Faster Algorithms for Learning Convex Functions

Ali Siahkamari [*] [1] Durmus Alp Emre Acar [*] [1] Christopher Liao [1] Kelly Geyer [1] Venkatesh Saligrama [1]
Brian Kulis [1]

Abstract

The task of approximating an arbitrary convex function arises in several learning problems such as convex regression, learning with a difference of convex (DC) functions, and learning Bregman or f-divergences. In this paper, we develop and analyze an approach for solving a broad range of convex function learning problems that is faster than state-of-the-art approaches. Our approach is based on a 2-block ADMM method where each block can be computed in closed form. For the task of convex Lipschitz regression, we establish that our proposed algorithm converges with iteration complexity of $O(n\sqrt{d}/\epsilon)$ for a dataset $\boldsymbol{X} \in \mathbb{R}^{n\times d}$ and $\epsilon > 0$. Combined with per-iteration computation complexity, our method converges with the rate $O(n^3 d^{1.5}/\epsilon + n^2 d^{2.5}/\epsilon + nd^3/\epsilon)$. This new rate improves the state of the art rate of $O(n^5 d^2/\epsilon)$ if $d = o(n^4)$. Further we provide similar solvers for DC regression and Bregman divergence learning. Unlike previous approaches, our method is amenable to the use of GPUs. We demonstrate on regression and metric learning experiments that our approach is over 100 times faster than existing approaches on some data sets, and produces results that are comparable to state of the art.

1. Introduction

The importance of convex functions in machine learning is undisputed. However, while most applications of convexity in machine learning involve fixed convex functions, an emerging trend in the field has focused on the problem of *learning* convex functions for a particular task. In this setting, data or supervision is used to tailor a convex function for some end goal. Recent machine learning examples include learning with a difference of convex (DC) functions (Siahkamari et al., 2020), learning input convex neural networks for tasks in reinforcement learning or vision (Amos et al., 2017), and learning divergences between points or distributions via learning the underlying convex function in a Bregman divergence (Siahkamari et al., 2019) or f-divergence (Zhang et al., 2020) for problems such as data generation, metric learning, and others.

In fact, work in convex function learning at least dates back to methods developed for the problem of *convex regression*, an important learning problem that is used commonly in econometrics and engineering for modeling demand, utility and production curves (Afriat, 1967; Varian, 1982). Convex regression is the problem of estimating a convex function when receiving a dataset $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$, where $\boldsymbol{x}_i \in \mathbb{R}^d$ are predictors and y_i are continuous responses. Consider

$$\mathcal{F} \triangleq \{f : \mathbb{R}^d \to \mathbb{R} \mid f \text{ is convex}\}, \tag{1}$$

as the class of all convex functions over $\mathbb{R}^d$. Then an estimator is proposed by minimizing the squared error between observations $\boldsymbol{x}_i$ and responses y_i,

$$\hat{f} \triangleq \arg\min_{f\in\mathcal{F}} \frac{1}{n}\sum_{i=1}^{n}(y_i - f(\boldsymbol{x}_i))^2 + \lambda\|f\|, \tag{2}$$

where $\|f\|$ is some penalty term. A key insight about this problem is that although Eq. (2) is an infinite dimensional minimization problem, Boyd et al. (2004) shows it can be solved as a convex optimization problem since the optimal solution can be shown to be piece-wise linear. Furthermore, generalization bounds for this problem have recently been developed in (Balázs, 2016; Siahkamari et al., 2020) using certain classes for $\|f\|$.

Many other convex function learning problems have a similar structure to them, and also yield resulting optimization problems that can be tractably solved. However, a key downside is that the computational complexity is often prohibitive for real-world applications, particularly due to the non-parametric nature of the resulting optimization problems. For instance, the state-of-the-art solver for convex regression is based on interior point methods and has a complexity of $O(n^5 d^2/\epsilon)$ for a given accuracy ϵ.

Thus, our goal in this paper is to study a class of methods that yields provably faster convergence for a number of dif-

[*]Equal contribution [1]Boston University, Boston, MA. Correspondence to: Ali Siahkamari <siaa@bu.edu>.

Proceedings of the 39th International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022. Copyright 2022 by the author(s).

ferent convex function learning problems. Our main insight is that many convex function learning problems may be expressed as optimization problems that can be solved with a 2-block ADMM algorithm such that each of the two blocks can be computed *in closed form*. For convex regression, the resulting method (which we call FCR, for Faster Convex Regression) is guaranteed to converge with computational complexity of $O(n^3d^{1.5}/\epsilon+n^2d^{2.5}/\epsilon+nd^3/\epsilon)$ for a dataset $\boldsymbol{X}\in\mathbb{R}^{n\times d}$ and $\epsilon>0$. This new rate improves the state of the art $O(n^5d^2/\epsilon)$ in (Balázs, 2016) available via interior point methods when $d=o(n^4)$. In addition to an improved convergence rate, FCR makes several contributions. Firstly, FCR is stand-alone and does not need any optimization software. FCR is based on tensor computations and can easily be implemented on GPUs. Furthermore, FCR can include regularization terms in the context of non-parametric convex regression.

We extend FCR solver for problems beyond convex regression. In particular, we provide 2-block ADMM solvers and present empirical comparisons for DC regression (Siahkamari et al., 2020) and Bregman divergence learning (Siahkamari et al., 2019). Finally, we demonstrate empirical results showing that our approach yields significantly more scalable convex learning methods, resulting in speedups of over 100 times on some problems.

Notation. We generally denote scalars as lower case letters, vectors as lower case bold letters, and matrices as upper case bold letters. For a dataset $\boldsymbol{X}\in\mathbb{R}^{n\times d}$, n represents the number of observations and d is the number of covariates. We define $x^+=\max\{x,0\}, x^-=\max\{-x,0\}$ and $\times$ as an outer product. We define $[n]$ to be the set of integers $\{1,\dots,n\}$. By $\|\nabla^* f(\boldsymbol{x})\|$ and $|\partial^*_{x_l} f(\boldsymbol{x})|$ we denote the largest subgradient and the largest partial sub-derivative.

1.1. Connections to Existing Methodologies

Convex regression has been extensively studied over the last two decades. Boyd et al. (2004) introduce the non-parametric convex regression problem Eq. (2). Balázs (2016) show that convex Lipschitz regression requires regularization in order to have generalization guarantees. In particular, it was shown that the penalty term $\|f\|=\sup_{\boldsymbol{x}}\|\nabla^* f(\boldsymbol{x})\|$ yields a generalization error of $O(n^{-2/d}\log n)$.

Balázs (2016); Mazumder et al. (2019) provide ADMM solvers for the convex regression problem; however, solutions have more than two blocks and are not guaranteed to converge in all circumstances. More recently, Chen & Mazumder (2020) provide an active-set type solver and Bertsimas & Mundru (2021) provide a delayed constraint generation algorithm which have favorable scalablity. They use a penalty of $\|f\|=\sum_i\|\nabla^* f(\boldsymbol{x}_i)\|_2^2$, which makes the loss function in Eq. (2) strongly convex and easier to solve. However, no theoretical generalization bound is known when using this new penalty term. Siahkamari et al. (2020) use $\|f\|=\sup_{\boldsymbol{x}}(\|\nabla^* f(\boldsymbol{x})\|_1)$ for learning a difference of convex functions where they provide a multi-block ADMM solver for this problem.

On the other hand, Ghosh et al. (2019) and Kim et al. (2021) study the parametric class of max-linear functions $\mathcal{F}_k=\{f:\mathbb{R}^d\to\mathbb{R}\mid f(\boldsymbol{x})=\max_{j=1}^k\langle\boldsymbol{a}_j,\boldsymbol{x}\rangle+b_j\}$ for $k<n$. They provide convex programming and alternating algorithms. The downside of these methods is the assumption that $\boldsymbol{x}_i$ are i.i.d samples from a normal distribution and furthermore that each feature $x_{i,l}$ is independent. These assumptions are needed for their algorithm to converge in theory. We note that this parametric class $\mathcal{F}_k$ if $k\ge n$, is the same as $\mathcal{F}$, when used in convex regression minimization problem Eq. (2).

Our methodology is most similar to Balázs (2016) and is in the context of learning theory where we do not require any distributional assumption on $\boldsymbol{x}_i$ and only require them be i.i.d. and bounded. We further need to know a bound on f but not on $\|f\|$.

2. Convex Regression with Lasso Penalty

Suppose we are given a dataset $\{(y_i,\boldsymbol{x}_i)\}_{i=1}^n$ where $\boldsymbol{x}_i\in\mathbb{R}^d$ are predictors, assumed drawn i.i.d. from a distribution P_X supported on a compact domain $\Omega\subset\{\|\boldsymbol{x}\|_\infty\le R\}$, with $n\ge d$, and y_i are responses such that $y_i=f(\boldsymbol{x}_i)+\varepsilon_i$ for centered, independent random noise ε_i. Assume $|\varepsilon_i|$ and $|f(\cdot)|$ both are bounded by M. Furthermore, f is convex and Lipschitz. We propose to solve the penalized convex regression problem in Eq. (2) with the penalty term $\|f\|\triangleq\sum_{l=1}^d\sup_{\boldsymbol{x}}|\partial^*_{x_l} f(\boldsymbol{x})|$. This results in the following convex optimization problem:

$$\min_{\hat{y}_i,\boldsymbol{a}_i}\frac{1}{n}\sum_{i=1}^n(\hat{y}_i-y_i)^2+\lambda\sum_{l=1}^d\max_{i=1}^n|a_{i,l}| \tag{3}$$
$$\text{s.t. }\hat{y}_i-\hat{y}_j-\langle\boldsymbol{a}_i,\boldsymbol{x}_i-\boldsymbol{x}_j\rangle\le 0\quad i,j\in[n]\times[n].$$

Then, we estimate $f(\boldsymbol{x})$ via

$$\hat{f}(\boldsymbol{x})\triangleq\max_i\langle\boldsymbol{a}_i,\boldsymbol{x}-\boldsymbol{x}_i\rangle+\hat{y}_i. \tag{4}$$

This is a model similar to Balázs (2016) with a minor modification of using a different penalty term in the loss function. Our penalty term depends on the sum of partial derivatives $\sum_{l=1}^d\sup_{\boldsymbol{x}}|\partial_{x_l} f(\boldsymbol{x})|$ rather than $\sup_{\boldsymbol{x}}\|\nabla f(\boldsymbol{x})\|_2^2$. This new penalty term acts similar to a $L1$ regularizer and encourages feature sparsity. It is easy to show the estimator is bounded i.e., $\sup_{\boldsymbol{x}\in\Omega}|\hat{f}(\boldsymbol{x})|\le M+4\|\hat{f}\|R$. Also $\|f+g\|\le\|f\|+\|g\|$, $\|fc\|=c\|f\|$ for $c\ge 0$. Hence, similar to Siahkamari et al. (2020), our penalty term is a valid seminorm which allows us to use their theorem here.

Proposition 1. *With the appropriate choice of λ which requires knowledge of M the bound on f and $n \geq d$, it holds that with probability at least $1-\delta$ over the data, the estimator $\hat{f}$ of (4) has excess risk upper bounded by*

$$\mathbb{E}[|f(\boldsymbol{x}) - \hat{f}(\boldsymbol{x})|^2] \leq O\left(\left(\frac{n}{d}\right)^{\frac{-2}{d+4}} \log\left(\frac{n}{d}\right) + \sqrt{\frac{\log(1/\delta)}{n}}\right).$$

2.1. Optimization

Our method utilizes the well-known ADMM (Gabay & Mercier, 1976) algorithm. ADMM is a standard tool for solving convex problems that consists of variable blocks with linear constraints (Eckstein & Yao, 2012). It has an iterative procedure to update the problem variables with provable convergence guarantees (He & Yuan, 2012).

We solve program (3) using ADMM; the main insight is that we can solve a 2-block ADMM formulation where each block can be computed in closed form. We first consider an equivalent form of the optimization problem as:

$$\min_{\substack{\hat{y}_i, \boldsymbol{a}_i, L_l \geq 0, \\ \boldsymbol{p}_i^+ \geq 0, \boldsymbol{p}_i^- \geq 0, \boldsymbol{u}_i \geq 0, s_{i,j} \geq 0}} \sum_{i=1}^n (\hat{y}_i - y_i)^2 + \lambda \sum_{l=1}^d L_l \tag{5}$$

$$\text{s.t.} \begin{cases} s_{i,j} + \hat{y}_i - \hat{y}_j - \langle \boldsymbol{a}_i, \boldsymbol{x}_i - \boldsymbol{x}_j \rangle = 0 & i,j \in [n] \times [n] \\ u_{i,l} + p_{i,l}^+ + p_{i,l}^- - L_l = 0 & i,l \in [n] \times [d] \\ a_{i,l} - p_{i,l}^+ + p_{i,l}^- = 0 & i,l \in [n] \times [d], \end{cases}$$

with the augmented Lagrangian

$$\begin{aligned} \ell(\hat{y}_i, \boldsymbol{a}_i, L_l, \boldsymbol{p}_i^+, \boldsymbol{p}_i^-, \boldsymbol{u}_i, s_{i,j}) &= \frac{1}{n} \sum_{i=1}^n (\hat{y}_i - y_i)^2 + \lambda \sum_{l=1}^d L_l \\ &+ \sum_i \sum_j \frac{\rho}{2} (s_{i,j} + \hat{y}_i - \hat{y}_j - \langle \boldsymbol{a}_i, \boldsymbol{x}_i - \boldsymbol{x}_j \rangle + \alpha_{i,j})^2 \\ &+ \sum_i \sum_l \frac{\rho}{2} (u_{i,l} + p_{i,l}^+ + p_{i,l}^- - L_l + \gamma_{i,l})^2 \\ &+ \sum_i \sum_l \frac{\rho}{2} (a_{i,l} - p_{i,l}^+ + p_{i,l}^- + \eta_{i,l})^2, \end{aligned}$$

where $\alpha_{i,j}$, $\gamma_{i,l}$ and $\eta_{i,j}$ are dual variables. We divide parameters into two blocks as $\boldsymbol{b}^1 = \{\hat{y}_i, \boldsymbol{a}_i\}$ and $\boldsymbol{b}^2 = \{L_l, \boldsymbol{p}_i^+, \boldsymbol{p}_i^-, \boldsymbol{u}_i, s_{i,j}\}$, where $i,j \in [n] \times [n], l \in [d]$. We find closed form solutions for each block given the solution to the other block.

2.1.1. First block $\boldsymbol{b}^1 = \{\hat{y}_i, \boldsymbol{a}_i\}$

We first note that we always normalize the dataset such that $\sum_i \boldsymbol{x}_i = \boldsymbol{0}$ and $\sum_i y_i = 0$. This will result in $\sum_i \hat{y}_i = 0$ and simplify the solutions.

By setting $\nabla_{\boldsymbol{a}_i} \ell = \boldsymbol{0}$ we can solve for $\boldsymbol{a}_i$ as:

$$\boldsymbol{a}_i = \boldsymbol{\Lambda}_i (\boldsymbol{\theta}_i + \hat{y}_i \boldsymbol{x}_i + \frac{1}{n} \sum_k \hat{y}_k \boldsymbol{x}_k), \tag{6}$$

where

$$\boldsymbol{\Lambda}_i \triangleq (\boldsymbol{x}_i \boldsymbol{x}_i^T + \frac{1}{n} I + \frac{1}{n} \sum_j \boldsymbol{x}_j \boldsymbol{x}_j^T)^{-1},$$

$$\boldsymbol{\theta}_i \triangleq \frac{1}{n} \left(\boldsymbol{p}_i^+ - \boldsymbol{p}_i^- - \boldsymbol{\eta}_i + \sum_j (\alpha_{i,j} + s_{i,j})(\boldsymbol{x}_i - \boldsymbol{x}_j) \right).$$

Similarly by setting $\partial_{\hat{y}_i} \ell = 0$ and substituting Eq. (6) for $\boldsymbol{a}_i$ we can solve for $\hat{y}_1, \ldots, \hat{y}_n$, simultaneously as a system of linear equations,

$$\hat{\boldsymbol{y}} = \boldsymbol{\Omega}^{-1} \left(\frac{2\boldsymbol{y}}{n^2 \rho} + \boldsymbol{v} - \boldsymbol{\beta} \right) \tag{7}$$

where $\boldsymbol{y} = [y_1, \ldots, y_n]^T$, $\hat{\boldsymbol{y}} = [\hat{y}_1, \ldots, \hat{y}_n]^T$, and

$$\beta_i \triangleq \frac{1}{n} \sum_j \alpha_{i,j} - \alpha_{j,i} + s_{i,j} - s_{j,i},$$

$$v_i \triangleq \boldsymbol{x}_i^T \boldsymbol{\Lambda}_i \boldsymbol{\theta}_i + \boldsymbol{x}_i^T \frac{1}{n} \sum_j \boldsymbol{\Lambda}_j \boldsymbol{\theta}_j - \frac{1}{n} \sum_j \boldsymbol{x}_j^T \boldsymbol{\Lambda}_j \boldsymbol{\theta}_j$$

$$\Omega_{i,j} \triangleq \left(\frac{2}{n^2 \rho} + 2 - \boldsymbol{x}_i^T \boldsymbol{\Lambda}_i \boldsymbol{x}_i \right) \mathbb{1}(i = j) - \frac{1}{n} D_{i,j},$$

$$\begin{aligned} D_{i,j} \triangleq\ & \boldsymbol{x}_i^T \left(\boldsymbol{\Lambda}_i + \boldsymbol{\Lambda}_j + \frac{1}{n} \sum_k \boldsymbol{\Lambda}_k \right) \boldsymbol{x}_j - \boldsymbol{x}_j^T \boldsymbol{\Lambda}_j \boldsymbol{x}_j \\ & - \frac{1}{n} \sum_k \boldsymbol{x}_k \boldsymbol{\Lambda}_k \boldsymbol{x}_j. \end{aligned}$$

2.1.2. Second block $\boldsymbol{b}^2 = \{L_l, \boldsymbol{p}_i^+, \boldsymbol{p}_i^-, \boldsymbol{u}_i, \}$

Set $\partial_s \ell = 0$ for $s \in \{s_{i,j}, \boldsymbol{p}_i^+, \boldsymbol{p}_i^-, \boldsymbol{u}_i, \}$. Hence

$$\begin{aligned} s_{i,j} &= (-\alpha_{i,j} - \hat{y}_i + \hat{y}_j + \langle \boldsymbol{a}_i, \boldsymbol{x}_i - \boldsymbol{x}_j \rangle)^+, \\ u_{i,l} &= (L_l - \gamma_{i,l} - |\eta_{i,l} + a_{i,l}|)^+, \\ p_{i,l}^+ &= \frac{1}{2}(L_l - \gamma_{i,l} - u_{i,l} + \eta_{i,l} + a_{i,l})^+, \\ p_{i,l}^- &= \frac{1}{2}(L_l - \gamma_{i,l} - u_{i,l} - \eta_{i,l} - a_{i,l})^-. \end{aligned} \tag{8}$$

Lastly set $\partial_{L_l} \ell = 0$ and plug the solutions for $u_{i,l}$, $p_{i,l}^+$ and $p_{i,l}^-$. Denote $c_{i,l} \triangleq |\eta_{i,l} + a_{i,l}|$, after rearrangement of the terms we have:

$$\frac{\lambda}{\rho} = \sum_i \begin{cases} 0 & \text{if } L_l - \gamma_{i,l} \geq c_{i,l} \\ \frac{1}{2}(\gamma_{i,l} + c_{i,l} - L_l) & \text{if } |L_l - \gamma_{i,l}| \leq c_{i,l} \\ \gamma_{i,l} - L_l & \text{if } L_l - \gamma_{i,l} \leq -c_{i,l}. \end{cases}$$

Note that the right hand side is a monotonic and piecewise linear function of L_l. It is easy to find the solution to this

Algorithm 1 **L-update**

```
Require: ${\gamma_i, c_i}_{i=1}^n$, and $\rho/\lambda$
 1: $knot_{2n}, \ldots, knot_1 \leftarrow sort\{\gamma_i + c_i, \gamma_i - c_i\}_{i=1}^n$
 2: $f \leftarrow \lambda/\rho$
 3: $f' \leftarrow 0$
 4: for $j = 2$ to $2n$ do
 5:     $f' \leftarrow f' + \frac{1}{2}$
 6:     $f \leftarrow f + f' \cdot (knot_j - knot_{j-1})$
 7:     if $f \leq 0$ then
 8:         return $\left(knot_j - \frac{f}{f'}\right)^+$
 9:     end if
10: end for
11: return $\left(knot_{2n} - \frac{f}{n}\right)^+$
```

problem with respect to L_l, using a sort and a simple algorithm that takes $O(n \log n)$ flops; see **L_update** Algorithm 1. Observe that it is possible to add monotonicity constraints for $\hat{f}$ by projecting either $p_{i,l}^+$ or $p_{i,l}^-$ to zero.

2.1.3. Dual variables

The update for dual variables follows from the standard ADMM algorithm updates:

$$\begin{aligned} \alpha_{i,j} &= \alpha_{i,j} + s_{i,j} && (9) \\ &+ \hat{y}_i - \hat{y}_j - \langle \boldsymbol{a}_i, \boldsymbol{x}_i - \boldsymbol{x}_j \rangle && i,j \in [n] \times [n] \\ \gamma_{i,l} &= \gamma_{i,l} + u_{i,l} + p_{i,l}^+ + p_{i,l}^- - L_l && i,l \in [n] \times [d] \\ \eta_{i,l} &= \eta_{i,l} + a_{i,l} - p_{i,l}^+ + p_{i,l}^- && i,l \in [n] \times [d]. \end{aligned}$$

2.1.4. Algorithms via ADMM

Algorithm 2 provides the full steps for a parallel ADMM optimizer for the convex regression problem. In each line in Algorithm 2, we use subscripts such as i, j and l on the left hand side. These updates may be run in parallel for $i \in [n]$, $i, j \in [n] \times [n]$, $i, l \in [n] \times [d]$ or $q \in [2]$. We initially set all block variables to zero and normalize the dataset such that $\sum_i y_i = 0$ and $\sum_i \boldsymbol{x}_i = 0$. We have implemented our algorithm using *PyTorch* (Paszke et al., 2019), which benefits from this parallel structure when using a GPU. Our code, along with a built-in tuner for our hyperparameter λ and T, is available on our GitHub repository [1].

2.2. Analysis

Our method has two sources of errors which are the error due to the ADMM procedure (Eq. 5) and the error due to estimating the ground truth convex function (Eq. 4). We characterize both errors based on ADMM convergence (He & Yuan, 2012).

Theorem 1. *[from He & Yuan (2012)] Consider the separable convex optimization problem,*

[1]https://github.com/Siahkamari/Piecewise-linear-regression

Algorithm 2 Convex regression

```
Require: $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$, $\rho$, $\lambda$, and $T$
 1: $\hat{y}_i = s_{i,j} = \alpha_{i,j} \leftarrow 0$
 2: $\boldsymbol{L} = \boldsymbol{a}_i = \boldsymbol{p}_i = \boldsymbol{u}_i = \boldsymbol{\eta}_i = \boldsymbol{\gamma}_i \leftarrow \boldsymbol{0}_{d\times 1}$
 3: for $t = 1$ to $T$ do
 4:     Update $\hat{\boldsymbol{y}}$ by Eq. (7)
 5:     Update $\boldsymbol{a}_i$ by Eq. (6)
 6:     $L_l \leftarrow$ L_update$(\{\gamma_{i,l}, |\eta_{i,l} + a_{i,l}|\}_{i\in[n]}, \lambda/\rho)$
 7:     Update $u_{i,l}, p_{i,l}^+, p_{i,l}^-, s_{i,j}$ by Eq. (8)
 8:     Update $\alpha_{i,j}, \gamma_{i,l}, \eta_{i,l}$ by Eq. (9)
 9: end for
10: return $f(\cdot) \triangleq \max_{i=1}^n(\langle \boldsymbol{a}_i, \cdot - \boldsymbol{x}_i \rangle + \hat{y}_i)$
```

$$\min_{\mathbf{b}^1 \in \mathcal{S}_1, \mathbf{b}^2 \in \mathcal{S}_2} \left[\psi(\mathbf{b}^1, \mathbf{b}^2) = \psi_1(\mathbf{b}^1) + \psi_2(\mathbf{b}^2)\right]$$
$$s.t: \mathbf{A}\mathbf{b}^1 + \mathbf{B}\mathbf{b}^2 + \boldsymbol{b} = \mathbf{0},$$

where $(\mathbf{b}^1, \mathbf{b}^2)$ *are the block variables,* $(\mathcal{S}_1, \mathcal{S}_2)$ *are convex sets that includes all zero vectors,* $(\mathbf{A}, \mathbf{B})$ *are the coefficient matrices and* $\mathbf{b}$ *is a constant vector. Let* $\mathbf{b}_t^1$ *and* $\mathbf{b}_t^2$ *be solutions at iteration* t *of a two block ADMM procedure with learning rate* ρ *where* $(\mathbf{b}_0^1, \mathbf{b}_0^2)$ *are all zero vectors. Denote average of iterates as* $(\tilde{\mathbf{b}}_T^1, \tilde{\mathbf{b}}_T^2) = \left(\frac{1}{T}\sum_{t=1}^T \mathbf{b}_t^1, \frac{1}{T}\sum_{t=1}^T \mathbf{b}_t^2\right)$*. For all* $\boldsymbol{\kappa}$ *we have,*

$$\psi(\tilde{\mathbf{b}}_T^1, \tilde{\mathbf{b}}_T^2) - \psi(\mathbf{b}_*^1, \mathbf{b}_*^2) - \boldsymbol{\kappa}^T(\mathbf{A}\tilde{\mathbf{b}}_T^1 + \mathbf{B}\tilde{\mathbf{b}}_T^2 + \boldsymbol{b})$$
$$\leq \frac{1}{T}\left(\frac{\rho}{2}\|\mathbf{B}\mathbf{b}_*^2\|^2 + \frac{1}{2\rho}\|\boldsymbol{\kappa}\|^2\right),$$

where $(\mathbf{b}_1^*, \mathbf{b}_2^*)$ *are the optimal solutions.*

We arranged Theorem 1 such that it explicitly depends on the problem dependent constants such as $\|\mathbf{B}\mathbf{b}_*^2\|^2$, the number of iterations, T, as well as the learning rate ρ. A proof is provided in Appendix B.1.

Next, we present convergence analysis in terms of regularized MSE of the output of the ADMM algorithm (2). This has the further benefit of finding an appropriate learning rate ρ and a range for regularization coefficient λ that minimizes the computational complexity.

Theorem 2. *Let* $\{\hat{y}_i^t, \boldsymbol{a}_i^t\}_{i=1}^n$ *be the output of Algorithm 2 at the* t^{th} *iteration,* $\tilde{y}_i \triangleq \frac{1}{T}\sum_{t=1}^T \hat{y}_i^t$ *and* $\tilde{\boldsymbol{a}}_i \triangleq \frac{1}{T}\sum_{t=1}^T \boldsymbol{a}_i^t$*. Denote* $\tilde{f}_T(\boldsymbol{x}) \triangleq \max_i \langle \tilde{\boldsymbol{a}}_i, \boldsymbol{x} - \boldsymbol{x}_i \rangle + \tilde{y}_i$*. Assume* $\max_{i,l}|x_{i,l}| \leq 1$ *and* $\mathbb{V}\mathrm{ar}(\{y_i\}_{i=1}^n) \leq 1$*. If we choose* $\rho = \frac{\sqrt{d}\lambda^2}{n}$*, for* $\lambda \geq \frac{3}{\sqrt{2nd}}$ *and* $T \geq n\sqrt{d}$ *we have:*

$$\frac{1}{n}\sum_{i=1}^n (\tilde{f}_T(\boldsymbol{x}_i) - y_i)^2 + \lambda\|\tilde{f}_T\|$$
$$\leq \min_{\hat{f} \in \mathcal{F}}\left(\frac{1}{n}\sum_{i=1}^n \left(\hat{f}(\boldsymbol{x}_i) - y_i\right)^2 + \lambda\|\hat{f}\|\right) + \frac{6n\sqrt{d}}{T+1}.$$

Corollary 1. *Our method needs $T = \frac{6n\sqrt{d}}{\epsilon}$ iterations to achieve ϵ error. Each iteration requires $\mathcal{O}(n^2 d + nd^2)$ flops operations. Prepossessing costs $\mathcal{O}(nd^3)$. Therefore the total computational complexity is $\mathcal{O}\left(\frac{n^3 d^{1.5} + n^2 d^{2.5} + nd^3}{\epsilon}\right)$.*

We note that assumptions of Theorem 2 are simply satisfied by normalizing the training dataset. The main difficulty for the derivation of Theorem 2 is: $(\tilde{\boldsymbol{a}}_i, \tilde{y}_i)$ might be violating the constraints in (5). This could result in $\tilde{f}_T(\boldsymbol{x}_i) \neq \tilde{y}_i$ and $\|\tilde{f}_T\| \neq \sum_{l=1}^{d} \tilde{L}_l$. Therefore, the main steps of the proof of Theorem 2 is to characterize and bound the effect of such constraint violations on our objective function. Then we set $\boldsymbol{\kappa}$ in Theorem 1 in a way to cover for the effects of constraint violations. For a full proof, we refer to Appendix B.2.

3. Approximating a Bregman Divergence

We next consider the application of learning a Bregman divergence from supervision. This problem is a type of metric learning problem, where we are given supervision (pairs of similar/dissimilar pairs, or triplets consisting of relative comparisons amongst data points) and we aim to learn a task-specific distance or divergence measure from the data. Classical metric learning methods typically learn a linear transformation of the data, corresponding to a Mahalanobis-type distance function,

$$(\boldsymbol{x} - \boldsymbol{y})^T M (\boldsymbol{x} - \boldsymbol{y}),$$

where M is a positive semi-definite matrix (note that this generalizes the squared Euclidean distance, where M would be the identity matrix). More generally, Mahalanobis distances are examples of Bregman divergences, which include other divergences such as the KL-divergence as special cases.

Bregman divergences are parameterized by an underlying strictly convex function, sometimes called the *convex generating function* of the divergence. Recently, Siahkamari et al. (2019) formulate learning a Bregman divergence as learning the underlying convex function parameterizing the divergence. The resulting optimization problem they study is similar to convex regression problem Eq. (5). Here we will discuss an improvement to their approach with a slightly different loss function and our Lasso penalty term. We then derive an algorithm using 2-block ADMM; the existing solver for this problem uses standard linear programming.

In particular suppose we observe the classification dataset $S_n = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$, for $\boldsymbol{x}_i \in \mathbb{R}^d$ and y_i is an integer. Let

$$D_f(\boldsymbol{x}_i, \boldsymbol{x}_j) = f(\boldsymbol{x}_i) - f(\boldsymbol{x}_j) - \langle \nabla^* f(\boldsymbol{x}_j), \boldsymbol{x}_i - \boldsymbol{x}_j \rangle,$$

be a Bregman divergence with convex generating function $f(\boldsymbol{x}) \in \mathcal{F}$. Let the pairwise similarity loss be

$$\ell(D_f(\boldsymbol{x}_i, \boldsymbol{x}_j), y_i, y_j) = \mathbb{1}\big[\mathbb{1}[y_i = y_j] = (D_f(\boldsymbol{x}_i, \boldsymbol{x}_j) \geq 1)\big].$$

We estimate f the underlying convex function of the Bregman divergence D_f by,

$$\hat{f} \triangleq \arg\min_{f \in \mathcal{F}} \sum_{i=1}^{n} \sum_{j=1}^{n} \ell(D_f(\boldsymbol{x}_i, \boldsymbol{x}_j), y_i, y_j) + \lambda \|f\|.$$

We note that this approach can easily be generalized to other possible metric learning losses.

3.1. Optimization

Now we provide an algorithm for the Bregman divergence learning based on 2-block ADMM. We solve for each block in closed form. We first re-formulate the optimization problem to standard form and introducing some auxiliary variables we have:

$$\min_{\substack{z_i, \boldsymbol{a}_i, \zeta_{i,j} \geq 0, \\ L_l \geq 0, \boldsymbol{p}_i^+ \geq 0, \boldsymbol{p}_i^+ \geq 0, \\ \boldsymbol{u}_i \geq 0, s_{i,j} \geq 0, t_{i,j} \geq 0}} \sum_{i=1}^{n} \sum_{j=1}^{n} \zeta_{i,j} + \lambda \sum_{d} L_l \tag{10}$$

$$\text{s.t.} \begin{cases} \iota_{i,j} s_{i,j} - \iota_{i,j} + t_{i,j} + 1 - \zeta_{i,j} = 0 & i, j \in [n] \times [n] \\ s_{i,j} + z_i - z_j - \langle \boldsymbol{a}_i, \boldsymbol{x}_i - \boldsymbol{x}_j \rangle = 0 & i, j \in [n] \times [n] \\ p_{i,l}^+ + p_{i,l}^- + u_{i,l} - L_l = 0 & i, l \in [n] \times [d] \\ a_{i,l} = p_{i,l}^+ - p_{i,l}^- & i, l \in [n] \times [d], \end{cases}$$

where $\iota_{i,j} = 2(\mathbb{1}[y_i = y_j] - \frac{1}{2})$.

Now we write the augmented Lagrangian:

$$\begin{aligned} &\ell(z_i, \boldsymbol{a}_i, \zeta_{i,j}, L_l, \boldsymbol{p}_i^+, \boldsymbol{p}_i^-, \boldsymbol{u}_i, s_{i,j}, t_{i,j}) \\ &= \sum_{i=1}^{n} \sum_{j=1}^{n} \zeta_{i,j} + \lambda \sum_{l=1}^{d} L_l \\ &+ \sum_{i} \sum_{j} \frac{\rho}{2} (s_{i,j} + z_i - z_j - \langle \boldsymbol{a}_i, \boldsymbol{x}_i - \boldsymbol{x}_j \rangle + \alpha_{i,j})^2 \\ &+ \sum_{i} \sum_{l} \frac{\rho}{2} (u_{i,l} + p_{i,l}^+ + p_{i,l}^- - L_l + \gamma_{i,l})^2 \\ &+ \sum_{i} \sum_{l} \frac{\rho}{2} (a_{i,l} - p_{i,l}^+ + p_{i,l}^- + \eta_{i,l})^2 \\ &+ \sum_{i} \sum_{j} \frac{\rho}{2} (\iota_{i,j} s_{i,j} - \iota_{i,j} + t_{i,j} + 1 - \zeta_{i,j} + \tau_{i,j})^2, \end{aligned}$$

where $\alpha_{i,j}, \gamma_{i,j}, \eta_{i,j}$ and $\tau_{i,j}$ are dual variables. Next we divide the variables into two blocks and solve for each in closed form.

3.1.1. First block $\boldsymbol{b}^1 = \{z_i, \boldsymbol{a}_i, \zeta_{i,j}\}$

Setting $\partial_{\zeta_{i,j}} \ell = 0$ gives:

$$\zeta_{i,j} = (\frac{-1}{n\rho} + \tau_{i,j} + \iota_{i,j} s_{i,j} - \iota_{i,j} + t_{i,j} + 1)^+. \quad (11)$$

Collecting the terms containing $\boldsymbol{a}_i$ in Eq.(10) and comparing to those in Eq. (5), the solution for $\boldsymbol{a}_i$ follows:

$$\boldsymbol{a}_i \triangleq \boldsymbol{\Lambda}_i(\boldsymbol{\theta}_i + z_i \boldsymbol{x}_i + \frac{1}{n}\sum_k z_k \boldsymbol{x}_k). \quad (12)$$

Set $\partial_{z_i} \ell = 0$, $\sum_i z_i = 0$ and $\sum_i \boldsymbol{x}_i = \boldsymbol{0}$. Using Eq. (12) we can solve for $z_1, \ldots, z_n$ as

$$\boldsymbol{z} = \boldsymbol{\Omega}_{\text{breg}}^{-1}(\boldsymbol{\nu} - \boldsymbol{\beta}), \quad (13)$$

where $\Omega_{\text{breg}_{i,j}} = (2 - \boldsymbol{x}_i^T \Lambda_i \boldsymbol{x}_i)\mathbb{1}[i = j] - \frac{1}{n} D_{i,j}$.

3.1.2. Second block $\boldsymbol{b}^2 = \{L_l, \boldsymbol{p}_i, \boldsymbol{u}_i, s_{i,j}, t_{i,j}\}$

Comparing Eq. (10) and Eq. (5), we find that L_l, $\boldsymbol{p}_i$ and $\boldsymbol{u}_i$, have the same solutions as in sec 2.1.2. Hence we only proceed to solve for $s_{i,j}$ and $t_{i,j}$. Set $\partial_{t_{i,j}} \ell = \partial_{t_{i,j}} = 0$. Some algebra gives:

$$s_{i,j} = \frac{1}{2}(\pi_{i,j}^2 + \iota_{i,j}\pi_{i,j}^1 - \iota_{i,j}(\pi_{i,j}^1 - \iota_{i,j}\pi_{i,j}^2)^+)^+, \quad (14)$$
$$t_{i,j} = (\pi_{i,j}^1 - \iota_{i,j} s_{i,j})^+,$$

where

$$\pi_{i,j}^1 \triangleq -\tau_{i,j} + \iota_{i,j} - 1 + \zeta_{i,j},$$
$$\pi_{i,j}^2 \triangleq -\alpha_{i,j} - z_i + z_j + \langle \boldsymbol{a}_i, \boldsymbol{x}_i - \boldsymbol{x}_j \rangle.$$

3.1.3. Dual variables

We only see a new constraint different from Eq. (5) with dual multiplier $\tau_{i,j}$. The update is

$$\tau_{i,j} = \tau_{i,j} + \iota_{i,j} s_{i,j} - \iota_{i,j} + t_{i,j} + 1 - \zeta_{i,j}. \quad (15)$$

Algorithm 3 in the appendix provides full-steps for solving the Bregman divergence learning problem.

4. Difference of Convex (DC) Regression

As a further example, we extend our convex regression solver to the difference of convex regression as studied in Siahkamari et al. (2020). DC functions are set of functions f that can be represented as $f = \phi^1 - \phi^2$ for a choice of two *convex* functions. DC functions are a very rich class—for instance, they are known to contain all $\mathcal{C}^2$ functions. DC regression has been studied in Cui et al. (2018); Siahkamari et al. (2020); Bagirov et al. (2020). We provide a 2-block ADMM solver for the difference of convex estimator proposed in Siahkamari et al. (2020), with a minor change of choosing a different penalty

$$\|f\| \triangleq \inf_{\phi^1,\phi^2} \sum_{q=1}^{2}\sum_{l=1}^{d} \sup_{\boldsymbol{x}} |\partial_{x_l}^* \phi^q(\boldsymbol{x})|$$
$$\text{s.t. } \phi_1, \phi_2 \text{ are convex, } \phi_1 - \phi_2 = f.$$

In particular, consider a regression dataset $\{(y_i, \boldsymbol{x}_i)\}_{i=1}^n$ with $\boldsymbol{x}_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$. Denote the class of difference of convex functions

$$\mathcal{DC} \triangleq \{f : \mathbb{R}^d \to \mathbb{R} \mid f = \phi^1 - \phi^2, (\phi^1, \phi^2) \text{ are convex}\},$$

In order to estimate f, we minimize the penalized least square over the $\mathcal{DC}$ class:

$$\hat{f} \triangleq \arg\min_{f \in \mathcal{DC}} \frac{1}{n}\sum_{i=1}^{n}(y_i - f(\boldsymbol{x}_i))^2 + \lambda\|f\|.$$

This results in a convex program:

$$\min_{\hat{y}_i^q, \boldsymbol{a}_i^q} \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i^1 - \hat{y}_i^2 - y_i)^2 + \lambda \sum_{q=1}^{2}\sum_{l=1}^{d} \max_{i=1}^{n} |a_{i,l}^q|$$
$$\text{s.t.} \begin{cases} \hat{y}_i^1 - \hat{y}_j^1 - \langle \boldsymbol{a}_i^1, \boldsymbol{x}_i - \boldsymbol{x}_j \rangle \le 0 & i,j \in [n]\times[n], \\ \hat{y}_i^2 - \hat{y}_j^2 - \langle \boldsymbol{a}_i^2, \boldsymbol{x}_i - \boldsymbol{x}_j \rangle \le 0 & i,j \in [n]\times[n], \end{cases} \quad (16)$$

and the estimator for f is given as

$$\hat{f}(\boldsymbol{x}) \triangleq \max_i \langle \boldsymbol{a}_i^1, \boldsymbol{x} - \boldsymbol{x}_i \rangle + \hat{y}_i^1 - \max_i \langle \boldsymbol{a}_i^2, \boldsymbol{x} - \boldsymbol{x}_i \rangle + \hat{y}_i^2.$$

We note that the linear constraint sets for $\{\hat{y}_i^1, \boldsymbol{a}_i^1\}$ and $\{\hat{y}_i^2, \boldsymbol{a}_i^2\}$ have no variable in common in the linear program (16). Fixing $\hat{\boldsymbol{y}}^2$ allows us to solve for $\hat{\boldsymbol{y}}^1$ using Eq. (7) and vice-versa. From there we can decouple $\hat{\boldsymbol{y}}^1$ and $\hat{\boldsymbol{y}}^2$ solutions by solving a system of linear equations,

$$\hat{\boldsymbol{y}}^q = \frac{(-1)^{q+1}}{2}\left(\boldsymbol{\Omega} + \frac{2I}{n^2\rho}\right)^{-1}\left(\frac{4\boldsymbol{y}}{n^2\rho} + v^1 - \boldsymbol{\beta}^1 - \boldsymbol{v}^2 + \boldsymbol{\beta}^2\right) + \frac{1}{2}\left(\boldsymbol{\Omega} - \frac{2\boldsymbol{I}}{n^2\rho}\right)^{-1}(\boldsymbol{v}^1 - \boldsymbol{\beta}^1 + \boldsymbol{v}^2 - \boldsymbol{\beta}^2). \quad (17)$$

Algorithm 4 in Appendix provides the full steps to solve the difference of convex regression problem.

5. Experiments

In this section we provide experiments on real datasets from the UCI machine learning repository as well as on synthetic datasets. We compare running times between our proposed approach and the baseline interior point method for all three

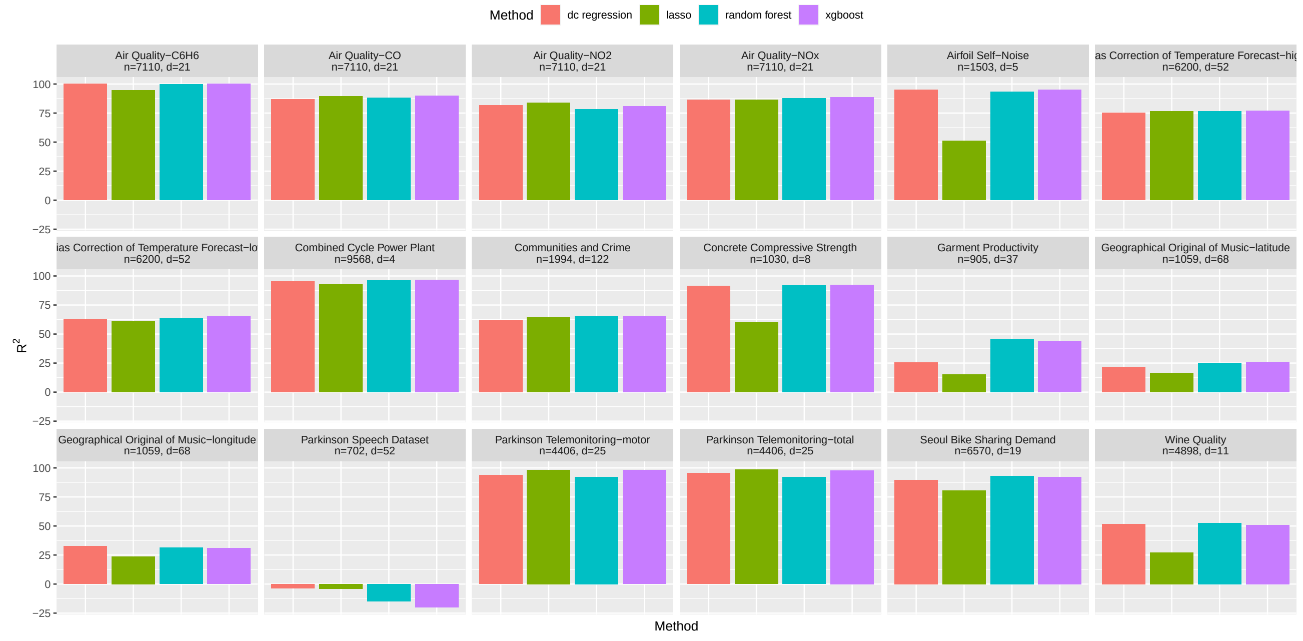

Figure 1. Regression results on UCI datasets

Table 1. Comparison of Convex Regression run time against baseline on Synthetic Data

		Seconds	
n	d	Baseline (Siahkamari et al., 2020)	This Paper
1000	2	32.3	3.63
1000	4	30.8	3.75
1000	8	54.5	4.03
1000	16	112.3	4.12
1000	32	189.3	4.37

Table 2. Comparison of DC Regression run time against baseline on Synthetic Data

		Seconds	
n	d	Baseline (Siahkamari et al., 2020)	This Paper
1000	2	140.9	3.67
1000	4	113.9	3.79
1000	8	210.3	4.05
1000	16	351.6	4.15
1000	32	823.2	4.42

problems (convex regression, DC regression, and Bregman divergence learning). We also compare both our DC regression algorithm as well as our Bregman divergence learning algorithm to state-of-the-art regression and classification methods, and show that our approach is close to state-of-the-art in terms of accuracy. We compare our (DC)-regression algorithm with state-of-the-art regression models.

For the ADMM-based methods, we use a V100 Nvidia GPU processor with 11 gigabyte of GPU memory, and 4 cores of CPU. For all the other methods we use a 16 core CPU.

5.1. Timing Results for Convex and DC Regression

To demonstrate the effectiveness of our proposed approach, we first compare on synthetic data the existing interior point method to our 2-block ADMM method on standard convex regression and DC regression. Here, we fixed the number of data points at 1000 and tested data of dimensionality $d \in \{2, 4, 8, 12, 32\}$.

Results are shown in Tables 1 and 2. We see that our method ranges from 8.8x faster than the interior point method to up to 186x faster. Note that the interior point method fails on large data sets, and cannot be run for many of the UCI data sets tested later.

5.2. Results on Real Data

Now we experiment with our DC regression algorithm and the Bregman divergence learning algorithm (PBDL) on real data sets, and report accuracy and timing results. For PBDL, we compare against its predecessor (PBDL_0) as well as state of the art classification models. All results are reported either on the pre-specifed train/test split or a 5-fold cross validation set based on instruction for each dataset.

For the purpose of comparisons, we compare to the previous (DC)-regression and PBDL algorithms. Furthermore, we pick two popular tree-based algorithms XGboost and Random Forest. We also consider Lasso as a baseline. According to Fernández-Delgado et al. (2014; 2019), Random

Forest achieves the state of the art on most regression and classification tasks. More recently, XGboost has been shown to achive the state of the art on many real datasets and *Kaggle.com* competitions (Chen & Guestrin, 2016). However, we stress that achieving state-of-the-art results is not the purpose of our experiments.

We provide the details of hyper-parameters for each method.

DC-regression: We choose λ from a grid $10^{-3:3}$ by 5 fold cross validation. Then we do at most 2 more rounds of grid search around the optimal λ at the first round. We fix $\rho = 0.01$ and choose T by early stopping, i.e., whenever validation error improvement is less than 10^{-3} after n iterations of ADMM.

PBDL: We predict the classes using a 5 nearest neighbour scheme $i_{\text{nearest}}(\boldsymbol{x}) = \arg\min_i D_\phi(\boldsymbol{x}, \boldsymbol{x}_i)$. where D_ϕ is learned Bregman divergence. Tuning λ is similar to DC-regression.

PBDL_0: We use the existing code from the authors for learning a Bregman divergence based on Groubi solvers. We use their built-in parameter tuner for λ which is based on 5 fold cross validation.

XGboost Regressor/Classifier: max_depth and $learning\_rate$ is found by 5 fold cross validation over $\{1, \ldots, 10\} \times \{0.05, 0.1, 0.5\}$. Other parameters set to default. We use the xgboost 1.6.0 package in Chen & Guestrin (2016).

Random Forest Regressor/Classifier: $n\_features$ is found by 5 fold cross validation over $[d^{0.25}, d^{0.75}]$. We use the sklearn package in Pedregosa et al. (2011).

Lasso: We use the sklearn package function LassoCV.

5.2.1. Datasets

We chose all regression datasets from UCI which have number of instances $10^3 \leq n \leq 10^4$. These were 30 datasets at the time of submission of this paper. We discard 16 of these datasets due to various reasons such as the data set not being available. Some of these datasets have multiple target variables, therefore we report the out of sample R^2 results with a suffix for each target variable.

For classification, we chose all datasets originally used in Siahkamari et al. (2019). Further, we include the *abalone* dataset which was too large for the previous PBDL algorithm. We use these datasets as multi-class classification problems. We report accuracy as well as the training runtime.

5.2.2. Results

Regression: For our regression experiments we present the out of sample R^2 on 18 datasets in figure 1 . We observe our method has similar performance to that of XGboost and Random Forest despite having $n \times d$ parameters. In solar-flare and Parkinson datasets, other methods overfit whereas our algorithm avoids overfitting and it is more robust. Maximum number of instances which we were able to experiment on with reasonable time is about 10^4 and is 10x larger than Siahkamari et al. (2020) has experimented on. The detailed results are in Appendix.

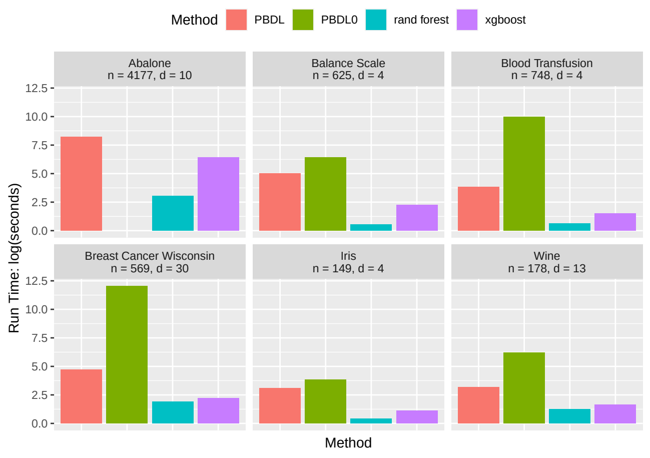

Figure 2. Classification log(runtime) for all fits on UCI datasets

Classification: For our classification experiments we compare the logarithm of runtimes of all methods in Figure 2. In terms of speed, we are on average 30x faster than the original PBDL_0 algorithm. Also PBDL_0 fails to handle the Abalone dataset with $n = 4177$, where the new PBDL takes less than a minute to finish one fit. We are slower than XGboost and Random Forest. However, we only rely on a python script code vs an optimized compiled code. We note that the divergence function learned in PBDL can be further used for other tasks such as ranking and clustering. We present the accuracy in Figure 3 in Appendix. We observe that our Bregman divergence Learning Algorithm (PBDL) as well as the original (PBDL_0) have similar performance to that of XGboost and Random Forest.

6. Conclusion

In this paper we, studied the nonparametric convex regression problem with a $L1$ regularization penalty. we provided a solver for this problem and proved the iteration complexity be $6n\sqrt{d}/\epsilon$. The total computational complexity is $O(n^3d^{1.5}/\epsilon + n^2d^{2.5}/\epsilon + nd^3/\epsilon)$, which improves that of $O(n^5d^2/\epsilon)$ already known for this problem. We also extended our solver to the problem of difference of convex (DC) regression and the problem of learning an arbitrary Bregman divergence. We provided comparisons to state of the art regression and classification models.

Acknowledgements

This research was supported by the Secretary of Defense for Research and Engineering under Air Force Contract No. FA8702-15-D-0001, Army Research Office Grant W911NF2110246, the National Science Foundation grants CCF-2007350 and CCF-1955981, and the Hariri Data Science Faculty Fellowship Grants, and a gift from the ARM corporation.

References

Afriat, S. N. The construction of utility functions from expenditure data. *International economic review*, 8(1): 67–77, 1967.

Amos, B., Xu, L., and Kolter, J. Z. Input convex neural networks. In *International Conference on Machine Learning*, pp. 146–155. PMLR, 2017.

Bagirov, A. M., Taheri, S., Karmitsa, N., Sultanova, N., and Asadi, S. Robust piecewise linear l 1-regression via nonsmooth dc optimization. *Optimization Methods and Software*, pp. 1–21, 2020.

Balázs, G. Convex regression: theory, practice, and applications. 2016.

Bertsimas, D. and Mundru, N. Sparse convex regression. *INFORMS Journal on Computing*, 33(1):262–279, 2021.

Boyd, S., Boyd, S. P., and Vandenberghe, L. *Convex optimization*. Cambridge university press, 2004.

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

Chen, W. and Mazumder, R. Multivariate convex regression at scale. *arXiv preprint arXiv:2005.11588*, 2020.

Cui, Y., Pang, J.-S., and Sen, B. Composite difference-max programs for modern statistical estimation problems. *SIAM Journal on Optimization*, 28(4):3344–3374, 2018.

Eckstein, J. and Yao, W. Augmented lagrangian and alternating direction methods for convex optimization: A tutorial and some illustrative computational results. *RUTCOR Research Reports*, 32(3), 2012.

Fernández-Delgado, M., Cernadas, E., Barro, S., and Amorim, D. Do we need hundreds of classifiers to solve real world classification problems? *The journal of machine learning research*, 15(1):3133–3181, 2014.

Fernández-Delgado, M., Sirsat, M. S., Cernadas, E., Alawadi, S., Barro, S., and Febrero-Bande, M. An extensive experimental survey of regression methods. *Neural Networks*, 111:11–34, 2019.

Gabay, D. and Mercier, B. A dual algorithm for the solution of nonlinear variational problems via finite element approximation. *Computers & mathematics with applications*, 2(1):17–40, 1976.

Ghosh, A., Pananjady, A., Guntuboyina, A., and Ramchandran, K. Max-affine regression: Provable, tractable, and near-optimal statistical estimation. *arXiv preprint arXiv:1906.09255*, 2019.

He, B. and Yuan, X. On the o(1/n) convergence rate of the douglas–rachford alternating direction method. *SIAM Journal on Numerical Analysis*, 50(2):700–709, 2012.

Kim, S., Bahmani, S., and Lee, K. Max-linear regression by scalable and guaranteed convex programming. *arXiv preprint arXiv:2103.07020*, 2021.

Mazumder, R., Choudhury, A., Iyengar, G., and Sen, B. A computational framework for multivariate convex regression and its variants. *Journal of the American Statistical Association*, 114(525):318–331, 2019.

Nagurney, A. *Network economics: A variational inequality approach*, volume 10. Springer Science & Business Media, 1998.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32:8026–8037, 2019.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Siahkamari, A., Saligrama, V., Castanon, D., and Kulis, B. Learning bregman divergences. *arXiv e-prints*, pp. arXiv–1905, 2019.

Siahkamari, A., Gangrade, A., Kulis, B., and Saligrama, V. Piecewise linear regression via a difference of convex functions. In *International Conference on Machine Learning*, pp. 8895–8904. PMLR, 2020.

Varian, H. R. The nonparametric approach to demand analysis. *Econometrica: Journal of the Econometric Society*, pp. 945–973, 1982.

Zhang, X., Li, Y., Zhang, Z., and Zhang, Z.-L. f-gail: Learning f-divergence for generative adversarial imitation learning. *arXiv preprint arXiv:2010.01207*, 2020.

Appendix to 'Faster Convex Lipschitz Regression via 2 blocks ADMM'

A. Algorithms

Algorithm 3 Learning a Bregman divergence

```
 1: Require: $\{(\boldsymbol{x}_i, y_i) \mid \boldsymbol{x}_i \in \mathbb{R}^d, y_i \in \mathbb{N}\}_{i\in[n]}$ and $\{\rho,\ \lambda,\ T\}$
 2: $z_i = s_{i,j} = t_{i,j} = \alpha_{i,j} = \tau_{i,j} \leftarrow 0$
 3: $\boldsymbol{L} = \boldsymbol{a}_i = \boldsymbol{p}_i = \boldsymbol{u}_i = \boldsymbol{\eta}_i = \boldsymbol{\gamma}_i \leftarrow \mathbf{0}_{d\times 1}$
 4: for $t = 1$ to $T$ do
 5:     Update $\zeta_{i,j}$ by Eq. (11)
 6:     Update $\boldsymbol{z}$ by Eq. (13)
 7:     Update $\boldsymbol{a}_i$ by Eq. (12)
 8:     $L_l \leftarrow$ L_update$(\{\gamma_{i,l}, |\eta_{i,l} + a_{i,l}|\}_{i\in[n]}, \lambda/\rho)$
 9:     Update $s_{i,j}, t_{i,j}$ by Eq. (14)
10:     Update $u_{i,l}, p^+_{i,l}, p^-_{i,l}$ by Eq. (8)
11:     Update $\alpha_{i,j}, \gamma_{i,l}, \eta_{i,l}$ by Eq. (9)
12:     Update $\tau_{i,j}$ by Eq. (15)
13: end for
14: return: $f(\cdot) \triangleq \max_{i=1}^n(\langle \boldsymbol{a}_i, \cdot - \boldsymbol{x}_i\rangle + z_i)$
```

Algorithm 4 Difference of convex regression

```
Require: $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$, $\rho$, $\lambda$, and $T$
 1: $\hat{y}^q_i = s^q_{i,j} = \alpha^q_{i,j} \leftarrow 0$
 2: $\boldsymbol{L}^q = \boldsymbol{a}^q_i = \boldsymbol{p}^q_i = \boldsymbol{u}^q_i = \boldsymbol{\eta}^q_i = \boldsymbol{\gamma}^q_i \leftarrow \mathbf{0}_{d\times 1}$
 3: for $t = 1$ to $T$ do
 4:     Update $\hat{\boldsymbol{y}}^q$ by Eq. (17)
 5:     Update $\boldsymbol{a}^q_i$ by Eq. (6)
 6:     $L^q_l \leftarrow$ L_update$(\{\gamma^q_{i,l}, |\eta^q_{i,l} + a^q_{i,l}|\}_{i\in[n]}, \lambda/\rho)$
 7:     Update $u^q_{i,l}, p^{q+}_{i,l}, p^{q-}_{i,l}, s^q_{i,j}$ by Eq. (8)
 8:     Update $\alpha^q_{i,j}, \gamma^q_{i,l}, \eta^q_{i,l}$ by Eq. (9)
 9: end for
10: return $\phi^q(\cdot) \triangleq \max_{i=1}^n(\langle \boldsymbol{a}^q_i, \cdot - \boldsymbol{x}_i\rangle + \hat{y}^q_i)$
```

B. Derivations

B.1. ADMM Convergence

Theorem 1 gives convergence of the ADMM procedure. We follow analysis similar to He & Yuan (2012) and give all steps for the sake of completeness. We first restate the convergence as follows:

Theorem 1. *Consider the separable convex optimization problem:*

$$\min_{\mathbf{b}^1\in\mathcal{S}_1, \mathbf{b}^2\in\mathcal{S}_2} \left[\psi(\mathbf{b}^1, \mathbf{b}^2) = \psi_1(\mathbf{b}^1) + \psi_2(\mathbf{b}^2)\right]$$
$$s.t : \mathbf{A}\mathbf{b}^1 + \mathbf{B}\mathbf{b}^2 + \boldsymbol{b} = \mathbf{0},$$

where ($\mathbf{b}^1$, $\mathbf{b}^2$) are the block variables, ($\mathcal{S}_1$, $\mathcal{S}_2$) are convex sets that includes all zero vectors, ($\mathbf{A}, \mathbf{B}$) are the coefficient matrices and $\mathbf{b}$ is a constant vector. Let $\mathbf{b}^1_t$ and $\mathbf{b}^2_t$ be solutions at iteration t of a two block ADMM procedure with learning *rate ρ. i.e:*

$$\begin{aligned}
\mathbf{b}^1_{t+1} &= \arg\min_{\mathbf{b}^1\in\mathcal{S}_1} \psi_1(\mathbf{b}^1) + \frac{\rho}{2}\left\|\mathbf{A}\mathbf{b}^1 + \mathbf{B}\mathbf{b}^2_t + \mathbf{b} - \frac{1}{\rho}\mathbf{d}_t\right\|^2 \\
\mathbf{b}^2_{t+1} &= \arg\min_{\mathbf{b}^2\in\mathcal{S}_2} \psi_2(\mathbf{b}^2) + \frac{\rho}{2}\left\|\mathbf{A}\mathbf{b}^1_{t+1} + \mathbf{B}\mathbf{b}^2 + \mathbf{b} - \frac{1}{\rho}\mathbf{d}_t\right\|^2 \\
\mathbf{d}_{t+1} &= \mathbf{d}_t - \rho\left(\mathbf{A}\mathbf{b}^1_{t+1} + \mathbf{B}\mathbf{b}^2_{t+1} + \mathbf{b}\right),
\end{aligned}$$

where $\mathbf{b}^1_0 = \mathbf{b}^2_0 = \mathbf{d}_0 = 0$.

Denote the average of iterates as $(\tilde{\mathbf{b}}^1_T, \tilde{\mathbf{b}}^2_T) = \left(\frac{1}{T}\sum_{t=1}^T \mathbf{b}^1_t, \frac{1}{T}\sum_{t=1}^T \mathbf{b}^2_t\right)$. For all $\boldsymbol{\kappa}$ we have

$$\psi(\tilde{\mathbf{b}}^1_T, \tilde{\mathbf{b}}^2_T) - \psi(\mathbf{b}^1_*, \mathbf{b}^2_*) - \boldsymbol{\kappa}^T(\mathbf{A}\tilde{\mathbf{b}}^1_T + \mathbf{B}\tilde{\mathbf{b}}^2_T + \boldsymbol{b}) \le \frac{1}{T}\left(\frac{\rho}{2}\|\mathbf{B}\mathbf{b}^2_*\|^2 + \frac{1}{2\rho}\|\boldsymbol{\kappa}\|^2\right),$$

where $(\mathbf{b}^*_1, \mathbf{b}^*_2)$ are the optimal solutions.

Before presenting the proof, we define a dual variable like sequence $\{\overline{\mathbf{d}}_t\}_t$ with the update relation as

$$\overline{\mathbf{d}}_{t+1} = \mathbf{d}_t - \rho(\mathbf{A}\mathbf{b}^1_{t+1} + \mathbf{B}\mathbf{b}^2_t + \mathbf{b}).$$

The only difference between $\overline{\mathbf{d}}_{t+1}$ and $\mathbf{d}_{t+1}$ is due to $\mathbf{b}^2_t$ and $\mathbf{b}^2_{t+1}$. We have the following relation:

$$\overline{\mathbf{d}}_{t+1} - \mathbf{d}_{t+1} = \rho\mathbf{B}(\mathbf{b}^2_{t+1} - \mathbf{b}^2_t). \tag{18}$$

Lemma 1. *[from Nagurney (1998)] Let $\mathbf{b}^* = \min_{\mathbf{b}\in\mathcal{S}} \psi(\mathbf{b})$ where ψ is continuously differentiable and $\mathcal{S}$ is closed and convex. Then $\mathbf{b}^*$ satisfies,*

$$\langle \nabla\psi(\mathbf{b}^*), \mathbf{b} - \mathbf{b}^*\rangle \ge 0, \qquad \forall \mathbf{b}\in\mathcal{S}.$$

Applying Lemma 1 to optimization problems of ADMM gives,

$$\left\langle \nabla\psi\left(\mathbf{b}^1_{t+1}\right) + \rho\mathbf{A}^T\left(\mathbf{A}\mathbf{b}^1_{t+1} + \mathbf{B}\mathbf{b}^2_t + \mathbf{b} - \frac{1}{\rho}\mathbf{d}_t\right), \mathbf{b}^1 - \mathbf{b}^1_{t+1}\right\rangle \ge 0, \quad \forall\mathbf{b}^1\in\mathcal{S}_1 \tag{19}$$

$$\left\langle \nabla\psi\left(\mathbf{b}^2_{t+1}\right) + \rho\mathbf{B}^T\left(\mathbf{A}\mathbf{b}^1_{t+1} + \mathbf{B}\mathbf{b}^2_{t+1} + \mathbf{b} - \frac{1}{\rho}\mathbf{d}_t\right), \mathbf{b}^2 - \mathbf{b}^2_{t+1}\right\rangle \ge 0, \quad \forall\mathbf{b}^2\in\mathcal{S}_2. \tag{20}$$

Combining Eq. 19, 20, and the $\{\overline{\mathbf{d}}_t\}_t$ update rule we get, $\forall\mathbf{b}^1\in\mathcal{S}_1, \forall\mathbf{b}^2\in\mathcal{S}_2$,

$$\left\langle\nabla\psi\left(\mathbf{b}^1_{t+1}\right) - \mathbf{A}^T\overline{\mathbf{d}}_{t+1}, \mathbf{b}^1 - \mathbf{b}^1_{t+1}\right\rangle \ge 0, \quad \left\langle\nabla\psi\left(\mathbf{b}^2_{t+1}\right) - \mathbf{B}^T\overline{\mathbf{d}}_{t+1} - \rho\mathbf{B}^T\mathbf{B}(\mathbf{b}^2_t - \mathbf{b}^2_{t+1}), \mathbf{b}^2 - \mathbf{b}^2_{t+1}\right\rangle \ge 0. \tag{21}$$

Theorem 1 is a direct conclusion of the following Lemma.

Lemma 2. *For all $\mathbf{b}^1, \mathbf{b}^2, \kappa$ we have,*

$$\psi(\mathbf{b}^1, \mathbf{b}^2) - \psi(\mathbf{b}^1_{t+1}, \mathbf{b}^2_{t+1}) + \boldsymbol{\kappa}^T(\mathbf{A}\mathbf{b}^1_{t+1} + \mathbf{B}\mathbf{b}^2_{t+1} + \boldsymbol{b}) - \overline{\mathbf{d}}^T_{t+1}(\mathbf{A}\mathbf{b}^1 + \mathbf{B}\mathbf{b}^2 + \mathbf{b}) + Z_t - Z_{t+1} \ge 0,$$

where $Z_t = \frac{\rho}{2}\|\mathbf{b}^2 - \mathbf{b}^2_t\|^2_{\mathbf{B^TB}} + \frac{1}{2\rho}\|\boldsymbol{\kappa} - \mathbf{d}_t\|^2$ and $\|\mathbf{x}\|^2_{\mathbf{M}} = \mathbf{x^T M x}$.

Let $\mathbf{b}^1, \mathbf{b}^2$ be $(\mathbf{b}^*_1, \mathbf{b}^*_2)$. By definition we have $\mathbf{A}\mathbf{b}^1_* + \mathbf{B}\mathbf{b}^2_* + \mathbf{b} = \mathbf{0}$ which cancels an inner product. Rearranging and averaging over time gives

$$\begin{aligned}
&\psi(\mathbf{b}^1_{t+1}, \mathbf{b}^2_{t+1}) - \psi(\mathbf{b}^1_*, \mathbf{b}^2_*) - \boldsymbol{\kappa}^T(\mathbf{A}\mathbf{b}^1_{t+1} + \mathbf{B}\mathbf{b}^2_{t+1} + \boldsymbol{b}) \le Z_t - Z_{t+1} \\
&\frac{1}{T}\left(\sum_{t=0}^{T-1}\psi(\mathbf{b}^1_{t+1}, \mathbf{b}^2_{t+1})\right) - \psi(\mathbf{b}^1_*, \mathbf{b}^2_*) - \boldsymbol{\kappa}^T(\mathbf{A}\tilde{\mathbf{b}}^1_T + \mathbf{B}\tilde{\mathbf{b}}^2_T + \boldsymbol{b}) \le \frac{1}{T}\left(\sum_{t=0}^{T-1} Z_t - Z_{t+1}\right).
\end{aligned}$$

The RHS telescopes. Since $Z_t \geq 0$ we have $RHS \leq Z_0$. We lower bound LHS with Jensen Inq. as $\psi(\tilde{\mathbf{b}}^1_T, \tilde{\mathbf{b}}^2_T) \leq \frac{1}{T}\sum_{t=0}^{T-1}\psi(\mathbf{b}^1_{t+1}, \mathbf{b}^2_{t+1})$. Combining LHS and RHS we get

$$\psi(\tilde{\mathbf{b}}^1_T, \tilde{\mathbf{b}}^2_T) - \psi(\mathbf{b}^1_*, \mathbf{b}^2_*) - \boldsymbol{\kappa}^T(\mathbf{A}\tilde{\mathbf{b}}^1_T + \mathbf{B}\tilde{\mathbf{b}}^2_T + \boldsymbol{b}) \leq \frac{1}{T}Z_0,$$

which is the statement in Theorem 1 assuming the initial variables are 0s.

We continue to prove Lemma 2. We start with convexity definition for both ψ_1 and ψ as,

$$\psi_1(\mathbf{b}^1) - \psi_1(\mathbf{b}^1_{t+1}) - (\mathbf{b}^1 - \mathbf{b}^1_{t+1})^T\nabla\psi_1(\mathbf{b}^1_{t+1}) \geq 0; \quad \psi_2(\mathbf{b}^2) - \psi_2(\mathbf{b}^2_{t+1}) - (\mathbf{b}^2 - \mathbf{b}^2_{t+1})^T\nabla\psi_2(\mathbf{b}^2_{t+1}) \geq 0.$$

Summing the relations and plugging in Eq. 21 give,

$$\psi(\mathbf{b}^1, \mathbf{b}^2) - \psi(\mathbf{b}^1_{t+1}, \mathbf{b}^2_{t+1}) \underbrace{-(\mathbf{b}^1 - \mathbf{b}^1_{t+1})^T\mathbf{A}^T\overline{\mathbf{d}}_{t+1} - (\mathbf{b}^2 - \mathbf{b}^2_{t+1})^T\mathbf{B}^T\overline{\mathbf{d}}_{t+1} - \rho(\mathbf{b}^2 - \mathbf{b}^2_{t+1})^T\mathbf{B}^T\mathbf{B}(\mathbf{b}^2_t - \mathbf{b}^2_{t+1})}_{T_1} \geq 0. \tag{22}$$

The update rule of $\overline{\mathbf{d}}$ gives $\mathbf{A}\mathbf{b}^1_{t+1} + \mathbf{B}\mathbf{b}^2_t + \mathbf{b} + \frac{1}{\rho}(\overline{\mathbf{d}}_{t+1} - \mathbf{d}_t) = \mathbf{0}$. Then for all $\boldsymbol{\kappa}$, we have $(\boldsymbol{\kappa} - \overline{\mathbf{d}}_{t+1})^T(\mathbf{A}\mathbf{b}^1_{t+1} + \mathbf{B}\mathbf{b}^2_t + \mathbf{b} + \frac{1}{\rho}(\overline{\mathbf{d}}_{t+1} - \mathbf{d}_t)) = 0$. Adding such a term to T_1 does not change its value. After adding the zero inner product, we rearrange T_1 as

$$\begin{aligned} T_1 = &\underbrace{\boldsymbol{\kappa}^T(\mathbf{A}\mathbf{b}^1_{t+1} + \mathbf{B}\mathbf{b}^2_{t+1} + \boldsymbol{b}) - \overline{\mathbf{d}}^T_{t+1}(\mathbf{A}\mathbf{b}^1 + \mathbf{B}\mathbf{b}^2 + \mathbf{b})}_{Y_1} + \rho\underbrace{(\mathbf{b}^2 - \mathbf{b}^2_{t+1})^T\mathbf{B}^T\mathbf{B}(\mathbf{b}^2_{t+1} - \mathbf{b}^2_t)}_{Y_2} \\ &+ \underbrace{(\boldsymbol{\kappa} - \overline{\mathbf{d}}_{t+1})^T(\mathbf{B}(\mathbf{b}^2_t - \mathbf{b}^2_{t+1}) + \frac{1}{\rho}(\overline{\mathbf{d}}_{t+1} - \mathbf{d}_t))}_{Y_3}. \end{aligned}$$

Plugging Eq. 18 in Y_3 gives $Y_3 = \frac{1}{\rho}(\boldsymbol{\kappa} - \overline{\mathbf{d}}_{t+1})^T(\mathbf{d}_{t+1} - \mathbf{d}_t)$.

We use the following norm square relation and prove it at the end,

Lemma 3. *For a symmetric* $\mathbf{M} = \mathbf{M}^T$ *we have,*

$$(\mathbf{x} - \mathbf{y})^T\mathbf{M}(\mathbf{z} - \mathbf{t}) = \frac{1}{2}\left(\|\mathbf{x} - \mathbf{t}\|^2_{\mathbf{M}} - \|\mathbf{x} - \mathbf{z}\|^2_{\mathbf{M}}\right) + \frac{1}{2}\left(\|\mathbf{y} - \mathbf{z}\|^2_{\mathbf{M}} - \|\mathbf{y} - \mathbf{t}\|^2_{\mathbf{M}}\right).$$

Using Lemma 3, we get

$Y_2 = \frac{1}{2}\left(\|\mathbf{b}^2 - \mathbf{b}^2_t\|^2_{\mathbf{B^TB}} - \|\mathbf{b}^2 - \mathbf{b}^2_{t+1}\|^2_{\mathbf{B^TB}}\right) - \frac{1}{2}\|\mathbf{b}^2_{t+1} - \mathbf{b}^2_t\|^2_{\mathbf{B^TB}}$

$Y_3 = \frac{1}{\rho}\frac{1}{2}\left(\|\boldsymbol{\kappa} - \mathbf{d}_t\|^2_{\mathbf{I}} - \|\boldsymbol{\kappa} - \mathbf{d}_{t+1}\|^2_{\mathbf{I}}\right) + \frac{1}{\rho}\frac{1}{2}\left(\|\overline{\mathbf{d}}_{t+1} - \mathbf{d}_{t+1}\|^2_{\mathbf{I}} - \|\overline{\mathbf{d}}_{t+1} - \mathbf{d}_t\|^2_{\mathbf{I}}\right)$,

since $\mathbf{B^TB}$ and $\mathbf{I}$ are symmetric matrices.

We combine Y_2 and Y_3 as

$$\begin{aligned} \rho Y_2 + Y_3 = &\frac{\rho}{2}\left(\|\mathbf{b}^2 - \mathbf{b}^2_t)\|^2_{\mathbf{B^TB}} - \|\mathbf{b}^2 - \mathbf{b}^2_{t+1}\|^2_{\mathbf{B^TB}}\right) + \frac{1}{2\rho}\left(\|\boldsymbol{\kappa} - \mathbf{d}_t\|^2_{\mathbf{I}} - \|\boldsymbol{\kappa} - \mathbf{d}_{t+1}\|^2_{\mathbf{I}}\right) \\ &\underbrace{-\frac{1}{2\rho}\|\overline{\mathbf{d}}_{t+1} - \mathbf{d}_t\|^2_{\mathbf{I}}}_{\leq 0} + \underbrace{\frac{1}{2\rho}\left(\|\overline{\mathbf{d}}_{t+1} - \mathbf{d}_{t+1}\|^2_{\mathbf{I}} - \rho^2\|\mathbf{b}^2_{t+1} - \mathbf{b}^2_t\|_{\mathbf{B^TB}}\right)}_{=0 \text{ due to Eq. } 18} \\ \leq &\frac{\rho}{2}\left(\|\mathbf{b}^2 - \mathbf{b}^2_t)\|^2_{\mathbf{B^TB}} - \|\mathbf{b}^2 - \mathbf{b}^2_{t+1}\|^2_{\mathbf{B^TB}}\right) + \frac{1}{2\rho}\left(\|\boldsymbol{\kappa} - \mathbf{d}_t\|^2_{\mathbf{I}} - \|\boldsymbol{\kappa} - \mathbf{d}_{t+1}\|^2_{\mathbf{I}}\right). \end{aligned}$$

Plugging it back to T_1,

$$T_1 = Y_1 + \rho Y_2 + Y_3 \leq Y_1 + \frac{\rho}{2}\left(\|\mathbf{b}^2 - \mathbf{b}^2_t)\|^2_{\mathbf{B^TB}} - \|\mathbf{b}^2 - \mathbf{b}^2_{t+1}\|^2_{\mathbf{B^TB}}\right) + \frac{1}{2\rho}\left(\|\boldsymbol{\kappa} - \mathbf{d}_t\|^2_{\mathbf{I}} - \|\boldsymbol{\kappa} - \mathbf{d}_{t+1}\|^2_{\mathbf{I}}\right) \leq Y_1 + Z_t - Z_{t+1}. \tag{23}$$

Upper bounding T_1 term in Eq. 22 with Eq. 23 gives

$$\psi(\mathbf{b}^1, \mathbf{b}^2) - \psi(\mathbf{b}^1_{t+1}, \mathbf{b}^2_{t+1}) + \boldsymbol{\kappa}^T(\mathbf{A}\mathbf{b}^1_{t+1} + \mathbf{B}\mathbf{b}^2_{t+1} + \boldsymbol{b}) - \overline{\mathbf{d}}^T_{t+1}(\mathbf{A}\mathbf{b}^1 + \mathbf{B}\mathbf{b}^2 + \mathbf{b}) + Z_t - Z_{t+1} \geq 0,$$

which is the statement in Lemma 2.□

Proof of Lemma 1 [from Nagurney (1998)].

Let $\phi(\mathbf{t}) = \psi(\mathbf{b}^* + t(\mathbf{b} - \mathbf{b}^*))$ for $t \in [0, 1]$. We know that ϕ achieves its minimum at $t = 0$. Since $\mathcal{S}$ is convex and closed, we have, $0 \leq [\nabla \phi(t)]_{t=0} = \langle \nabla \psi(\mathbf{b}^*), \mathbf{b} - \mathbf{b}^* \rangle$.□

Proof of Lemma 3.

Expand RHS as

$$\begin{aligned} RHS &= \frac{1}{2}\left(\mathbf{t}^T\mathbf{M}\mathbf{t} - 2\mathbf{x}^T\mathbf{M}\mathbf{t} - \mathbf{z}^T\mathbf{M}\mathbf{z} + 2\mathbf{x}^T\mathbf{M}\mathbf{z}\right) + \frac{1}{2}\left(\mathbf{z}^T\mathbf{M}\mathbf{z} - 2\mathbf{y}^T\mathbf{M}\mathbf{z} - \mathbf{t}^T\mathbf{M}\mathbf{t} + 2\mathbf{t}^T\mathbf{M}\mathbf{y}\right) \\ &= -\mathbf{x}^T\mathbf{M}\mathbf{t} + \mathbf{x}^T\mathbf{M}\mathbf{z} - \mathbf{y}^T\mathbf{M}\mathbf{z} + \mathbf{t}^T\mathbf{M}\mathbf{y} \\ &= (\mathbf{x} - \mathbf{y})^T\mathbf{M}(\mathbf{z} - \mathbf{t}). \end{aligned}$$

We reach the LHS. □

B.2. Proof for Theorem 2 (computational complexity)

Theorem 2. *Let $\{\hat{y}^t_i, \boldsymbol{a}^t_i\}^n_{i=1}$ be the output of Algorithm(2) at the t^{th} iteration, $\tilde{y}_i \triangleq \frac{1}{T}\sum_{t=1}^T \hat{y}^t_i$ and $\tilde{\boldsymbol{a}}_i \triangleq \frac{1}{T}\sum_{t=1}^T \boldsymbol{a}^t_i$. Denote $\tilde{f}_T(\boldsymbol{x}) \triangleq \max_i \langle \tilde{\boldsymbol{a}}_i, \boldsymbol{x} - \boldsymbol{x}_i \rangle + \tilde{y}_i$. Assume $\max_{i,l} |x_{i,l}| \leq 1$ and $\mathbb{Var}(\{y_i\}^n_{i=1}) \leq 1$. If we choose $\rho = \frac{\sqrt{d}\lambda^2}{n}$, for $\lambda \geq \frac{3}{\sqrt{2nd}}$ and $T \geq n\sqrt{d}$ we have:*

$$\frac{1}{n}\sum_i (\tilde{f}_T(\boldsymbol{x}_i) - y_i)^2 + \lambda\|\tilde{f}_T\| \leq \min_{\hat{f}} \left(\frac{1}{n}\sum_i \left(\hat{f}(\boldsymbol{x}_i) - y_i\right)^2 + \lambda\|\hat{f}\|\right) + \frac{6n\sqrt{d}}{T+1}.$$

For the proof, we make extensive use of Theorem (1) which provides convergence rate of a general 2-block ADMM for a separable convex program with linear constraints

$$\begin{aligned} &\min_{\mathbf{b}^1 \in \mathcal{S}_1, \mathbf{b}^2 \in \mathcal{S}_2} \left[\psi(\mathbf{b}^1, \mathbf{b}^2) = \psi_1(\mathbf{b}^1) + \psi_2(\mathbf{b}^2)\right] \\ &s.t: \mathbf{A}\mathbf{b}^1 + \mathbf{B}\mathbf{b}^2 + \boldsymbol{b} = \mathbf{0}. \end{aligned} \tag{24}$$

It guarantees for all $\boldsymbol{\kappa}$ the average solutions of a 2-block ADMM $(\tilde{\mathbf{b}}^1_T, \tilde{\mathbf{b}}^2_T)$ satisfies

$$\psi(\tilde{\mathbf{b}}^1_T, \tilde{\mathbf{b}}^2_T) - \psi(\mathbf{b}^1_*, \mathbf{b}^2_*) - \boldsymbol{\kappa}^T(\mathbf{A}\tilde{\mathbf{b}}^1_T + \mathbf{B}\tilde{\mathbf{b}}^2_T + \boldsymbol{b}) \leq \frac{1}{T}\left(\frac{\rho}{2}\|\mathbf{B}\mathbf{b}^2_*\|^2 + \frac{1}{2\rho}\|\boldsymbol{\kappa}\|^2\right), \tag{25}$$

where $(\mathbf{b}^*_1, \mathbf{b}^*_2)$ are the optimal solutions of program (24).

However in Eq. (5) we are solving a specific version of program (24) of the form

$$\min_{\hat{y}_i, \boldsymbol{a}_i, \boldsymbol{p}_i, L_l \geq 0, \boldsymbol{u}_i \geq 0, s_{i,j} \geq 0} \left[\psi(\hat{y}_i, \boldsymbol{a}_i, \boldsymbol{p}_i, L_l, \boldsymbol{u}_i, s_{i,j}) = \frac{1}{n}\sum_{i=1}^n (\hat{y}_i - y_i)^2 + \lambda\sum_{l=1}^d L_l\right] \tag{26}$$

$$\text{s.t.} \begin{cases} -s_{i,j} + \hat{y}_i - \hat{y}_j - \langle \boldsymbol{a}_j, \boldsymbol{x}_i - \boldsymbol{x}_j \rangle = 0 & i, j \in [n] \times [n] \\ u_{i,l} + |p_{i,l}| - L_l = 0 & i, l \in [n] \times [d] \\ a_{i,l} - p_{i,l} = 0 & i, l \in [n] \times [d]. \end{cases}$$

To match the notations between (26) and (24) denote the optimal/average ADMM solutions to program (26) as

$$\begin{aligned} \mathbf{b}^1_* &= [\hat{y}_1, \ldots, \hat{y}_n, a_{1,1}, a_{1,2}, \ldots, a_{n,d}]^T \\ \mathbf{b}^2_* &= [L_1, \ldots, L_d, p_{1,1}, \ldots, p_{n,d}, u_{1,1}, \ldots, u_{n,d}, s_{1,1}, \ldots, s_{n,n}]^T \\ \tilde{\mathbf{b}}^1_T &= [\tilde{y}_1, \ldots, \tilde{y}_n, \tilde{a}_{1,1}, \tilde{a}_{1,2}, \ldots, \tilde{a}_{n,d}]^T \\ \tilde{\mathbf{b}}^2_T &= [\tilde{L}_1, \ldots, \tilde{L}_d, \tilde{p}_{1,1}, \ldots, \tilde{p}_{n,d}, \tilde{u}_{1,1}, \ldots, \tilde{u}_{n,d}, \tilde{s}_{1,1}, \ldots, \tilde{s}_{n,n}]^T. \end{aligned} \tag{27}$$

Therefore the separable losses for the average iterates of ADMM for program (26) are

$$\psi_1(\tilde{\mathbf{b}}_T^1) = \frac{1}{n}\sum_{i=1}^{n}(\tilde{y}_i - y_i)^2, \tag{28}$$
$$\psi_2(\tilde{\mathbf{b}}_T^2) = \lambda\sum_{l=1}^{d}\tilde{L}_l.$$

Note that convergence rate for $\psi_1(\tilde{\mathbf{b}}_T^1) + \psi_2(\tilde{\mathbf{b}}_T^2)$ is available by setting $\boldsymbol{\kappa} = 0$ in (25). However this is not sufficient for convergence of our objective $\frac{1}{n}\sum_i(\tilde{f}_T(\boldsymbol{x}_i) - y_i)^2 + \lambda\|\tilde{f}_T\|$. The reason is $(\tilde{\mathbf{b}}_T^1, \tilde{\mathbf{b}}_T^2)$ might be violating the constraints in (26). This could result in $\tilde{f}_T(\boldsymbol{x}_i) \neq \tilde{y}_i$ and $\|\tilde{f}_T\| \neq \sum_{l=1}^{d}\tilde{L}_l$. Therefore main steps of the proof of Theorem (2) is to characterize and bound the effect of such constraint violations on our objective function. Let us specify how much each linear constraint in program (26) is violated,

$$\begin{aligned}
\varepsilon^1_{i,j} &\triangleq -\tilde{s}_{i,j} + \tilde{y}_i - \tilde{y}_j - \langle\tilde{\boldsymbol{a}}_j, \boldsymbol{x}_i - \boldsymbol{x}_j\rangle, \\
\varepsilon^2_{i,l} &\triangleq \tilde{u}_{i,l} + |\tilde{p}_{i,l}| - \tilde{L}_l, \\
\varepsilon^3_{i,l} &\triangleq \tilde{a}_{i,l} - \tilde{p}_{i,l}.
\end{aligned} \tag{29}$$

We break the proof to smaller parts by first providing some intermediate lemmas.

Lemma 4. $\psi(\mathbf{b}^1_*, \mathbf{b}^2_*) = \frac{1}{n}\sum_i(\hat{y}_i - y_i)^2 + \lambda\sum_l L_l \leq 1.$

Proof. Note that $\boldsymbol{b}^1 = \boldsymbol{b}^2 = 0$ is a feasible solution and $(\hat{y}_i, L_l)$ is the optimal solution. Also algorithm 2 normalizes the dataset such that $\sum_{i=1}^{n} y_i = 0$, therefore

$$\begin{aligned}
\psi(\mathbf{b}^1_*, \mathbf{b}^2_*) &\leq \psi(\mathbf{0}, \mathbf{0}) \\
&= \frac{1}{n}\sum_{i=1}^{n} y_i^2 = \mathbb{Var}(\{y_i\}_{i=1}^n) \leq 1.
\end{aligned}$$

□

Lemma 5. $\|\mathbf{B}\mathbf{b}^2_*\|_2^2 \leq 18\frac{n^2}{\lambda^2}$.

Proof. Since $p_{i,l}$ is a feasible solution we have $|p_{i,l}| \leq L_l$. We also have:

$$\begin{aligned}
0 \leq s_{i,j} &= \hat{y}_i - \hat{y}_j - \langle\boldsymbol{a}_j, \boldsymbol{x}_i - \boldsymbol{x}_j\rangle && \text{(by constraints definitions)} \\
&\leq \langle\boldsymbol{a}_i - \boldsymbol{a}_j, \boldsymbol{x}_i - \boldsymbol{x}_j\rangle && \text{(convexity)} \\
&\leq 2\sum_l L_l|x_{i,l} + x_{j,l}| && (\max_i |a_{i,l}| \leq L_l) \\
&\leq 4\sum_l L_l && (\max_i |x_{i,l}| \leq 1).
\end{aligned}$$

Arranging the constraints in (26) into $(n^2 + nd + nd)$ rows and separating the $\mathbf{b}^1_*, \mathbf{b}^2_*$ coefficients as $\mathbf{A}$ and $\mathbf{B}$ in order we have:

$$\begin{aligned}
&\mathbf{A}\mathbf{b}^1_* + \mathbf{B}\mathbf{b}^2_* = 0 \\
&\mathbf{B}\mathbf{b}^2_* = [-s_{1,1}, \ldots, -s_{n,n}, 0_{1,1}, \ldots, 0_{n,d}, -p_{1,1}, \ldots, -p_{n,d}]^T.
\end{aligned}$$

Taking the norm

$$\begin{aligned}\|\mathbf{B}\mathbf{b}_*^2\|_2^2 &= \sum_{i,j} s_{i,j}^2 + \sum_{i,l} p_{i,l}^2 \\ &\leq n^2(4\sum_{l=1}^{d} L_l)^2 + n\sum_{l=1}^{d} L_l^2 \\ &\leq 18\frac{n^2}{\lambda^2}. && \text{(Lemma 4)}\end{aligned}$$

□

Lemma 6. $0 \leq \tilde{f}_T(\boldsymbol{x}_i) - \tilde{y}_i \leq \max_j(\varepsilon_{i,j}^1)$.

Proof.

$$\begin{aligned}\tilde{y}_i \leq \tilde{f}_T(\boldsymbol{x}_i) &= \max_j \langle \tilde{\boldsymbol{a}}_j, \boldsymbol{x}_i - \boldsymbol{x}_j\rangle + \tilde{y}_j && \text{(Definition)} \\ &= \max_j(\tilde{y}_i - \tilde{s}_{i,j} + \varepsilon_{i,j}^1) && (\varepsilon_{i,j}^1 \text{ definition}) \\ &\leq \tilde{y}_i + \max_j(\varepsilon_{i,j}^1). && (\tilde{s}_{i,j} \geq 0)\end{aligned}$$

□

Lemma 7.

$$\frac{1}{n}\sum_i \left(\tilde{f}_T(\boldsymbol{x}_i) - y_i\right)^2 \leq \frac{1}{n}\sum_i (\tilde{y}_i - y_i)^2 + \frac{1}{n}\sum_i \max_j(\varepsilon_{i,j}^1)^2 + 2\sqrt{\frac{1}{n}\sum_i \max_j(\varepsilon_{i,j}^1)^2}\sqrt{1 + \frac{1}{T}\frac{9\rho n^2}{\lambda^2}}.$$

Proof.

$$\begin{aligned}\frac{1}{n}\sum_i \left(\tilde{f}_T(\boldsymbol{x}_i) - y_i\right)^2 &= \frac{1}{n}\sum_i \left(\tilde{f}_T(\boldsymbol{x}_i) - \tilde{y}_i + \tilde{y}_i - y_i\right)^2 \\ &= \frac{1}{n}\sum_i (\tilde{y}_i - y_i)^2 + \frac{1}{n}\sum_i \left(\tilde{f}_T(\boldsymbol{x}_i) - \tilde{y}_i\right)^2 + \frac{2}{n}\left(\tilde{f}_T(\boldsymbol{x}_i) - \tilde{y}_i\right)\left(\tilde{y}_i - y_i\right) \\ &\leq \frac{1}{n}\sum_i (\tilde{y}_i - y_i)^2 + \frac{1}{n}\sum_i \max_j(\varepsilon_{i,j}^1)^2 + \frac{2}{n}|\max_j(\varepsilon_{i,j}^1)||\tilde{y}_i - y_i|. && \text{(Lemma 6)}\end{aligned}$$

Let us focus on the last term on the RHS.

$$\begin{aligned}\frac{\left(\frac{1}{n}\sum_i \left|\max_j(\varepsilon_{i,j}^1)||\tilde{y}_i - y_i|\right)^2}{\frac{1}{n}\sum_i(\max_j \varepsilon_{i,j}^1)^2} &\leq \frac{1}{n}\sum_i (\tilde{y}_i - y_i)^2 && \text{(Cauchy–Schwartz)} \\ &\leq \frac{1}{n}\sum_i (\tilde{y}_i - y_i)^2 + \lambda\sum_l \tilde{L}_l && (\tilde{L}_l \geq 0) \\ &\leq \frac{1}{n}\sum_i (\hat{y}_i - y_i)^2 + \lambda\sum_l L_l + \frac{1}{T}\frac{\rho}{2}\|\mathbf{B}\mathbf{b}_*^2\|^2 && \text{(Use Theorem 1 with } \boldsymbol{\kappa} = 0) \\ &\leq 1 + \frac{1}{T}\frac{\rho}{2}\|\mathbf{B}\mathbf{b}_*^2\|^2 && \text{(Lemma 4)} \\ &\leq 1 + \frac{1}{T}\frac{9\rho n^2}{\lambda^2}. && \text{(Lemma 5)}\end{aligned}$$

□

On the other hand for regularization term we have:

Lemma 8. $\|\tilde{f}_T\| \leq \sum_l |\tilde{L}_l| + \sum_l \max_i |\varepsilon^2_{i,l}| + \max_i |\varepsilon^3_{i,l}|$.

Proof.

$$\begin{aligned}
\|\tilde{f}_T\| &= \sum_l \max_i |\tilde{\boldsymbol{a}}_{i,l}| && \text{(definition)}\\
&= \sum_l \max_i |\tilde{\boldsymbol{p}}_{i,l} + \varepsilon^3_{i,l}| && (\varepsilon^3_{i,l} \text{ definition})\\
&\leq \sum_l |\tilde{L}_l| + \sum_l \max_i |\varepsilon^2_{i,l}| + \max_i |\varepsilon^3_{i,l}|. && (\varepsilon^2_{i,l} \text{ definition})
\end{aligned}$$

□

Next we combine the previous lemmas to prove Theorem 2. Before proceeding lets incorporate the definitions of constraint violations Eq. (29) in $\boldsymbol{\kappa}^T(\mathbf{A}\tilde{\mathbf{b}}^1_T + \mathbf{B}\tilde{\mathbf{b}}^2_T + \boldsymbol{b})$ to get:

$$\boldsymbol{\kappa}^T(\mathbf{A}\tilde{\mathbf{b}}^1_T + \mathbf{B}\tilde{\mathbf{b}}^2_T + \boldsymbol{b}) = {\boldsymbol{\kappa}^1}^T\boldsymbol{\varepsilon}^1 + {\boldsymbol{\kappa}^2}^T\boldsymbol{\varepsilon}^2 + {\boldsymbol{\kappa}^3}^T\boldsymbol{\varepsilon}^3, \tag{30}$$

where

$$\begin{aligned}
&\boldsymbol{\kappa}^1 \triangleq [\kappa^1_{1,1}, \ldots, \kappa^2_{n,n}]^T, && \boldsymbol{\kappa}^2 \triangleq [\kappa^2_{1,1}, \ldots, \kappa^2_{n,d}]^T, && \boldsymbol{\kappa}^3 \triangleq [\kappa^3_{1,1}, \ldots, \kappa^3_{n,d}]^T\\
&\boldsymbol{\varepsilon}^1 \triangleq [\varepsilon^1_{1,1}, \ldots, \varepsilon^2_{n,n}]^T, && \boldsymbol{\varepsilon}^2 \triangleq [\varepsilon^2_{1,1}, \ldots, \varepsilon^2_{n,d}]^T, && \boldsymbol{\varepsilon}^3 \triangleq [\varepsilon^3_{1,1}, \ldots, \varepsilon^3_{n,d}]^T.
\end{aligned}$$

Substituting (30) in (25) we have:

$$\begin{aligned}
\Delta &\triangleq \psi(\tilde{\mathbf{b}}^1_T, \tilde{\mathbf{b}}^2_T) - \psi(\mathbf{b}^1_*, \mathbf{b}^2_*) - {\boldsymbol{\kappa}^1}^T\boldsymbol{\varepsilon}^1 - {\boldsymbol{\kappa}^2}^T\boldsymbol{\varepsilon}^2 - {\boldsymbol{\kappa}^3}^T\boldsymbol{\varepsilon}^3\\
&\leq \frac{1}{T}\left(\frac{\rho}{2}\|\mathbf{B}\mathbf{b}^2_*\|^2 + \frac{1}{2\rho}\|\boldsymbol{\kappa}^1\|^2 + \frac{1}{2\rho}\|\boldsymbol{\kappa}^2\|^2 + \frac{1}{2\rho}\|\boldsymbol{\kappa}^3\|^2\right).
\end{aligned} \tag{31}$$

Proof. Add up both sides of Lemma 8 and Lemma 7 and subtract $\psi(\mathbf{b}^1_*, \mathbf{b}^2_*)$. The total approximation error due to ADMM optimization schema is LHS:

$$\begin{aligned}
LHS &\triangleq \frac{1}{n}\sum_i \left(\tilde{f}_T(\boldsymbol{x}_i) - y_i\right)^2 + \lambda\|\tilde{f}_T\| - \min_{\hat{f}}\left(\frac{1}{n}\sum_i \left(\hat{f}(\boldsymbol{x}_i) - y_i\right)^2 + \lambda\|\hat{f}\|\right)\\
&\leq \psi(\tilde{\mathbf{b}}^1_T, \tilde{\mathbf{b}}^2_T) - \psi(\mathbf{b}^1_*, \mathbf{b}^2_*)\\
&+ \frac{1}{n}\sum_i (\varepsilon^1_{i,l*})^2 + 2\underbrace{\sqrt{\frac{1}{n}\sum_i (\max_j \varepsilon^1_{i,j})^2}}_{\epsilon_1}\underbrace{\sqrt{1 + \frac{1}{T}\frac{9\rho n^2}{\lambda^2}}}_{\epsilon_2} + \lambda\sum_l \left(\max_i |\varepsilon^2_{i,l}| + \max_i |\varepsilon^3_{i,l}|\right)\\
&\triangleq RHS.
\end{aligned}$$

Assume $\epsilon_1 \leq \epsilon_2$ then,

$$RHS \leq \psi(\tilde{\mathbf{b}}^1_T, \tilde{\mathbf{b}}^2_T) - \psi(\mathbf{b}^1_*, \mathbf{b}^2_*) + 3\sqrt{\frac{1}{n}\sum_i (\max_j \varepsilon^1_{i,j})^2}\sqrt{1 + \frac{1}{T}\frac{9\rho n^2}{\lambda^2}} + \lambda\sum_l \left(\max_i |\varepsilon^2_{i,l}| + \max_i |\varepsilon^3_{i,l}|\right).$$

In Eq. (31) set

$$\kappa^1_{i,j} = \begin{cases} -\frac{3\sqrt{1+\frac{1}{T}\frac{9\rho n^2}{\lambda^2}}\varepsilon^1_{i,j}}{\sqrt{n\sum_i(\max_k \varepsilon^1_{i,k})^2}} & \text{if } j = \arg\max_k \varepsilon^1_{i,k} \\ 0 & \text{otherwise} \end{cases}$$
$$\kappa^2_{i,l} = \begin{cases} -\lambda \operatorname{sign} \varepsilon^2_{i,l} & \text{if } i = \arg\max_k |\varepsilon^2_{k,l}| \\ 0 & \text{otherwise} \end{cases}$$
$$\kappa^3_{i,l} = \begin{cases} -\lambda \operatorname{sign} \varepsilon^3_{i,l} & \text{if } i = \arg\max_k |\varepsilon^3_{k,l}| \\ 0 & \text{otherwise.} \end{cases}$$

We have

$$\begin{aligned} RHS = \Delta &\leq \frac{1}{T}\left(\frac{\rho}{2}\|\mathbf{B}\mathbf{b}^2_*\|^2 + \frac{9}{2n\rho}(1+\frac{1}{T}\frac{9\rho n^2}{\lambda^2}) + \frac{d}{\rho}\lambda^2\right) \\ &\leq \frac{1}{T}\left(\frac{9\rho n^2}{\lambda^2} + \frac{9}{2n\rho}(1+\frac{1}{T}\frac{9\rho n^2}{\lambda^2}) + \frac{d}{\rho}\lambda^2\right). \end{aligned}$$

Set $\rho = \frac{\sqrt{d}\lambda^2}{n}$, and then we get: $LHS \leq RHS = \Delta \leq \frac{6n\sqrt{d}}{T}$ if $\lambda \geq \frac{3}{\sqrt{2nd}}$ and $n\rho T \geq \frac{9}{2}$.

Now assume $\epsilon_1 \geq \epsilon_2$. We get:

$$RHS \leq \psi(\tilde{\mathbf{b}}^1_T, \tilde{\mathbf{b}}^2_T) - \psi(\mathbf{b}^1_*, \mathbf{b}^2_*) + 3\frac{1}{n}\sum_i(\max_j \varepsilon^1_{i,j})^2 + \lambda\sum_l \big(\max_i |\varepsilon^2_{i,l}| + \max_i |\varepsilon^3_{i,l}|\big).$$

In Eq. (31) set

$$\kappa^1_{i,j} = \begin{cases} -\frac{4}{n}\varepsilon^1_{i,j} & \text{if } j = \arg\max_k \varepsilon^1_{i,k} \\ 0 & \text{otherwise} \end{cases}$$
$$\kappa^2_{i,l} = \begin{cases} -\lambda \operatorname{sign} \varepsilon^2_{i,l} & \text{if } i = \arg\max_k |\varepsilon^2_{k,l}| \\ 0 & \text{otherwise} \end{cases}$$
$$\kappa^3_{i,l} = \begin{cases} -\lambda \operatorname{sign} \varepsilon^3_{i,l} & \text{if } i = \arg\max_k |\varepsilon^3_{k,l}| \\ 0 & \text{otherwise.} \end{cases}$$

We get

$$\begin{aligned} \Delta &= \psi(\tilde{\mathbf{b}}^1_T, \tilde{\mathbf{b}}^2_T) - \psi(\mathbf{b}^1_*, \mathbf{b}^2_*) + 4\frac{1}{n}\sum_i(\max_j \varepsilon^1_{i,j})^2 + \lambda\sum_l \big(\max_i |\varepsilon^2_{i,l}| + \max_i |\varepsilon^3_{i,l}|\big) \\ &\leq \frac{1}{T}\left(\frac{9\rho n^2}{\lambda^2} + \frac{16}{2\rho n^2}\sum_i(\max_j \varepsilon^1_{i,j})^2 + \frac{d}{\rho}\lambda^2\right). \end{aligned}$$

It's straightforward to see if $n\rho T \geq 8$ we have

$$LHS \leq RHS \leq \Delta - \frac{16}{2\rho n^2 T}\sum_i(\max_j \varepsilon^1_{i,j})^2 = O(\frac{1}{T}(\frac{\rho n^2}{\lambda^2} + \frac{d}{\rho}\lambda^2)).$$

Set $\rho = \frac{\sqrt{d}\lambda^2}{n}$ we get: $LHS \leq \frac{3n\sqrt{d}}{T}$. Only if $T \geq \frac{8}{\sqrt{d}\lambda^2}$. $\square$

C. Experimental results

This section provides the regression and classification experimental results in tabular format. Table 3 compares the regression performance of our method against baseline methods on benchmark datasets. Table 4 compares the classification accuracy of our method against baseline methods. We present the run time of our method on selected UCI datasets in Table 5. Here, we omit the baseline method because it did not run to completion on most of the datasets. Hyperparameters were fixed for all timing results.

Table 3. Comparison of Regression performance on UCI datasets.

			$R^2 \times 100$			
dataset	n	d	**dc regression**	lasso	random forest	xgboost
Parkinson Speech Dataset	702	52	-3.7	-4	-14.8	-20
Garment Productivity	905	37	25.5	15.2	45.9	44.2
Concrete Compressive Strength	1030	8	91.7	59.9	91.8	92.5
Geographical Original of Music-1	1059	68	21.9	16.6	25.1	26
Geographical Original of Music-2	1059	68	32.6	23.8	31.3	30.8
Solar Flare-1	1066	23	3.3	6.1	-28	6.2
Solar Flare-2	1066	23	-6.1	-2.1	-17	2.1
Airfoil Self-Noise	1503	5	95.2	51.1	93.3	95
Communities and Crime	1994	122	62.1	64.5	65.4	65.8
SML2010	3000	24	90.8	96	93.4	92.1
SML2010	3000	24	77.3	94	92.4	90
Parkinson’s Telemonitoring	4406	25	93.8	98.4	92.3	98.2
Parkinson’s Telemonitoring	4406	25	95.5	98.5	92	98
Wine Quality	4898	11	51.8	27.2	52.6	51
Bias Correction of Temperature Forecast-1	6200	52	62.9	60.8	64.2	65.8
Bias Correction of Temperature Forecast-2	6200	52	75.2	76.5	76.3	76.7
Seoul Bike Sharing Demand	6570	19	89.7	80.7	93.1	92.3
Air Quality-1	7110	21	87	89.5	88.2	89.9
Air Quality-2	7110	21	100	94.7	99.8	100
Air Quality-3	7110	21	86.6	86.4	87.7	88.5
Air Quality-4	7110	21	81.7	84	78.1	80.7
Combined Cycle Power Plant	9568	4	95.5	92.9	96.5	96.7

Table 4. Comparison of Classification performance on UCI datasets.

			Accuracy (%)				Run time of all fits (sec)	
dataset	n	d	**pbdl**	pbdl0	rand forest	xgboost	**pbdl**	pbdl0
Iris	149	4	96.0	98.7	96	96.6	22.1	46.2
Wine	178	13	96.0	96.6	96.7	95.5	23.6	496.2
Blood Transfusion	748	4	72.6	74.8	72.9	78.4	46.3	21514.0
Breast Cancer Wisconsin	569	30	94.4	96.5	96.1	96.0	114.0	224407
Balance Scale	625	4	84.6	93.0	83	92.2	150.3	617.0
Abalone	4177	10	22.6	N/A	24.9	26.2	3688.0	N/A

Table 5. DC Regression run time on UCI datasets. (Baseline is ommitted because it does not run to completion on most datasets)

dataset	n	d	This Paper (Seconds)
Parkinson Speech Dataset	702	52	3.4
Garment Productivity	905	37	4.12
Concrete Compressive Strength	1030	8	3.28
Geographical Original of Music	1059	68	4.44
Solar Flare	1066	23	3.64
Airfoil Self-Noise	1503	5	4.92
Communities and Crime	1994	122	12.82
SML2010	3000	24	21.78

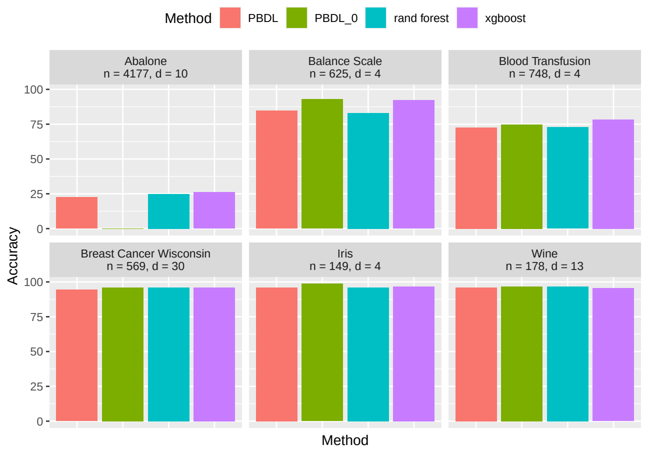

Figure 3. Classification accuracy for all fits on UCI datasets