 & 0.0236 \\
0.0236 & 0 & -0.0236 & 0 \\
0.2107 & 0.2107 & 0.2107 & 0.2107
\end{pmatrix}.
$$

ACKNOWLEDGMENT

This research was supported by the AT Scale and Data Model Convergence (DMC) initiatives via the Laboratory Directed Research and Development (LDRD) investments at Pacific Northwest National Laboratory (PNNL). PNNL is a national laboratory operated for the U.S. Department of Energy (DOE) by Battelle Memorial Institute under Contract No. DE-AC05-76RL0-1830.

REFERENCES

[1] B. Amos *et al.*, "Tutorial on amortized optimization," *Foundations and Trends® in Machine Learning*, vol. 16, no. 5, pp. 592–732, 2023.

[2] M.-F. Balcan, T. Dick, T. Sandholm, and E. Vitercik, "Learning to branch," in *International conference on machine learning*. PMLR, 2018, pp. 344–353.

[3] H. H. Bauschke and P. L. Combettes, *Convex Analysis and Monotone Operator Theory in Hilbert Spaces*. New York, New York, USA: Springer, 2019.

[4] A. Beck, *First-order methods in optimization*. SIAM, 2017.

[5] I. Bello, H. Pham, Q. V. Le, M. Norouzi, and S. Bengio, "Neural combinatorial optimization with reinforcement learning," *arXiv:1611.09940*, 2017.

[6] Y. Bengio, A. Lodi, and A. Prouvost, "Machine learning for combinatorial optimization: a methodological tour horizon," *European Journal of Operational Research*, 2020.

[7] M. Benzi, "Preconditioning techniques for large linear systems: A survey," *Journal of Computational Physics*, vol. 182, no. 2, pp. 418–477, 2002. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0021999102971767

[8] D. Bertsimas and B. Stellato, "The voice of optimization," *Machine Learning*, vol. 110, no. 2, pp. 249–277, 2021.

[9] A. Deza and E. B. Khalil, "Machine learning for cutting planes in integer programming: A survey," *arXiv preprint arXiv:2302.09166*, 2023.

[10] P. L. Donti, D. Rolnick, and J. Z. Kolter, "Dc3: A learning method for optimization with hard constraints," *arXiv preprint arXiv:2104.12225*, 2021.

[11] J. Drgona, A. Tuor, J. Koch, M. Shapiro, and D. Vrabie, "NeuroMANCER: Neural Modules with Adaptive Nonlinear Constraints and Efficient Regularizations," 2023. [Online]. Available: https://github.com/pnnl/neuromancer

[12] J. Eckstein, "Splitting methods for monotone operators with applications to parallel optimization," PhD thesis, MIT, 1989.

[13] H. Ferreau, "mpcbenchmarking," https://github.com/ferreau/mpcBenchmarking, 2015.

[14] F. Fioretto, P. V. Hentenryck, T. W. Mak, C. Tran, F. Baldo, and M. Lombardi, "Lagrangian duality for constrained deep learning," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2020, pp. 118–135.

[15] F. Fioretto, T. W. Mak, and P. Van Hentenryck, "Predicting ac optimal power flows: Combining deep learning and lagrangian dual methods," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, 2020, pp. 630–637.

[16] D. Gabay, "Applications of the method of multipliers to variational inequalities," in *Augmented Lagrangian Methods: Applications to the Solution of Boundary-Value Problems*. Amsterdam, The Netherlands: North Holland, 1983.

[17] E. Ghadimi, A. Teixeira, I. Shames, and M. Johansson, "Optimal parameter selection for the alternating direction method of multipliers (admm): Quadratic problems," *IEEE Transactions on Automatic Control*, vol. 60, no. 3, pp. 644–658, 2015.

[18] P. Giselsson and S. Boyd, "Metric selection in fast dual forward–backward splitting," *Automatica*, vol. 62, pp. 1–10, 2015.

[19] ——, "Linear convergence and metric selection for douglas-rachford splitting and admm," *IEEE Transactions on Automatic Control*, vol. 62, no. 2, pp. 532–544, 2017.

[20] P. Gupta, M. Gasse, E. Khalil, P. Mudigonda, A. Lodi, and Y. Bengio, "Hybrid models for learning to branch," *Advances in neural information processing systems*, vol. 33, pp. 18 087–18 097, 2020.

[21] E. Khalil, H. Dai, Y. Zhang, B. Dilkina, and L. Song, "Learning combinatorial optimization algorithms over graphs," in *NIPS*, 2017, pp. 6348–6358.

[22] E. Khalil, P. Le Bodic, L. Song, G. Nemhauser, and B. Dilkina, "Learning to branch in mixed integer programming," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 30, 2016.

[23] A. V. Konstantinov and L. V. Utkin, "A new computationally simple approach for implementing neural networks with output hard constraints," in *Doklady Mathematics*. Springer, 2024, pp. 1–9.

[24] W. Kool, H. Van Hoof, and M. Welling, "Attention, learn to solve routing problems!" *arXiv preprint arXiv:1803.08475*, 2018.

[25] J. Kotary, F. Fioretto, and P. Van Hentenryck, "Learning hard optimization problems: A data generation perspective," *Advances in Neural Information Processing Systems*, vol. 34, pp. 24 981–24 992, 2021.

[26] ——, "Fast approximations for job shop scheduling: A lagrangian dual deep learning method," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, 2022, pp. 7239–7246.

[27] D. Kouzoupis, A. Zanelli, H. Peyrl, and H. J. Ferreau, "Towards proper assessment of qp algorithms for embedded model predictive control," in *2015 European Control Conference (ECC)*, 2015, pp. 2609–2616.

[28] A. Lodi and G. Zarpellon, "On learning and branching: a survey," *Top*, vol. 25, pp. 207–236, 2017.

[29] S. Misra, L. Roald, and Y. Ng, "Learning for constrained optimization: Identifying optimal active constraint sets," *INFORMS Journal on Computing*, vol. 34, no. 1, pp. 463–480, 2022.

[30] Nasdaq, "Nasdaq end of day us stock prices," https://data.nasdaq.com/databases/EOD/documentation, 2022, accessed: 2023-08-15.

[31] S. Park and P. Van Hentenryck, "Self-supervised primal-dual learning for constrained optimization," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, 2023, pp. 4052–4060.

[32] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," 2017.

[33] M. B. Paulus, G. Zarpellon, A. Krause, L. Charlin, and C. Maddison, "Learning to cut by looking ahead: Cutting plane selection via imitation learning," in *International conference on machine learning*. PMLR, 2022, pp. 17 584–17 600.

[34] R. Sambharya, G. Hall, B. Amos, and B. Stellato, "End-to-end learning to warm-start for real-time quadratic optimization," in *Learning for Dynamics and Control Conference*. PMLR, 2023, pp. 220–234.

[35] O. Vinyals, M. Fortunato, and N. Jaitly, "Pointer networks," in *Advances in Neural Information Processing Systems (NeurIPS*, 2015, pp. 2692–2700.