A Surrogate Objective Framework for Prediction+Programming with Soft Constraints

Anonymous Author(s)
Affiliation
Address
email

Abstract

Prediction+programming is a common real-world paradigm where problem parameters are forecast before solving the programming task. However, the criteria by which the prediction model is trained are often inconsistent with the goal of the downstream programming problem. Recently, decision-focused prediction approaches, such as SPO+ and direct optimization, have been proposed to fill this gap. However, they cannot well handle the soft constraints with the max operator required in many real-world objectives. This paper proposes a novel analytically differentiable surrogate objective framework for real-world linear and semi-definite negative quadratic programming problems with soft constraints. This framework gives the theoretical bounds on constraints' multipliers, and derives the closed-form solution with respect to predictive parameters and thus gradients for any variable in the problem. We evaluate our method in three applications extended with soft constraints: synthetic linear programming, portfolio optimization, and resource provisioning, demonstrating that our method outperforms traditional two-staged methods and other decision-focused approaches.

1 Introduction

Mathematical programming (a.k.a. mathematical optimization), e.g., linear and quadratic programming, has been widely applied in decision-making processes, such as resource scheduling [1], goods production planning [2], portfolio optimization [3], and power scheduling [4]. In practice, problem parameters (e.g., goods demands, and equity returns) are often contextual and predicted by models with observed features (e.g., history time series). With the popularity of machine learning techniques and increasing available data, prediction+programming has become a normal paradigm [5].

Prediction becomes critical to the performance of the full prediction+programming workflow since modern programming solvers (e.g., Gurobi [6] and CPLEX [7]) can already efficiently find optimal solutions for most large scale programming problems. Traditionally, prediction is treated separately as a general supervised learning problem and learned through minimizing a general loss function (e.g., mean squared error for regression). However, studies have shown that minimization of the fitting errors does not necessarily lead to better final decision performance [8, 9, 10, 11].

Recently, efforts on using programming objective to guide the learning of prediction models have been made. For linear objectives, the 'Smart Predict then Optimize' ([10]) proposes the SPO+ loss function to measure the prediction errors against programming objectives, while direct optimization ([12]) updates the prediction model's parameters by perturbation. For quadratic objectives, OptNet [13, 14] implements the programming as an implicit layer whose gradients can be computed by differentiating the KKT conditions and then back propagates to prediction neural network. CVXPY [15] uses similar technologies with OptNet but extends to more general cases of convex programming. However, all the above state-of-the-art approaches do not contain the soft constraint in their objectives or directly

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

handle them, which is however often required in practice. We consider linear 'soft-constraints', a penalty in the form of $\max(z, 0)$, where $z = Cx - d$ is a projection of decision variable $x \in \mathbb{R}^n$ and context variables $C \in \mathbb{R}^{m \times n}, d \in \mathbb{R}^m$. Such soft constraints could be, for example, the waste of provisioned resources over demands or the extra tax paid when violating regulatory rules. Unfortunately, the $\max(\cdot, 0)$ operator is not differentiable and thus cannot be directly handled by these existing approaches. To differentiate soft constraints is a primary motivation of this paper.

In this paper, we derive a surrogate objective framework for a broad set of real-world linear and quadratic programming problems with generalized soft constraints and implement decision-focused differentiable predictions. The framework consists of three steps: 1) rewriting all hard constraints into piece-wise linear soft constraints with a bounded penalty multiplier; 2) using a differentiable element-wise surrogate to substitute the piece-wise objective, and solving the original function numerically to decide which segment the optimal point is on; 3) analytically solving the local surrogate and obtaining the gradient; the gradient is identical to that of piecewise surrogate since the surrogate is convex/concave and the optimal point is unique.

Our main contributions are summarized as follows. First, we propose a differentiable surrogate objective function that incorporates both soft and hard constraints. As the foundation of our methodology, in Section 3, we prove that, with reasonable assumptions generally satisfied in the real world, for linear and semi-definite negative quadratic programming problems, the constraints can be transformed into soft constraints; then we propose an analytically differentiable surrogate function for the soft constraints $\max(\cdot, 0)$. Second, we present the derived analytical and closed-form solutions for three general programming problems extended with soft constraints in Section 4 – linear programming with soft constraints, quadratic programming with soft constraints, and asymmetric soft constraint minimization. Our method bypasses the KKT-equation solving process and makes the calculation of gradients straightforward for predicting context variables in any part of the problem. Finally, we apply with theoretical derivations and evaluate our approach in three scenarios, including synthetic linear programming, portfolio optimization, and resource provisioning in Section 4, empirically demonstrate that our method outperforms two-stage and other predict+programming approaches.

2 Preliminaries

2.1 Real-world Programming Problems with Soft Constraints

Our target is to solve a broad set of real-world programming problems extended with a $max(z, 0)$ term in their objectives where z depends on decision variables and predicted context parameters. In practice, $max(z, 0)$ is very common; for example, it may model overhead of under-provisioning, over-provisioning of goods, and penalty of soft regulation violations in investment portfolios.

The general problem formulation is

$$\max_x g(\theta, x) - \alpha^T \max(z, 0),\ z = Cx - d,\ \text{s.t. } Ax \leq b, Bx = c,\ x \geq 0 \tag{1}$$

where g is a utility function, $x \in \mathbb{R}^n$ the decision variable, and $\theta \in \mathbb{R}^n$ the predicted parameters.

We have two key observations summarized from real-world problems, which form assumptions of following derivations in this paper. First, constraint parameters usually represent weights, quantities or their thresholds in practice, and thus are always non-negative. In other words, $A \in \mathbb{R}^{m_1 \times n} \geq 0$, $b \in \mathbb{R}^{m_1} \geq 0$, $B \in \mathbb{R}^{m_2 \times n} \geq 0$,and $c \in \mathbb{R}^{m_2} \geq 0$ hold. Second, soft constraints are common in the form of $z = Cx - d$, where $z \in \mathbb{R}^{m_3}$, $C \in \mathbb{R}^{m_3 \times n}$, and $d \in \mathbb{R}^{m_3}$. Soft constraints have rich meanings in reality, which may describe the penalty of goods under-provisioning or over-provisioning, a vanish in marginal profits, or running out of current materials.

Now we look into three representative instances extracted from real-world applications.

Linear programming with soft constraints, where $g(x, \theta) = \theta^T x$. The problem formulation is

$$\max_x \theta^T x - \alpha^T \max(Cx - d, 0), \text{s.t. } Ax \leq b, Bx = c, x \geq 0 \tag{2}$$

where $\alpha \geq 0$. Consider the application of logistics where θ represents goods' prices, $\{A, B, C\}$ are the capability of transportation tools, and $\{b, c\}$ are the thresholds. Obviously, $A, b, B, c \geq 0$ hold.

Quadratic programming with soft constraints, where $g(\theta, x) = \theta^T x - x^T Q x$. One example is the classic minimum variance portfolio problem [16] with semi-definite positive covariance matrix Q

and expected return θ to be predicted. We extend it with soft constraints which, for example, may represent regulations on portions of equities for some fund types. Formally with $\alpha \geq 0$, we have:

$$\max_x \theta^T x - x^T Q x - \alpha^T \max(Cx - d, 0), \text{s.t. } Bx = c,\ x \geq 0. \tag{3}$$

Optimization of asymmetric soft constraints. This set of programming problems have the objective to match some expected quantities by penalizing excess and deficiency with probably different weights. Such formulation represents widespread resource provisioning problems, e.g., power[4] and cloud resources[1], where we minimize the cost of under-provisioning and over-provisioning against demands. Formally with $\alpha_1, \alpha_2 > 0$, we have:

$$\max_x -(\alpha_1^T \max(Cx - d, 0) + \alpha_2^T \max(d - Cx, 0)),\ \text{s.t. } Bx = c,\ x \geq 0 \tag{4}$$

In this paper, we consider a challenging task where C is a matrix to be predicted with known constants d. In reality, the $Cx - d$ term may represent the "wasted" part when satisfying the actual need of d.

2.2 Prediction+Programming

For compactness we write Eq.1 as $\max_{x \in \mathcal{X}} f(x, \theta)$, where f is the objective function and $\mathcal{X}$ is the feasible domain. The solver for f is to solve $x^* = \text{argmax}_{x \in \mathcal{X}} f(x, \theta)$. With parameters θ known, Eq.2–4 can be solved by mature solvers like Gurobi [6] and CPLEX [7].

In prediction+programming, θ is unknown and needs to be predicted from some observed features $\xi \in \Xi$. The prediction model is trained on a dataset $D = \{(\xi_i, \theta_i)\}_{i=1}^N$. In this paper, we consider the prediction model, namely Φ, as a neural network parameterized with ψ. In traditional supervised learning, Φ_ψ is learned by minimizing a generic loss, e.g., L1 (Mean Absolute Error) or L2 (Mean Squared Error), which measures the expected distance between predictions and real values. However, such loss minimization is often inconsistent with the programming objective f, especially when the prediction model is biased [11].

Instead, decision-focused learning directly optimizes ψ with respect to the programming objective f, that is, $\max_\psi \mathbf{E}_{(\xi,\theta)\sim D}[f(\hat{x^*}(\Phi_\psi(\xi)), \theta)]$. The full computational flow is illustrated in Fig.1. In the gradient-based learning, update of ψ's gradient is $\frac{\partial r}{\partial \psi} = \frac{\partial \hat{\theta}}{\partial \psi} \frac{\partial \hat{x^*}}{\partial \hat{\theta}} \frac{\partial r}{\partial \hat{x^*}}$, where utility $r = f(\hat{x^*}, \theta)$. The Jacobian $\partial\hat{\theta}/\partial\psi$ is computed implicitly by auto-differentiation of deep learning frameworks (e.g, PyTorch [17]), and $\partial r/\partial \hat{x^*}$ is analytical. The main challenge is to compute $\partial \hat{x^*}/\partial\hat{\theta}$, which depends on differentiating the argmax operation. One recent approach is to rewrite the objective to be convex (by adding a quadratic regularizer if necessary), build and differentiate the optimality conditions (e.g., KKT conditions) [18] which map $\hat{\theta}$ to the solution $\hat{x^*}$, and then apply implicit function theorem to obtain $\partial \hat{x^*}/\partial\hat{\theta}$. In this paper, we propose a novel approach that rewrites the problem as an unconstrained problem with soft constraints and derives analytical solutions, based on our observations on the real-world problem structures and coefficient properties.

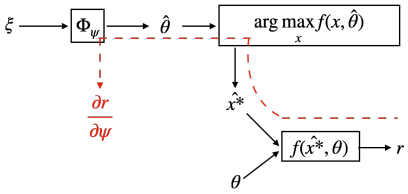


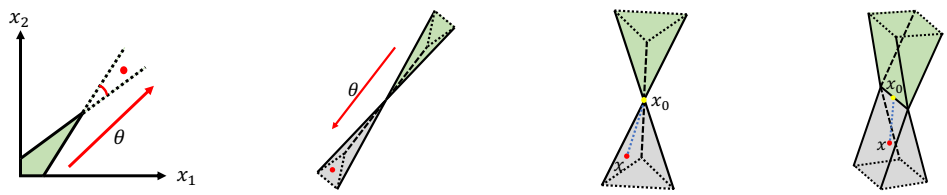
Figure 1: Computation graph of decision-focused prediction.

3 Methodology

Our main idea is to derive a surrogate function for f with a closed-form solution such that the Jacobian of the surrogate $\frac{\partial x}{\partial \theta}$ is analytical, making the computation of gradient straightforward. Unlike other recent work [13, 14, 15, 3], our method does not need to solve KKT optimality condition system. Instead, by adding high costs for infeasibility, we convert the constrained problem into an unconstrained one. With the assumption of concavity, we prove that there exist constant vectors $\beta_1, \beta_2, \beta_3$, such that Eq.1 can be equivalently transformed into an unconstrained problem:

$$\max_x L(x) = \max_x g(x, \theta) - \alpha^T \max(Cx - d, 0) - \beta_1^T \max(Ax - b, 0) - \beta_2^T |Bx - c| - \beta_3^T \max(-x, 0) \tag{5}$$

The structure of this section is as follows. Section 3.1 proves that the three types of hard constraints can be softened by deriving bounds of $\beta_1, \beta_2, \beta_3$; for this paper, we will assign each entry of the three vectors the equal value (with a slight abuse of notation, we denote $\beta_1 = \beta_2 = \beta_3 = \beta$ for the



(a) 2D acute angle (b) 3D acute angle (c) Non-degraded cone (d) Degraded cone

Figure 2: (a) and (b) are 2 and 3-dimensional "*acute angles*"; (c) and (d) shows two corresponding *activated cones* for given acute angles. The green area is the feasible region, x is the red point and x_0 is the yellow point; the red θ is the derivative of an objective $g(x, \theta) = \theta^T x$.

proofs of bounds; we align with the worst bound applicable to the problem formulation.) Section 3.2 proposes a novel surrogate function of $\max(\cdot, 0)$, such that the analytical form of $\frac{\partial x}{\partial \theta}$ can be easily derived via techniques of implicit differentiation [18] and matrix differential calculus [19] on equations derived by convexity [20]. Based on such derived $\frac{\partial x}{\partial \theta}$, we develop our end-to-end learning algorithm of prediction+programming, with its detailed procedure described in the appendix.

3.1 Softening the Hard Constraints

For any hard constraints $w = Ax - b \leq 0$, we denote its equivalent soft constraints as $H(w) = \beta^T \max(w, 0)$. $H(w)$ should satisfy two conditions: 1) for $w \leq 0$ (feasible x), $H(w) = 0$; 2) for $w \geq 0$ (infeasible x), $H(w)$ is larger than $g(x, \theta)$'s increments gained by violating $Ax - b \leq 0$. Intuitively, the second condition requires a sufficiently large-valued $\beta > 0$ to ensure that the optimization on the unconstrained surrogate objective never violates the original $Ax \leq b$; to make this possible, we assume that the $l2$-norm of the derivative of the objective f before conversion is bounded by constant E. The difficulty of requirement 2) is that the distance of a point to the convex hull l is not bounded by the sum of distance l_i to single hyper-plane i in general cases, so the utility gain obtained from violating constraints is unbounded. Fig. 2-(a) shows such an example which features the small angle between hyper-planes of the activated cone. We will refer such kind of 'unbounding' as "*acute angles*" below.

The main effort of this subsection is analyzing and bounding the effect caused by such acute angles. Given a convex hull $\mathcal{C} = \{z \in \mathbb{R}^n | Az \leq b, z \geq 0\}$ ($A \geq 0$ is not required) and any point $x \notin \mathcal{C}$, let $x_0 \in \mathcal{C}$ be the nearest point of $\mathcal{C}$ to x, and $A'x \geq b'$ represent all active constraints at x, then all such active constraints must pass through x_0.[1] The domain $\mathcal{K} = \{z \in \mathbb{R}^n | A'z \geq b', z \geq 0\}$ is a cone or degraded cone where the tip of the cone is a subspace. For the rest of the paper, we will call $\mathcal{K}$ *activated cone*, as shown in Fig. 2. Note that for any degraded activated cone, $x - x_0$ is always perpendicular to the tip subspace; therefore, we may project the cone onto the complementary space of the tip and get the same multiplier bound on the projected activated cone with lower dimensions.

Ideally, we aim to eliminate the utility gain $R = f(x, \theta) - \max_{x_1 \in \mathcal{C}} f(x_1, \theta)$ obtained from violating $A'x \leq b'$ with the penalty $\beta^T(A'x - b')$, *i.e.,* ensure $\beta^T(A'x - b') \geq R$ hold for any x. We now give a crucial lemma, which is the core of our proof for most of our theorem:

Lemma 1. *The utility gain R is bounded by $f(x, \theta) - f(x_0, \theta) \leq \frac{E}{\cos p_0} \sum_{i=1}^{n} (A'_i x - b'_i)$, where p_0 is the maximin angle among angles between θ and hyperplanes in the activated cone.*

Thus, setting $\beta = \frac{E}{\cos p_0} \mathbf{1}$ will give us a feasible β, and it suffices to find the lower bound for $\cos p_0$. For the rest of the section, we find the lower bound of $\cos p_0$ by exploiting the assumed properties of A', *e.g.,* $A' \geq 0$; we give the explanation of the full proofs for all theoretical results in the appendix.

3.1.1 Conversion of Inequality Constraints $Ax \leq b$

Let us first consider the constraints $w = Ax - b \leq 0$, where $A \geq 0$, $b \geq 0$. It is easy to prove that given $A \geq 0$ and $b \geq 0$, the distance of a point to the convex hull $Ax \leq b$ is bounded. More rigorously, we have the following theorem, which guarantees the feasibility of softening the hard constraints of inequalities $Ax \leq b$:

[1] Otherwise, there must exist an active constraint i and a small neighbourhood of x_0, say $B(x_0, \epsilon)$, such that $\forall z \in B(x_0, \epsilon)$, $A_i z - b_i \neq 0$, which implies that either constraint i is inactive (< 0) or $x_0 \notin \mathcal{C}$ (> 0).

Theorem 2. *For the optimization objective $\theta^T x$ with constraints $Ax \leq b$, $A \geq 0$, and $b \geq 0$, the utility gain R obtained from violating $w = Ax - b \leq 0$ has an upper bound of $O(\sum_i \max(w_i, 0)E)$.*

3.1.2 Conversion of Inequality Constraints $x \geq 0$

With inequality constraints $Ax \leq b$ converted, we now further enforce $x \geq 0$ into soft constraints. It seems that this constraint may form a very small angle to those in $w = Ax - b$. However, as $-x$ is aligned with the axes, we can augment $x \geq 0$ into soft constraints by proving the following theorem:

Theorem 3. *When there is at least one entry of $x \geq 0$ in the activated cone, the utility gain R from deviating the feasible region is bounded by $O(\frac{n^{1.5}E\sum_i max(w_i,0)}{\sin p})$, where p is the smallest corner between axes and other constraint hyper-planes and w is the union of $Ax - b$ and $-x$.*

Hence, we can set $\beta = O(\frac{n^{1.5}E}{\sin p})$. Specially, for cardinal constraints we may set $\beta = O(n^{1.5}E)$:

Corollary 4. *For cardinal constraints where the entries of A are binary, the utility gain R of violating $x \geq 0$ constraint is bounded by $O(n^{1.5}E\sum_i \max(w_i, 0))$, where $w_i = A_i x - b_i$ or $-x$.*

3.1.3 Conversion of Equality Constraints $Bx = c$

Finally, we convert $Bx = c$ into soft constraints. This is particularly difficult, as $Bx = c$ implies both $Bx \leq c$ and $-Bx \leq -c$, which will almost always cause acute angles. Let's first consider a special case where there is only one equality constraints and A is an element matrix $I^{n\times n}$.

Theorem 5. *If there is only one equality constraint $B^T x = c$ (e.g. $\sum_i x_i = 1$) and special inequality constraints $x \geq 0$, $Ix \leq b$, then the utility gain R from violating constraints is bounded by $O(\frac{n^{1.5}E\sum_i \max(w_i,0)}{\sin p})$, where p is the same with theorem 3, w is the union of $B^T x - c$ and $-x$.*

Intuitively, when there is only one equality constraint, the equality constraint can be viewed as an unequal one, for at most one side of the constraint can be in a non-degraded activated cone. Thus, we can directly apply the proof of Theorem 2 and 3, deriving the same bound $O(\frac{n^{1.5}E}{\sin p})$ for β.

Finally, we give bounds for general $Ax \leq b$, $Bx = c$ with $A, b, B, c \geq 0$ as below:

Theorem 6. *Given constraints $Ax \leq b$, $x \geq 0$, and $Bx = c$, where $A, B, b, c \geq 0$, the utility gain R obtained from violating constraints is bounded by $O(\sqrt{n}\lambda_{max}\sum_i \max(w_i, 0))$, where λ_{max} is the upper bound for eigenvalues of $P^T P$ ($P : x \to Px$ is an orthogonal transformation for an n-sized subset of normalized row vectors in A, B and $-I$).*

Though there are no direct bounds on λ_{max} with respect to n and the angles between hyper-planes, empirical results (see appendix for results) show that for a diverse set of synthetic data distributions, $\lambda_{max} = O(n^2)$. Therefore, empirically we can use a bound $O(\frac{n^{2.5}E}{\sin p})$ for β.[2]

So far, we have proven that all hard constraints can be transformed into soft constraints with bounded multipliers. For compactness, Eq.5 is rewritten in a unified form:

$$L(x) = g(x, \theta) - \gamma^T \cdot max(C'x - d'), \text{ where } \gamma = \begin{bmatrix} \alpha \\ O(\frac{n^{2.5}E}{\sin p})\mathbf{1} \end{bmatrix}, \; C' = \begin{bmatrix} C \\ A \\ -B \\ B \\ -I \end{bmatrix}, \; d' = \begin{bmatrix} d \\ b \\ -c \\ c \\ 0 \end{bmatrix} \tag{6}$$

3.2 The Unconstrained Soft Constraints

Eq.5 is non-differentiable for the $\max$ operator. We thus seek a relaxing surrogate. The most apparent choice of such surrogate $S(z)$ for $z = C'x - d'$ is sigmoidal functions; however, it is difficult to derive a closed-form solution for such functions, since sigmoidal functions yield z in the denominator, and z cannot be directly solved because C is not invertible (referring to appendix for detailed reasons). Therefore, we have to consider a piecewise roundabout where we can first numerically solve the optimal point to determine which segment the optimal point is on, and then expand the segment to the whole space. To make this feasible, two assumptions must be made: 1) this function must be differentiable, and 2) the optimal point must be unique; to ensure this, the surrogate should be a

[2]For real programming problems, we can sample A' from A and estimate the largest eigenvalue of $P^T P$.

convex/concave piece-wise function. The second property is for maintaining the optimal point upon segment expansion. Fortunately, there is one simple surrogate function satisfying our requirement:

$$S(z) = \begin{cases} 0 & \text{if } z < -\frac{1}{4K} \\ K(z + \frac{1}{4K})^2 & \text{if } -\frac{1}{4K} \leq z \leq \frac{1}{4K} \\ z & \text{if } z \geq \frac{1}{4K} \end{cases} \tag{7}$$

Let M and U be diagonal matrices as the indicator of $S(z)$, which are $M_{i,i} = 2K[-\frac{1}{4K} \leq z_i \leq \frac{1}{4K}]$ and $U_{i,i} = [\frac{1}{4K} < z_i]$, where $[\cdot]$ is an indicator function. $K > 0$ is a hyper-parameter that needs to be balanced. Larger K makes the function more loyal to original; however, if K is too large, then the training process would be destabilized, because when the prediction error is large at the start of the training process, $\frac{\partial f}{\partial \hat{\theta}}|_{\hat{x}}$ might be too steep. Then consider the optimal point for the unconstrained optimization problem maximizing $\theta^T x - \gamma^T \max(C'x - d', 0)$, with ground truth θ_{real}, we can obtain:

$$\begin{aligned} \theta = C'^T M \text{diag}(\gamma)(C'x - d') + C'^T(\frac{1}{4K}M + U)\gamma, \quad \frac{\partial x}{\partial \theta} = (C'^T M \text{diag}(\gamma) C')^{-1} \\ \frac{\partial f(x, \theta_{\text{real}})}{\partial x} = \theta_{\text{real}} - C'^T M \text{diag}(\gamma)(C'x - d') - C'^T(\frac{1}{4K}M + U)\gamma \end{aligned} \tag{8}$$

With the above derivations, we can train our prediction model with stochastic gradient descent (SGD). For each batch of problem instances, we first solve programming problems numerically using solvers like Gurobi to get the matrix M and U, and then calculate gradients with the analytical solution. The parameters of the prediction model are updated by such gradients. The sketch of our algorithm is outlined in the appendix.

4 Applications and Experiments

We apply and evaluate our approach on three formulated problems described in Section 2, *i.e.,*linear programming, quadratic programming, and asymmetric soft constraint minimization. These problems are closely related to three applications respectively: synthetic linear programming, portfolio optimization, and resource provisioning, which are constructed using synthetic or real-world datasets.

In our experiments, the performance is measured in regret, which is the difference between the objective value when solving programming over predicted parameters and the objective value when solving programming over actual parameters. For each experiment, we choose two generic two-stage methods with $L1$-loss and $L2$-loss, as well as one more advanced method for baseline. We choose SPO+ for synthetic linear programming and DF proposed by Wilder *et al.* [9] for portfolio optimization,[3] as the former is specially designed for for linear objective, and the latter is designed for quadratic objective with application on a very similar problem [3]. For resource provisioning, we use a problem-specific weighted L1 loss, as both SPO+ and DF are not designed for gradients with respect to variables in the soft constraints. All reported results for each method are obtained by averaging on 15 independent runs with different random seeds.

4.1 Synthetic Linear Programming

For the formulation in Eq.2 without equality constraints, we equivalently solve Eq.6 with $C' = \begin{bmatrix} C \\ A \\ -I \end{bmatrix}, d' = \begin{bmatrix} d \\ b \\ 0 \end{bmatrix}, \gamma = \begin{bmatrix} \alpha \\ O(\frac{n^{1.5}E}{\sin p})\mathbf{1} \\ O(\frac{n^{1.5}E}{\sin p})\mathbf{1} \end{bmatrix}$, where p is the minimum angle between hyper-planes of $Ax \leq b$ and the axes. Then, we have $x = (C'^T M \text{diag}(\gamma) C')^{-1}(\theta + C'^T M \text{diag}(\gamma) d' - C'^T(\frac{1}{4K}M + U)\gamma)$, and the derivatives can be obtained straightforwardly, as described in Eq.8.

Problem setup. The prediction dataset $\{(\xi_i, \theta_i)\}_{i=1}^N$ is generated by a general structural causal model ([21]), ensuring it is representative and consistent with physical process in nature. The programming parameters are generated for various scales in numbers of decision variables, hard constraints, and soft constraints. Full details are given in the appendix.

Experimental setup. All four methods use the same prediction model – a fully connected neural network of two hidden layers with 128 neurons for each and ReLU [22] for activation. We use

[3]The method [3] and its variant with dimension reduction [9] have same performance in terms of regret on this problem, thus we choose the former for comparison convenience.

		Regret (the lower, the better)			
N	Problem Size	L1	L2	SPO+ [10]	Ours
100	(40, 40, 0)	2.454±0.232	2.493±0.295	2.506±0.294	**2.258±0.311**
	(40, 40, 20)	2.626±0.307	2.664±0.303	2.667±0.281	**2.350±0.263**
	(80, 80, 0)	5.736±0.291	5.831±0.361	5.711±0.309	**5.200±0.506**
	(80, 80, 40)	4.786±0.403	4.786±0.596	4.939±0.382	**4.570±0.390**
1000	(40, 40, 0)	1.463±0.143	1.447±0.155	1.454±0.148	**1.346±0.144**
	(40, 40, 20)	1.626±0.141	1.613±0.110	1.618±0.103	**1.506±0.102**
	(80, 80, 0)	3.768±0.132	3.718±0.117	3.573±0.113	**3.431±0.100**
	(80, 80, 40)	2.982±0.176	2.913±0.172	2.879±0.148	**2.781±0.165**
5000	(40, 40, 0)	1.077±0.105	1.080±0.109	1.090±0.105	**1.037±0.100**
	(40, 40, 20)	1.283±0.070	1.277±0.077	1.298±0.077	**1.220±0.071**
	(80, 80, 0)	2.959±0.086	2.943±0.091	2.926±0.079	**2.845±0.064**
	(80, 80, 40)	2.239±0.122	2.224±0.106	2.234±0.122	**2.172±0.098**

Table 1: Performance comparison (regret mean with std. deviation) for the synthetic linear programming problem. N is the size of the training dataset, and problem size is a triplet (# of decision variables' dimension, # of hard constraints, # of soft constraints).

Adagrad [23] as the optimizer, with learning rate 0.01 and gradient clipped at $1e-4$. We train each method for 40 epochs, and early stop when valid performance degrades for 4 consecutive epochs.

Performance analysis. Our experiments cover four programming problem scales with three prediction dataset sizes. Results are summarized in Table 1. In all cases, our method performs consistently better than two-stage methods and SPO+. Even for the cases with only hard constraints (*i.e.,* the third parameter of problem size is 0), our method still has significant advantage, demonstrating its effectiveness on handling hard constraints. Surprisingly, although the main objective is linear, SPO+ often performs even worse than two-stage methods. Detailed analysis (see the appendix) shows that SPO+ quickly reaches the test optimality and then over fits. This may be due to that, unlike our method, SPO+ loss is not designed to align with the soft constraints. Besides, with the increment of the samples in train data, the performance of all methods is improved significantly and the performance gap among ours and two-stage methods becomes narrow, which implies that prediction of two-stage methods becomes better and with lower biases. Even so, our method has better sample efficiency than two-stage methods.

We also investigated effect of the hyper-parameter K in our surrogate $\max$ function, detailed in the appendix. Through our experiments, K's effect to regret is not monotonic, and its optimal value varies for different problem sizes. Interestingly, K's effect is approximately smooth. Thus, in practice, we use simple grid search to efficiently find the best setting of K.

4.2 Portfolio Optimization

We consider the minimum variance portfolio problem [16], and extend its objective with a term of soft constraints, as shown in Eq.3. Here, x is the decision variable vector – equity weights, while θ is the equity returns. The semi-definite positive Q is the covariance matrix of returns θ. We rewrite the constraints of x to fit our surrogate framework, as $Ax \leq b$ where $A = \begin{bmatrix} -\mathbf{1}_{1\times n} \\ \mathbf{1}_{1\times n} \\ -I \end{bmatrix}$, $b = \begin{bmatrix} -1 \\ 1 \\ 0 \end{bmatrix}$. Let $C' = \begin{bmatrix} C \\ A \end{bmatrix}, d' = \begin{bmatrix} d \\ b \end{bmatrix}, \gamma = \begin{bmatrix} \alpha \\ O(n^{1.5})\mathbf{1} \end{bmatrix}$. Then we have $x = (2Q + C'^T M \text{diag}(\gamma) C')^{-1}(\theta + C'^T M \text{diag}(\gamma) d' - C'^T (U + \frac{M}{4K})\gamma)$, and the gradients are as follows:

$$\begin{aligned} \frac{\partial f(x, \theta_{\text{real}}, Q_{\text{real}})}{\partial x} &= \theta_{\text{real}} - 2Q_{\text{real}} x - C'^T M \text{diag}(\gamma)(C'x - d') - C'^T (\frac{1}{4K} M + U)\gamma \\ \frac{\partial x}{\partial \theta} &= (2Q + C'^T M \text{diag}(\gamma) C')^{-1}, \ \frac{\partial x}{\partial Q} = (\theta + C'^T M \text{diag}(\gamma) d' - C'^T (U + \frac{M}{4K})\gamma) \frac{\partial \frac{\partial x}{\partial \theta}}{\partial Q} \end{aligned} \tag{9}$$

See appendices for the derivation process and the simplified result of $\frac{\partial x}{\partial Q}$.

Problem and experimental setup. The prediction dataset is daily price data of SP500 from 2004 to 2017 downloaded by Quandl API [24] with the same settings in [3]. To make DF work with the non-differentiable objective, we use local 2nd-order Taylor expansion as a quick fix, which was used in [3] for non-convex optimization applications. Most settings are aligned with those in [3], including

	Regret measured in % (the lower, the better)			
#Equities	L1	L2	DF [9]	ours($K = 100$)
50	4.426±0.386	4.472±0.385	4.016±0.389	**3.662±0.238**
100	4.262±0.231	4.320±0.229	3.500±0.252	**3.214±0.138**
150	3.878±0.281	3.950±0.287	3.419±0.281	**3.109±0.162**
200	3.755±0.236	3.822±0.273	3.406±0.287	**3.152±0.183**
250	3.721±0.205	3.751±0.212	3.335±0.175	**3.212±0.135**

Table 2: Performance comparison (regret mean with std. deviation) for portfolio optimization.

dataset configuration, prediction model, learning rate (0.01), optimizer (Adam), gradient clip (0.01) and number of training epochs (20). We set the number of soft constraints to 0.4 times of n, where n is the number of candidate equities. For the soft constraint $\alpha^T \max(Cx - d, 0)$, $\alpha = \frac{15}{n}v$, where each element of v is generated randomly at uniform from $(0, 1)$; the elements of matrix C is generated independently from $\{0, 1\}$, where the probability of 0 is 0.9 and 1 is 0.1. K is set as 100.

Performance analysis. Table 2 summarizes the experimental results. In total, on all problem sizes (#equities), our method performs consistently and significantly better than both two-stage (L1 and L2) methods and the decision focused DF[9]. Among the three baselines, DF is significantly better than two-stage methods, similar to results in [3]. In fact, DF under this setting can be viewed as a variant of our method with infinite K and no conversion of softening $\sum_i x_i = 1$. The comparison to DF also demonstrates the advantage of our method on processing such non-differentiable cases against the simple 2nd-order Taylor expansion. Besides, with the increment of the number of equities, regrets of all methods decease, which indicates that for the constraint $\sum_i x_i = 1$, larger number of equities brings smaller entries of x on average (with the presence of Q, there are many non-zero entries of x), lowering the impact of prediction error for any single entry.

4.3 Resource Provisioning

The objective of this problem is to minimize the under-provisioning and over-provisioning costs, which are naturally formulated as soft constraints. In particular, we consider a hard problem setting – to predict the high dimensional matrix C. Then, for the problem formulation given in Eq.4, we have

$$\begin{aligned} \frac{\partial f(x, C'_{\text{real}})}{\partial x} &= -C'^T_{\text{real}} M \text{diag}(\gamma)(C'_{\text{real}} x - d') - C'^T (\frac{1}{4K} M_{\text{real}} + U_{\text{real}})\gamma \\ \frac{\partial x_i}{\partial C'_{k,l}} &= -((M\text{diag}(\gamma) C' P \rho)_k P_{i,l} + (P\rho)_l (P C'^T M \text{diag}(\gamma))_{i,k}) + P_{i,l} \cdot \eta_k \end{aligned} \tag{10}$$

where $\gamma = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ O(n^{1.5})\mathbf{1} \\ O(n^{1.5}) \\ O(n^{1.5}) \end{bmatrix}$, $C' = \begin{bmatrix} C \\ -C \\ -I \\ \mathbf{1_{1\times n}} \\ -\mathbf{1_{1\times n}} \end{bmatrix}$,$C'_{\text{real}} = \begin{bmatrix} C_{\text{real}} \\ -C_{\text{real}} \\ -I \\ \mathbf{1_{1\times n}} \\ -\mathbf{1_{1\times n}} \end{bmatrix}$, $d' = \begin{bmatrix} d \\ -d \\ 0 \\ 1 \\ -1 \end{bmatrix}$, M_{real} and U_{real} represent the segment determined by C_{real} and the optimal point x with predicted C, M and U represent the segment determined by C and the optimal point x with predicted C, $P = (C'^T M \text{diag}(\gamma) C')^{-1}$, $\rho = (C'^T M \text{diag}(\gamma) d' - C'^T (\frac{M}{4K} + U)\gamma)$, and $\eta = M\text{diag}(\gamma)d' - (\frac{M}{4K} + U)\gamma$.

Problem setup. We use ERCOT energy dataset [25], which contains hourly data of energy output from 2013 to 2018, 52535 data points in total. We use the last 20% samples for test. We aim to predict the matrix $C \in \mathbb{R}^{24\times 8}$, the loads of 24 hours in 8 regions. The decision vairable x is 8-dimensional, and $d = 0.5 \times \mathbf{1} + 0.1N(0, 1)$. We test five sets of (α_1, α_2), with α_1/α_2 ranging from 100 to 0.01.

Experimental setup. We use AdaGrad with learning rate of 0.01, and clip the gradient with norm 0.01. For the prediction model, we use McElWee's network [26] which was highly optimized for this task, with $(8 \times 24 \times 77)$-dimensional numerical features and embedding ones as input.

Performance analysis. Table 3 shows the experimental results. The absolute value of regret differs largely across different ratios of α_1/α_2. Our method is better than other methods, except for $\alpha_1/\alpha_2 = 1$, where the desired objective is exactly $L1$-loss and thus $L1$ performs the best. Interestingly, compared to L1/L2, the Weighted L1 performs better when $\alpha_1/\alpha_2 > 1$, but relatively worse otherwise. This is probably due to the dataset's inherent sample bias (*e.g.,* asymmetric distribution and occasional peaks), which causes the systematic bias (usually underestimation) of the prediction model. This bias exacerbates, when superposed with weighted penalty multipliers which

	Regret (the lower, the better)			
α_1/α_2	L1	L2	Weighted L1	Ours($K = 0.05$)
100	105.061±21.954	93.193±29.815	79.014±32.069	**20.829±8.289**
10	13.061±2.713	13.275±6.208	7.743±1.305	**2.746±1.296**
1	**4.267±0.618**	5.136±0.722	**4.267±0.618**	5.839±0.512
0.1	10.846±1.606	13.619±2.195	16.462±2.093	**10.240±1.248**
0.01	99.145±21.159	118.112±29.957	230.825±91.184	**94.341±29.821**

Table 3: Performance comparison (regret mean with std. deviation) for resource provisioning.

encourage the existing bias to fall on the wrong side. Besides, the large variance for weighted L1 at $\alpha_1/\alpha_2 = 0.01$ is caused by a pair of outliers.

5 Related Work

Differentiating $\mathbf{argmin/argmax}$ through optimality conditions. For convex optimization problems, the KKT conditions map coefficients to the set of solutions, and thus can be differentiated for argmin using implicit function theorem. Following this idea, existing work developed implicit layers of argmin in neural network, including OptNet [13] for quadratic programs (QP) problems and CVXPY [14] for more general convex optimization problems. Further with linear relaxation and QP regularization, Wilder *et al.* derived an end-to-end framework for combinatorial programs [9], which accelerates the computation by leverage the low-rank properties of decision vectors [3], and is further extended to mixed integer linear programs in MIPaaL [27]. Besides, for the relaxed LP problems, instead of differentiating KKT conditions, IntOpt [28] proposes an interior point based approach which computes gradients by differentiating homogeneous self-dual formulation.

Optimizing surrogate loss functions. Elmachtoub and Grigas [10] proposed a convex surrogate loss function, namely SPO+, measuring the decision error induced by a prediction, which can handle polyhedral, convex and mixed integer programs with linear objectives. TOPNet [29] proposes a learned surrogate approach for exotic forms of decision loss functions, which however is hard to generalize to handle constrained programs.

Differentiating argmin is critical for gradient methods to optimize decision-focused prediction models. Many kinds of efforts, including regularization for specific problems (e.g., differentiable dynamic programming [30], differentiable submodular optimization [31]), reparameterization [32], perturbation [33] and direct optimization ([12]), are spent for optimization with discrete or continuous variables and still actively investigated.

As a comparison, our work proposes a surrogate loss function for constrained linear and quadratic programs extended with soft constraints, where the soft constraints were not considered principally in previous work. Also, unlike OptNet [13] and CVXPY [15], our method does not need to solve KKT conditions. Instead, by adding reasonable costs for infeasibility, we convert the constrained optimization problem into an unconstrained one while keeping the same solution set, and then derive the required Jacobian analytically. To some degree, we reinvent the exact function [34] in prediction+programming.

6 Conclusion

In this paper, we propose a novel surrogate objective framework for prediction+programming problems on linear and negative definite quadratic objectives with linear soft constraints which contain non-differentiable argmax operator. This framework gives the theoretical bounds on constraints' multipliers, and derives the closed-form solution with respect to predictive parameters and thus gradients for any variable in the problem. We first convert the hard constraints into soft ones with reasonable constant multipliers, making the problem unconstrained, and then optimize a piecewise surrogate locally. We apply and empirically validate our method on three applications against traditional two-stage methods and other end-to-end methods. We believe our work is an important enhancement to the current prediction+programming toolbox. Future work is to model the prediction in distribution rather than point estimation in current prediction+programming methods, to further enhance optimization robustness.

References

[1] Chuan Luo, Bo Qiao, Xin Chen, Pu Zhao, Randolph Yao, Hongyu Zhang, Wei Wu, Andrew Zhou, and Qingwei Lin. Intelligent virtual machine provisioning in cloud computing. In *Proceedings of IJCAI 2020*, pages 1495–1502, 2020.

[2] Sirikarn Chansombat, Pupong Pongcharoen, and Christian Hicks. A mixed-integer linear programming model for integrated production and preventive maintenance scheduling in the capital goods industry. *International Journal of Production Research*, 57:1–22, 04 2018. doi: 10.1080/00207543.2018.1459923.

[3] Kai Wang, Bryan Wilder, Andrew Perrault, and Milind Tambe. Automatically learning compact quality-aware surrogates for optimization problems. In *NeurIPS 2020*, Vancouver, Canada, 2020.

[4] Emir Demirovi, Peter Stuckey, Tias Guns, James Bailey, Christopher Leckie, Kotagiri Ramamohanarao, and Jeffrey Chan. Dynamic programming for predict+optimise. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34:1444–1451, 04 2020. doi: 10.1609/aaai.v34i02.5502.

[5] Gartheeban Ganeshapillai, John Guttag, and Andrew Lo. Learning connections in financial time series. In Sanjoy Dasgupta and David McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 109–117, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR.

[6] LLC Gurobi Optimization. Gurobi optimizer reference manual, 2021. URL `http://www.gurobi.com`.

[7] IBM ILOG Cplex. V12. 1: User's manual for cplex. *International Business Machines Corporation*, 46(53):157, 2009.

[8] Yoshua Bengio. Using a financial training criterion rather than a prediction criterion. *Int. J. Neural Syst.*, 8(4):433–443, 1997. doi: 10.1142/S0129065797000422. URL `https://doi.org/10.1142/S0129065797000422`.

[9] Bryan Wilder, Bistra Dilkina, and Milind Tambe. Melding the data-decisions pipeline: Decision-focused learning for combinatorial optimization. In *The Thirty-Third AAAI Conference on Artificial Intelligence*, pages 1658–1665. AAAI Press, 2019.

[10] Adam N. Elmachtoub and Paul Grigas. Smart "predict, then optimize", 2020.

[11] Priya L. Donti, Brandon Amos, and J. Zico Kolter. Task-based end-to-end model learning in stochastic optimization. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 5490–5500, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

[12] Yang Song, Alexander Schwing, Richard, and Raquel Urtasun. Training deep neural networks via direct loss minimization. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 2169–2177, New York, New York, USA, 20–22 Jun 2016. PMLR.

[13] Brandon Amos and J. Zico Kolter. OptNet: Differentiable optimization as a layer in neural networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 136–145, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR. URL `http://proceedings.mlr.press/v70/amos17a.html`.

[14] Brandon Amos. *Differentiable Optimization-Based Modeling for Machine Learning*. PhD thesis, Carnegie Mellon University, May 2019.

[15] Akshay Agrawal, Brandon Amos, Shane Barratt, Stephen Boyd, Steven Diamond, and J. Zico Kolter. Differentiable convex optimization layers. In *Advances in Neural Information Processing Systems*, volume 32, pages 9562–9574. Curran Associates, Inc., 2019.

[16] Harry M. Markowitz. *Portfolio Selection: Efficient Diversification of Investments*. Yale University Press, 1959. ISBN 9780300013726. URL http://www.jstor.org/stable/j.ctt1bh4c8h.

[17] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[18] A. Griewank and A. Walther. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation, Second Edition*. Other Titles in Applied Mathematics. Society for Industrial and Applied Mathematics (SIAM, 3600 Market Street, Floor 6, Philadelphia, PA 19104), 2008. ISBN 9780898717761.

[19] Jan R. Magnus Jan R. Magnus. *Matrix Differential Calculus with Applications in Statistics and Econometrics, Third Edition*. Wiley Series in Probability and Statistics. Wiley, 2019. ISBN 9781119541202.

[20] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[21] Jonas Peters, Dominik Janzing, and Bernhard Schölkopf. *Elements of causal inference*. The MIT Press, 2017.

[22] Vinod Nair and Geoffrey Hinton. Rectified linear units improve restricted boltzmann machines vinod nair. volume 27, pages 807–814, 06 2010.

[23] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159, 07 2011.

[24] Sp500 data set. 2021. URL https://docs.quandl.com/.

[25] Ercot data set. 2021. URL https://github.com/kmcelwee/mediumBlog/tree/master/load_forecast/data.

[26] Kevin McElwee. Predict daily electric consumption with neural networks, 2020. URL https://www.brownanalytics.com/load-forecasting.

[27] Aaron Ferber, Bryan Wilder, Bistra Dilkina, and Milind Tambe. Mipaal: Mixed integer program as a layer. In *AAAI Conference on Artificial Intelligence*, 2020.

[28] Jayanta Mandi and Tias Guns. Interior point solving for lp-based prediction+optimisation. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 7272–7282. Curran Associates, Inc., 2020.

[29] Di Chen, Yada Zhu, Xiaodong Cui, and Carla Gomes. Task-based learning via task-oriented prediction network with applications in finance. In Christian Bessiere, editor, *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pages 4476–4482. International Joint Conferences on Artificial Intelligence Organization, 7 2020. doi: 10.24963/ijcai.2020/617. URL https://doi.org/10.24963/ijcai.2020/617. Special Track on AI in FinTech.

[30] Arthur Mensch and Mathieu Blondel. Differentiable dynamic programming for structured prediction and attention. In *35th International Conference on Machine Learning*, volume 80, pages 3459–3468, 2018.

[31] Josip Djolonga and Andreas Krause. Differentiable learning of submodular models. In *Advances in Neural Information Processing Systems*, volume 30, pages 1013–1023. Curran Associates, Inc., 2017.

[32] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, page II–1278–II–1286. JMLR.org, 2014.

[33] Quentin Berthet, Mathieu Blondel, Olivier Teboul, Marco Cuturi, Jean-Philippe Vert, and Francis Bach. Learning with differentiable perturbed optimizers. *arXiv preprint arXiv:2002.08676*, 2020.

[34] Dimitri P Bertsekas. Nonlinear programming. *Journal of the Operational Research Society*, 48 (3):334–334, 1997.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] We theoretically and empirically validated our contributions and scope claimed in the abstract in the paper.
   (b) Did you describe the limitations of your work? [Yes] See the end of "Choices of Surrogate $\max$ Functions" in the appendix.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] We have discussed about this issue and concluded that our work has no potential negative societal impact.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes] We have read the ethics review guidelines and ensured that our paper conforms to them.
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] We stated assumptions of all theoretical results.
   (b) Did you include complete proofs of all theoretical results? [Yes] See appendix for the complete proofs for all theoretical results.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See the publicized code.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] We specify all training details in the experiment section.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] We report error bars of standard deviation in the form of tables (instead of figures) due to space limit. We also conduct significance test in the appendix.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See "Computing Infrastructure" in the appendix.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] We cite all creators of assets that we used.
   (b) Did you mention the license of the assets? [Yes] See the publicized code.
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] See the publicized code.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes] The data are all publicly available.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes] The data does not contain any personally identifiable information or offensive content.

5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A] We did not use crowdsourcing, nor did we conduct research with human subjects.
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]