PtO DCA

Hong-Seok Choe

1 Introduction

1.1 Predict then Optimize

- Optimization oracle:
$$w^*(c) = \underset{w \in S}{\operatorname{argmin}}\ c^T w$$
- S: non-empty, compact, and convex
- c: true cost vector
- Candidate Regret Losses: Choose (1) over (2) because it captures the better model discrepancy.
$$\ell_{SPO}(\hat{c}, c) = c^T w^*(\hat{c}) - c^T w^*(c) \tag{1}$$
$$\ell_{SPO}(\hat{c}, c) = \hat{c}^T w^*(\hat{c}) - \hat{c}^T w^*(c) \tag{2}$$

2 Related Works

Ultimate Goal: Compute gradient of loss functions that can be leveraged into neural nets?

- Donti et al. Task-based End-to-end Model Learning in Stochastic Optimization
  - Key Idea 1 (Algorithmic): If any of the inequality constraints in the loss function are violated, we take a gradient step in the violated constraint; otherwise, we take a gradient step in the optimization objective.
  - Key Idea 2 (Analytical): Computing the gradient of an objective that depends upon the argmin operation from KKT optimality conditions.
- Wilder et al. Melding the Data-Decisions Pipeline: Decision-Focused Learning for Combinatorial Optimization
  - Key Idea: Add quadratic regularization term to the LP objective with and compute gradient using the idea in Donti et al.
- Mandi et al. Interior Point Solving for LP-based prediction + optimisation
  - Key Idea: Instead of QP penalty term, use logarithmic barrier term with homogeneous self-dual embedding to solve for gradients in the learning step
- Berthet et al. Learning with Differentiable Perturbed Optimizers
-

3 Observations

$$\ell_{SPO}(c,\hat{c}) = \inf_{\alpha\geq 0}\{\xi_S(c-\alpha\hat{c}) + \alpha z^*(\hat{c})\} - z^*(c)$$

- Support function $\xi_S(\cdot):\mathbb{R}^d \to \mathbb{R}$ of a (compact) set S:

$$\xi_S(c) := \max_{w\in S} c^T w$$

- $z^*(c) = -\xi_S(-c)$
- DC Structure when $\alpha \geq 0$ fixed
  - Fixing $\hat{c}$ (Convex):

$$\ell_{SPO}(c,\hat{c}) \approx \xi_S(c-\alpha\hat{c}) + \xi_S(-c)$$

  - Fixing c (DC):

$$\ell_{SPO}(c,\hat{c}) \approx \xi_S(c-\alpha\hat{c}) - \alpha\xi_S(-\hat{c})$$

4 Approaches

- Approach 1. Analytically compute/approximate gradient that can be used in gradient algorithms or embedded in backpropagation step in neural nets as in the survey paper

$$\frac{\partial \ell_{SPO}(c,\hat{c})}{\partial \hat{c}} = \frac{\partial \ell_{SPO}(c,\hat{c})}{\partial w^*(\hat{c})} \cdot \frac{\partial w^*(\hat{c})}{\partial \hat{c}}$$

  Find (LHS) analytically also considering the limit of α that goes to infinity.

  Using monotone convergence theorem, it turns out that

$$\ell_{SPO}(c,\hat{c}) = \lim_{\alpha\to\infty}\{\xi_S(c-\alpha\hat{c}) + \alpha z^*(\hat{c})\} - z^*(c)$$

  Suppose that we have a function g defined as

$$g(x,\alpha) = \lim_{\alpha\to+\infty}\{f(k-\alpha x) - f(-\alpha x)\} := \lim_{\alpha\to+\infty} F_\alpha(x)$$

  with f convex and k constant. The goal is to find $\frac{\partial g(x,\alpha)}{\partial x}$ (maybe an approximate one w.r.t f', ∇f, or ∂f). Can we argue something about uniform convergence using convexity so that we can use the following theorem

$$\frac{\partial}{\partial x}\lim_{\alpha\to\infty} F_\alpha(x) = \lim_{\alpha\to\infty}\frac{\partial F_\alpha(x)}{\partial x} \tag{1}$$

  If so, what are the analytic requirements and is it valid to consider partial derivative approximates of RHS and then take a limit?

  Other alternative approaches not using equation (1). Use the fact that $q(\alpha) := f(k-\alpha x) - f(-\alpha x)$ is monotone decreasing function.

- Approach 2. Danskin's Theorem and DCA

  Suppose $\phi(x,w) := x^T w$ is a continuous function of two arguments

$$\phi: \mathbb{R}^n \times S \to \mathbb{R}$$

where $S \subset \mathbb{R}^n$ is a compact set. $\phi(x, w)$ is convex in x for every $w \in S$. Define $\xi_S(x) = \max_{w \in S} \phi(x, w)$ and

$$S_0(x) = \left\{ \overline{w} : \phi(x, \overline{w}) = \max_{w \in S} \phi(x, w) \right\} \tag{2}$$

Danskin's theorem: $\xi_S(x)$ is differentiable at x if $S_0(x)$ consists of a single element $\overline{w}$. Furthermore, the derivative of $\xi_S(x)$ is given by

$$\frac{\partial \xi_S}{\partial x} = \frac{\partial \phi(x, \overline{w})}{\partial x} = \overline{w}.$$

If $\phi(x, w)$ is differentiable with respect to x for all $w \in S$, and if $\partial \phi / \partial x$ is continuous with respect to w for all S, then the subdifferential of $\xi_S(x)$ is given by

$$\partial \xi_S(x) = \operatorname{conv} \left\{ \frac{\partial \phi(x, w)}{\partial x} : w \in S_0(x) \right\}$$

- $\ell_{SPO}(c, \hat{c}) = \xi_S(c - \alpha \hat{c}) - \alpha \xi_S(-\hat{c}) := g(\hat{c}) - h(\hat{c})$
- $S_0(\hat{c})$ may not have a single element (i.e., a unique maximizer)
- $\partial g(\hat{c}) = \operatorname{conv} \{-\alpha w : w \in S_0(c - \alpha \hat{c})\}$
- $\partial h(\hat{c}) = \alpha \cdot \operatorname{conv} \{-w : w \in S_0(-\hat{c})\}$
- The maximizer of a convex function over a convex set S lies on the boundary of the S, More precisely, the collection of maximizers solving (2) is an *exposed face* of S). We can get rid of the convex hull part.

$$\begin{aligned} \partial g(\hat{c}) &= \{-\alpha w : w \in S_0(c - \alpha \hat{c})\} \\ \partial h(\hat{c}) &= \{-\alpha w : w \in S_0(-\hat{c})\} \end{aligned}$$

- Computing ∂g:

$$(-\alpha) \cdot \arg\max_{w \in S} (c - \alpha \hat{c})^T w$$

  or

$$-\alpha w^*(-c + \alpha \hat{c}) := (-\alpha) \cdot \arg\min_{w \in S} (-c + \alpha \hat{c})^T w$$

- Computing ∂h:

$$(-\alpha) \cdot \arg\max_{w \in S} (-\hat{c})^T w$$

  or

$$-\alpha w^*(\hat{c}) := (-\alpha) \cdot \arg\min_{w \in S} \hat{c}^T w$$

4.1 DCA Algorithm

- Idea: Alternating $\{x_k\}$, $\{y_k\}$

$$\alpha = \begin{cases} \inf_{x \in X} \{g(x) - h(x)\} \rightsquigarrow \arg\min x^* \\ \inf_{y \in Y} \{h^*(y) - g^*(y)\} \rightsquigarrow \arg\min y^* \end{cases}$$

- Choose feasible starting point $x_0 \in \operatorname{dom} g$: Start with an arbitrary value or maybe a clever heuristics/ideas for better performance?

- (1) $x_k \in \partial g^*(y_{k-1})$

$$\begin{aligned} g^*(y) &\geq g^*(y_{k-1}) + (y - y_{k-1})^T x_k \\ h^*(y) - g^*(y) &\leq h^*(y) - g^*(y_{k-1}) - (y - y_{k-1})^T x_k \end{aligned}$$

- (2) Minimize (RHS)

$$y_k = \arg\min_{y \in Y} \left\{ h^*(y) - g^*(y_{k-1}) - (y - y_{k-1})^T x_k \right\} = \arg\max_{y \in Y} \left\{ x_k^T y - h^*(y) \right\}$$

  Notice that

$$h(x_k) = \sup_{y \in Y} \left\{ x_k^T y - h^*(y) \right\} = x_k^T y_k - h^*(y_k)$$

  and

$$\begin{aligned} h(x) &= \sup_{y \in Y} \left\{ x^T y - h^*(y) \right\} \\ &\geq x^T y_k - h^*(y_k) \\ &= (x - x_k)^T y_k + x_k^T y_k - h^*(y_k) \\ &= (x - x_k)^T y_k + h(x_k) \end{aligned}$$

  Thus, y_k is a subgradient of h at x_k.

- (3) Similarly, $y_k \in \partial h(x_k)$ implies

$$\begin{aligned} h(x) &\geq h(x_k) + (x - x_k)^T y_k \\ g(x) - h(x) &\leq g(x) - h(x_k) - (x - x_k)^T y_k \end{aligned}$$

- (4) Minimize (RHS)

$$x_{k+1} = \arg\min_{x \in X} \left\{ g(x) - h(x_k) - (x - x_k)^T y_k \right\} \in \partial g^*(y_k) \tag{3}$$

- Brute-force computation of $\partial h(x_k)$: Simplification?

$$\begin{aligned} y_k &= \arg\max_{y \in Y} \left\{ x_k^T y - h^*(y) \right\} \\ &= \arg\max_{y \in Y} \left\{ x_k^T y - \sup_{x \in X} \left\{ x^T y - h(x) \right\} \right\} \\ &= \arg\max_{y \in Y} \left\{ x_k^T y - \sup_{x \in X} \left\{ x^T y - \max_{w \in S} (-\alpha x)^T w \right\} \right\} \end{aligned}$$

- Brute-force computation of $\partial g^*(y_k)$

$$\begin{aligned} x_{k+1} &= \arg\max_{x \in X} \left\{ x^T y_k - g(x) \right\} \\ &= \arg\max_{x \in X} \left\{ x^T y_k - \left\{ \max_{w \in S} (c - \alpha x)^T w \right\} \right\} \\ &= \arg\max_{x \in X} \left\{ \min_{w \in S} \left\{ x^T y_k - c^T w + \alpha x^T w \right\} \right\} \end{aligned}$$

  Computationally tractable: Reduces to finding arguments of the following maximin problem (can swap max and min using Sion's Minimax Theorem?)

$$\begin{aligned} \max_{x \in \mathbb{R}^n} \min_{w \in S} v(x, w) := &\max_{x \in \mathbb{R}^n} \min_{w \in S} \left\{ x^T y_k - c^T w + \alpha x^T w \right\} \\ &\min_{w \in S} \max_{x \in \mathbb{R}^n} \left\{ x^T y_k - c^T w + \alpha x^T w \right\} \end{aligned}$$

4.2 Notes on Conjugacy

- Scalar multiplication ($\alpha > 0$):

$$\begin{aligned} f(x) = \alpha g(x) &\iff f^*(y) = \alpha g^*(y/\alpha) \\ f(x) = \alpha g(x/\alpha) &\iff f^*(y) = \alpha g^*(y) \end{aligned}$$

- Affine Transformation:

$$\begin{aligned} f(x) = g(x) + a^T x + b &\iff f^*(y) = g^*(y - a) - b \\ f(x) = g(x - b) &\iff f^*(y) = b^T y + g^*(y) \end{aligned}$$

  If there is no restriction on α,

$$\begin{aligned} g^*(y) &= I_S\left(\frac{y}{(-\alpha)}\right) - \frac{1}{(-\alpha)} c^T y \\ h^*(y) &= I_S\left(\frac{y}{(-\alpha)}\right) \end{aligned}$$

  where $I_S(\cdot)$ is a $0-\infty$ convex indicator function. This does not work because $-\alpha < 0$ unless S is given symmetric.

- Computing subgradient of support function without using Danskin's Theorem:

$$\begin{aligned} g(x) &= \max_{w \in S} (c - \alpha x)^T w \\ \bar{w}_x &:= \arg\max_{w \in S} (c - \alpha x)^T w \end{aligned}$$

  Claim) $-\alpha \bar{w}_{\hat{x}}$ is a subgradient of g at $\hat{x}$

$$\begin{aligned} g(x) - g(\hat{x}) &= \max_{w \in S} (c - \alpha x)^T w - \max_{w \in S} (c - \alpha \hat{x})^T w \\ &= \max_{w \in S} (c - \alpha x)^T w - (c - \alpha \hat{x})^T \bar{w}_{\hat{x}} \\ &\geq (c - \alpha x)^T \bar{w}_{\hat{x}} - (c - \alpha \hat{x})^T \bar{w}_{\hat{x}} = (x - \hat{x})^T (-\alpha \bar{w}_{\hat{x}}) \end{aligned}$$

  Similarly, $-\alpha \cdot \arg\max_{w \in S} (-x)^T w$ is a subgradient of h at x

Theorem 1. *If f is closed and convex, then*

$$y \in \partial f(x) \iff x \in \partial f^*(y) \iff x^T y = f(x) + f^*(y)$$

and

$$x \in \arg\max_x \{y^T x - f(x)\} = \partial f^*(y)$$

- Support function is closed and convex (proper convex and lower semicontinuous $\implies$ closed)

$$\begin{aligned} g^*(y) &= x^T y - g(x), & \partial g^*(y) &= \arg\max_x \{y^T x - g(x)\} \\ h^*(y) &= x^T y - h(x), & \partial h^*(y) &= \arg\max_x \{y^T x - h(x)\} \end{aligned}$$

4.3 DCA Pseudocode (Minimization w.r.t. 1-D Vector)

Algorithm 1 DC Algorithm

```
1: Choose $x_0 \in \text{dom } g$
2: for $k \in N$ do
3:     Choose $y_k \in \partial h(x_k)$
4:     Choose $x_{k+1} \in \partial g^*(y_k)$
5:     if $\min\left\{|(x_{k+1} - x_l)_i|, \left|\frac{(x_{k+1}-x_l)_i}{(x_k)_i}\right|\right\} \leq \delta$ then
6:         return $x_{k+1}$
7:     end if
8: end for
```

1. Choose $x_0 \in \text{dom } g$:

   Start with $\hat{c}$ that is obtained using simple linear regression (i.e., least squares method) or the realized cost vector c in experimental cases

3. Choose $y_k \in \partial h(x_k)$:

   Find $y_k = (-\alpha) \cdot \arg\min_{w \in S} x_k^T w$

4. Choose $x_{k+1} \in \partial g^*(y_k)$:

   - Find $x_{k+1} = \arg\max_{x \in \mathbb{R}^n} \left\{x^T y_k - g(x)\right\} = \arg\min_{x \in \mathbb{R}^n} \left\{g(x) - x^T y_k\right\}$:
     Apply subgradient method to $\min_x\{g(x) - y_k^T x\}$. Involves finding $\partial(g(x) - y_k^T x) = (-\alpha) \cdot \arg\min_{w \in S}(\alpha x - c)^T w - y_k$ at each sub-iteration.
   - Sanity check: $y_k \in \partial g(x_{k+1}) = (-\alpha) \cdot \arg\max_{w \in S}(c - \alpha x_{k+1})^T w$ (Or maybe find x_{k+1} s.t. $y_k \in \partial g(x_{k+1})$ directly?)

- Optimization oracle $w^*(\cdot)$:

  For every c, the oracle returns $\min_{w \in S} c^T w$ and $\arg\min_{w \in S} c^T w$ in the actual implementation

4.4 Computational Experiment: Shortest Path Problem

- Linear hypothesis class $\mathcal{H} = \{f : f(x) = Bx \text{ for some } B \in \mathbb{R}^{d \times p}\}$
- 5×5 grid with 40 vertical and horizontal edges
- Dimension of cost vector $d = 40$ (corresponds to each edge)
- Number of features $p = 5$
- True parameter $B^* \in \mathbb{R}^{d \times p}$ where each entry $(B)_{ij} \sim Bern(1/2)$
- Training data $\{(x_i, c_i)\}_{i=1}^n$ and test data $\{(\tilde{x}_i, \tilde{c}_i)\}_{i=1}^{n_{\text{test}}}$ generation:
  - Feature vector $x_i \in \mathbb{R}^p$: follows i.i.d. multivariate Gaussian $x_i \sim \mathcal{N}(0, I_p)$
  - Cost vector $c_i \in \mathbb{R}^d$: for $j = 1, \ldots, d$, $c_{ij} = \left[\left(\frac{1}{\sqrt{p}}(B^* x_i)_j + 3\right)^{\text{deg}}\right] \cdot \epsilon_i^j$, where deg is a fixed positive integer parameter and $\epsilon_i^j \sim U[1 - \bar{\epsilon}, 1 + \bar{\epsilon}]$ i.i.d. for some parameter $\bar{\epsilon}$
- $pd = 200$ parameters to estimate, $n \in \{100, 1000, 5000\}$, $\text{deg} \in \{1, 2, 4, 6, 8\}$, $\bar{\epsilon} \in \{0, 0.5\}$
- For every $(n, \text{deg}, \bar{\epsilon})$ triplet, run 50 simulations
- For each simulation, evaluate the performance by computing the normalized SPO loss on a test set ($n_{\text{test}} = 10000$):

$$\text{NormSPOTest}(\hat{f}) := \frac{\sum_{i=1}^{n_{\text{test}}} \ell_{SPO}(\hat{f}(\tilde{x}_i), \tilde{c}_i)}{\sum_{i=1}^{n_{\text{test}}} z^*(\tilde{c}_i)} = \frac{\sum_{i=1}^{n_{\text{test}}} \{\tilde{c}_i^T w^*(\hat{B}\tilde{x}_i) - \tilde{c}_i^T w^*(\tilde{c}_i)\}}{\sum_{i=1}^{n_{\text{test}}} \tilde{c}_i^T w^*(\tilde{c}_i)}$$

4.5 Applying DCA to SPO Loss

$$\min_B \frac{1}{n}\sum_{i=1}^{n}\{\xi_S(c_i - \alpha Bx_i) - \alpha\xi_S(-Bx_i)\} := \min_B \frac{1}{n}\sum_{i=1}^{n} L_i(B)$$

Applying DCA to $L_i(B)$ to minimize loss w.r.t. matrix B. How can we minimize the sum of L_i's knowing each minimizer (Something similar to the batch-type SGD?) $\implies$ Stochastic DCA?

Straightforward extension hold when we apply DCA w.r.t matrices (exact same analysis) : can use alternating linear minorization algorithm

$$\mathcal{L}(B) = \frac{1}{n}\sum_{i=1}^{n}\xi_S(c_i - \alpha Bx_i) - \frac{1}{n}\sum_{i=1}^{n}\alpha\xi_S(-Bx_i) := \mathcal{G}(B) - \mathcal{H}(B)$$
$$\partial\mathcal{G}(B) = \frac{1}{n}\sum_{i=1}^{n}\partial G_i(B) = \frac{1}{n}\sum_{i=1}^{n}\partial g(Bx_i)$$
$$\partial\mathcal{H}(B) = \frac{1}{n}\sum_{i=1}^{n}\partial H_i(B) = \frac{1}{n}\sum_{i=1}^{n}\partial h(Bx_i)$$

Applying DCA to $\mathcal{L}(B)$ or find another pair of functions s.t $\mathcal{L}(B) \le \mathcal{G}(B) - \mathcal{H}(B)$ using known inequalities

- Choose $B_0 \in \text{dom } G$: Initialize $B_0 \in \mathbb{R}^{d\times p}$ (typically $B_0 \leftarrow 0$?)
- Choose $C_k \in \partial\mathcal{H}(B_k)$:

  Use the same argument as in the original paper

$$h(Bx_i) \ge h(B'x_i) + (-\alpha\bar{w}_{B'x_i})^T(Bx_i - B'x_i)$$
$$H(B) \ge H(B') + (-\alpha\bar{w}_{B'x_i}x_i^T) \bullet (B - B')$$

  Take $C_k \in \partial\mathcal{H}(B_k)$ where

$$\frac{1}{n}\sum_{i=1}^{n}(-\alpha \arg\max_{w\in S}(-B_kx_i)^T w)x_i^T = \frac{1}{n}\sum_{i=1}^{n}(-\alpha w^*(B_kx_i))x_i^T \subseteq \partial\mathcal{H}(B_k)$$

  where ab^T denotes the outer product of a and b.

  Similarly,

$$\frac{1}{n}\sum_{i=1}^{n}(-\alpha \arg\max_{w\in S}(c_i - \alpha Bx_i)^T w)x_i^T = \frac{1}{n}\sum_{i=1}^{n}(-\alpha w^*(-c_i + \alpha Bx_i))x_i^T \subseteq \partial\mathcal{G}(B)$$

- Choose $B_{k+1} \in \partial\mathcal{G}^*(C_k)$:

  Analogous to finding $x_{k+1} = \arg\min_{x\in\mathbb{R}^n}\left\{g(x) - x^T y_k\right\}$

  Let $B^* = \arg\max_B\{\langle B, C'\rangle - \mathcal{G}(B)\}$

$$\begin{aligned}\sup_B\{\langle B, C\rangle - \mathcal{G}(B)\} &\ge \langle B^*, C\rangle - \mathcal{G}(B^*)\\ &= \langle B^*, C\rangle - \mathcal{G}(B^*) + \langle B^*, C'\rangle - \langle B^*, C'\rangle\\ &= \sup_B\{\langle B, C'\rangle - \mathcal{G}(B)\} + \langle B^*, C - C'\rangle\\ \iff \mathcal{G}^*(C) &\ge \mathcal{G}^*(C') + \langle B^*, C - C'\rangle\end{aligned}$$

  so B^* is a subgradient of G^* at C'

  Find $B_{k+1} = \arg\min_{B\in\mathbb{R}^{d\times p}}\{\mathcal{G}(B) - \langle B, C_k\rangle\}$

Subgradient method involves finding $\partial(\mathcal{G}(B) - \langle C_k, B\rangle) = \partial\mathcal{G}(B) - C_k$ at each sub-iteration.

$$\mathcal{G}(B) - \langle C_k, B\rangle = \frac{1}{n}\sum_{i=1}^{n}\left\{G_i(B) - \langle(-\alpha w^*(Bx_i)x_i^T, B\rangle\right\}$$
$$\partial(\mathcal{G}(B) - \langle C_k, B\rangle) = \frac{1}{n}\sum_{i=1}^{n}\left\{-\alpha w^*(-c_i + \alpha Bx_i)x_i^T - (-\alpha w^*(Bx_i)x_i^T\right\}$$

Applying regularization: Same logic as in the proof in the Appendix
Further analysis working with inner product space and Hermitian matrices: See Adrian's paper

$$F^*(Y) = \sup_{X\in\mathcal{X}}\{\langle X, Y\rangle - F(X)\}$$

When F is proper and convex,

$$\partial F(X) = \{Y \in \mathcal{X} : F(X) + F^*(Y) = \langle X, Y\rangle\}$$

Algorithm 2 SPO DCA

```
 1: Initialize $B_0 \in \mathbb{R}^{d\times p}$, $\gamma$ (step size), $\delta$ (error tolerance)
 2: for $k = 0, 1, 2, \dots$ do
 3:     $C_k \leftarrow \frac{1}{n}\sum_{i=1}^{n}(-\alpha w^*(B_k x_i))x_i^T$
 4:     $B'_0 \leftarrow B_k$
 5:     for $l = 0, 1, 2, \dots$ do                                   ▷ Simple Subgradient Descent
 6:         $B'_{l+1} \leftarrow B'_l - \gamma\left(\frac{1}{n}\sum_{i=1}^{n}(-\alpha w^*(\alpha B'_l x_i - c_i))x_i^T - C_k\right)$
 7:         if $\|B'_{l+1} - B'_l\|_F < \delta$ (or other stopping rule) then
 8:             Stop. $B_{k+1} \leftarrow B'_{l+1}$
 9:         end if
10:     end for
11:
12:     if $\|B_{k+1} - B_k\|_F < \delta$ (or other stopping rule) then
13:         return $B_{k+1}$
14:     end if
15: end for
```

Algorithm 3 SPO+ Subgradient

```
1: Initialize $B_0 \in \mathbb{R}^{d\times p}$, $\gamma$ (step size), $\delta$ (error tolerance)
2: for $k = 0, 1, 2, \dots$ do                                       ▷ Simple Subgradient Descent
3:     $B_{k+1} \leftarrow B_k - \gamma\left(\frac{1}{n}\sum_{i=1}^{n} 2(w^*(c_i) - w^*(2B_k x_i - c_i))x_i^T\right)$
4:     if $\|B_{k+1} - B_k\|_F < \delta$ (or other stopping rule) then
5:         return $B_{k+1}$
6:     end if
7: end for
```

- Initialization $B_0 \leftarrow 0$ not converging in DCA
- Step size rule choosing γ ?
- Stopping rule w.r.t. objective values instead of Frobenius norm between consecutive iterates
- For evaluation, use the true SPO loss: $\ell_{SPO}^{w^*}(\hat{c}, c) := c^T w^*(\hat{c}) - z^*(c)$
- Apples to apples comparison: common subgradient (naïve and SGD) subroutine?

- Fix C. We want to solve

$$\min_B\{\mathcal{G}(B) - \langle B, C\rangle\}$$

  Consider

$$\min_{x\in\mathbb{R}}\left\{\max_{w\in S}\{(c-\alpha x)w\} - xy\right\}$$

  with $y=1$, $c=0$, $\alpha=2$, and $S=[1,2]$. The problem is unbounded? We may found upper bounds of the objective function so that these extreme cases do not occur.

Some Comments

- Resemblance in subgradient steps ($C_k \iff 2w^*(c_i)$): coincides with the linearizing intuition
- TO-DO: Regularization and Cross-Validation via `scikitlearn` (numpy reshape to 1-d vector?)
- Local optimality issue (DCA does not guarantee convergence to global optimum)
- Reduced tolerance to `1e-5`
- Faster Convergence with adaptive step size?? (design different step size or use SGD)
- The α-invariance does not hold with DCA objective (why?)

$$\min_B \frac{1}{n}\sum_{i=1}^{n}\{\xi_S(c_i - \alpha Bx_i) - \alpha\xi_S(-Bx_i)\} \tag{DCA}$$

$$\min_B \frac{1}{n}\sum_{i=1}^{n}\left\{\xi_S(c_i - \alpha Bx_i) - \alpha(Bx_i)^T w^*(c_i)\right\} \tag{SPO+}$$

  Since $\alpha > 0$, $\alpha\xi_S(-Bx_i) = \xi_S(-\alpha Bx_i)$

  When α increases, DCA yields higher normed SPO loss than that of SPO+ (tested with $\alpha = 4, 10, 100$)

  ~~Implications? Can we view α as a hyperparameter? Can we tune or find the optimal α?~~

- Observation 1: SPO+ training loss mostly 0 in naïve setting (when $deg \neq 1$ and $eps \neq 0$, it does not get 0)
- Observation 2: When $\alpha = 2$, DCA performs better than SPO+ with lower normed SPO loss with $\gamma_{DCA} = 0.1$, $\gamma_{SPO+} = 0.5$, $deg = 1$, $eps = 0$.
- Observation 3: Converges to different optimal matrix with different γs; Same training set, but different SPO losses.
- Strongly convex objective function in DCA: consider the functions $g(x) := f_1(x) + \frac{\rho\|x\|^2}{2}$ and $h(x) := f_2(x) + \frac{\rho\|x\|^2}{2}$. Then g and h are strongly convex functions with modulus ρ.

  $\implies$ faster convergence with higher degrees
- Simple SGD applied to the inner subgradient optimization step does not guarantee convergence. Stochastic version of DCA (Le Thi et al., 2019) in the following setting:

$$F(x) = G(x) - H(x) = \frac{1}{n}\sum_{i=1}^{n} g_i(x) - \frac{1}{n}\sum_{i=1}^{n} h_i(x)$$

Algorithm 4 Stochastic DC Algorithm (SDCA)

```
1: **Initialization** Choose $x_0 \in \mathbb{R}^d$, $s_0 = \{1, \dots, n\}$, and $k \leftarrow 0$
2: **repeat**
3:     1. Compute $y_i^k \in \partial h_i(x^k)$ if $i \in s_k$ and keep $y_i^k = y_i^{k-1}$ if $i \neq s_k$, $k > 0$. Set $y^k = \frac{1}{n}\sum_{i=1}^{n} y_i^k$
4:     2. Compute $x^{k+1} \in \partial G^*(y^k) = \min_x\{G(x) - \langle y^k, x\rangle\}$
5:     3. Set $k \leftarrow k+1$ and randomly choose a small subset $s_k \subset \{1, \dots, n\}$
6: **until** Stopping criteria
```

- DC proximal point algorithm (Sun et al., 2001):

$$f(x) = g(x) - h(x)$$

  Idea: Increasing the function h along the direction of the subgradient and then decreasing the function g by proximal step.

  Obs: Minimizing $\hat{g}(x) - \hat{f}(x) = cg + \frac{1}{2}\|\cdot\|^2 - (ch + \frac{1}{2}\|\cdot\|^2)$ for $c > 0$

Algorithm 5 DC Proximal Point Algorithm

```
1: Step 1. Given an initial point $x_0$ and $c_0 > c > 0$. Set $k = 0$
2: Step 2. Compute $w_k \in \partial h(x_k)$ and set $y_k = x_k + c_k w_k$
3: Step 3. Compute $x_{k+1} = (I + c_k \partial g)^{-1}(y_k)$ by proximal point algorithm. i.e., solve
       $\min_x \{g(x) + \frac{1}{2c_k}\|x - y_k\|^2\}$
4: Step 4. If $x_{k+1} = x_k$ stop. Otherwise, $k := k + 1$ and return to step 2.
```

- Frank-Wolfe Type Method (Khamaru & Wainwright, 2019):

$$f^* := \min_{x \in \mathcal{C}} \{g(x) - h(x)\}$$

  1. The difference between $f = g - h$ is bounded below over range of closed convex set $\mathcal{C}$
  2. The function g is continuously differentiable, whereas the function h is convex and continuous

Algorithm 6 Frank-Wolfe Type Method

```
1: Given initial vector $x^0 \in \int(\mathcal{C})$
2: for $k = 1, \ldots, K$ do
3:     Choose any $u^k \in \partial h(x^k)$
4:     Compute $s^k := \arg\min_{s \in \mathcal{C}} \langle s, \nabla g(x^k) - u^k \rangle$.
5:     Define $d^k := s^k - x^k$ and $g^k := -\langle d^k, \nabla g(x^k) - u^k \rangle$.     ▷ (Frank-Wolfe Gap)
6:     Set $\gamma^k = \min\{\frac{g^k}{C_0}, 1\}$ for some $C_0 \geq \mathcal{C}_f$
7:     Update $x^{k+1} = x^k + \gamma^k d^k$
8: end for
```

- DC-ADMM (Sun et al., 2017)

  Direct application of ADMM often leads to difficult nonconvex subproblems: convexify the subproblems through a linearization technique as done in DCA

  Setting:

$$\min_{x,y} \{f(x) - h(x) + g(y) \text{ s.t. } Ax + By = c\}$$

Algorithm 7 DC-ADMM

```
1: Set: parameters $\alpha > 0$, functions $\Phi, \Psi$
2: Initialization: $x^0, y^0, p^0$
3: for $k = 0, 1, \ldots$ do
4:     $\mathcal{L}^k_\alpha(x, y, p) := f(x) - \langle v^k, z \rangle + g(y) + \langle p, Ax + By - c \rangle + \frac{\alpha}{2}\|Ax + By - c\|_2^2$ where $v^k \in \partial h(x^k)$
5:     $x^{k+1} = \arg\min_x \mathcal{L}^k_\alpha(x, y^k, p^k) + B_\phi(x, x^k)$
6:     $y^{k+1} = \arg\min_y \mathcal{L}^k_\alpha(x^{k+1}, y, p^k) + B_\phi(y, y^k)$
7:     $p^{k+1} = p^k + \alpha(Ax^{k+1} + By^{k+1} - c)$
8: end for
```

- In DC literatures, the minimization of the following is concerned:

$$\min_x \{f(x) + g(x) - h(x)\}$$

  where

  1. f, g, h are closed proper convex functions
  2. g is differentiable convex with Lipschitz continuous gradient:

$$\|\nabla g(x) - \nabla g(y)\|_2 \le L_g \|x - y\|_2$$

- Maybe use softmax? See Nesterov's Smoothing
- Convergence rate analysis: see Kurdyka-Łojasiewicz functions and related conditions
- Still very slow: check subgradient method implementation. Accelerated version / early stopping
- Maintaining a validation set: should we adopt averaged iterates to decide the final model for (non-stochastic) subgradient methods?
- Subgradient method convergence criteria: Can we use

$$|f_{best}^{(k+1)} - f_{best}^{(k)}| < \epsilon$$

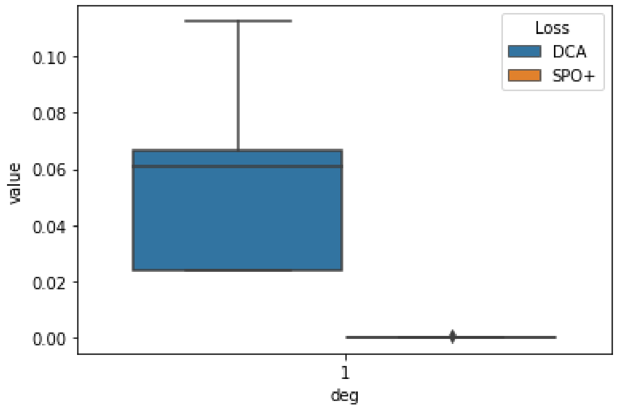
  i.e., only track the actual descents and examine the gap Or the gap between the subsequent best iterates $\|B_{best}^{(k+1)} - B_{best}^{(k)}\| < \epsilon$
  In the convergence rate proof:

$$f_{best}^{(k)} - f^* < \epsilon$$

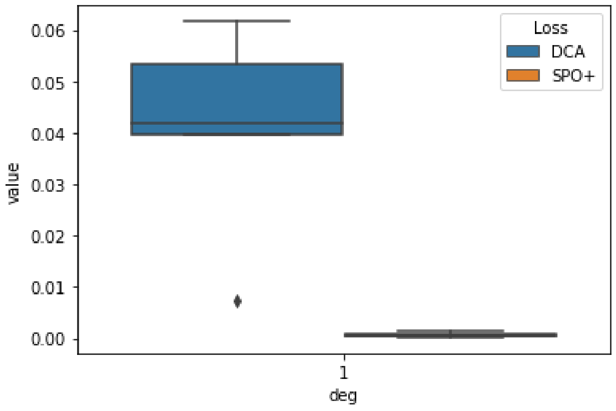
- Note that for a DC function $f = g - h$, a good convex minorization of f can be taken as $g + \text{co}(-h)$ where co stands for convex envelope. Knowing that the convex envelope of a concave function on a bounded polyhedral convex set can be easily computed.
- No good reference how the DCA performs with respect to the subroutine minimization: Assumes that the algorithm can find a new points $x^{k+1} \in \partial g^*(y^k)$

  Proof using ϵ-subdifferential in the subproblem?
- Caveat for DCA is that we want to obtain x^* such that $\partial h(x^*) \subset \partial g(x^*)$: "It is worth noting that if the simplified DCA terminates at some point x^* for which $\partial h(x^*)$ is not contained in $\partial g(x^*)$, then one can reduce the objective function value by restarting it from a new initial point $x_0 = x^*$ with $y_0 \in \partial h(x_0)$ such that $y_0 \notin \partial g(x_0)$." $\implies$ DCA re-training?
- Algorithms: Successive Convex Approximation (SCA)
- Algorithms: Concave-Convex Procedure (CCP)

Q. Any instances that DCA can beat SPO+? Complex structure in the problem that DCA can do better

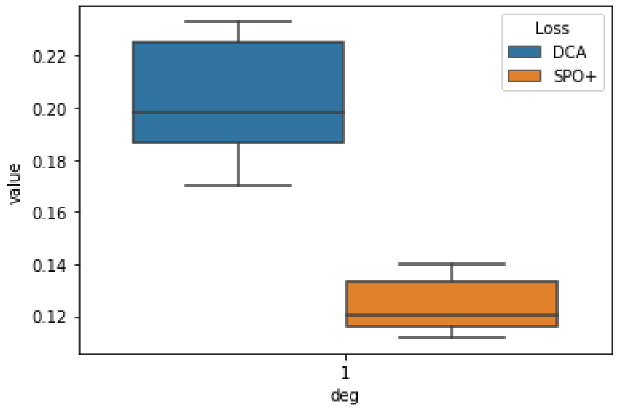


(a) Strong Convexity Trick

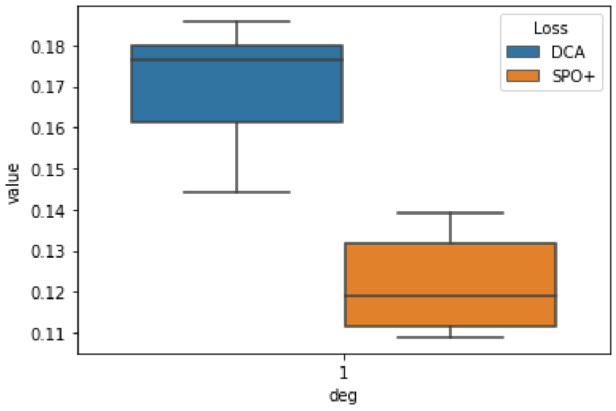


(b) Proximal Trick

Figure 1: $deg = 1$, $\epsilon = 0$

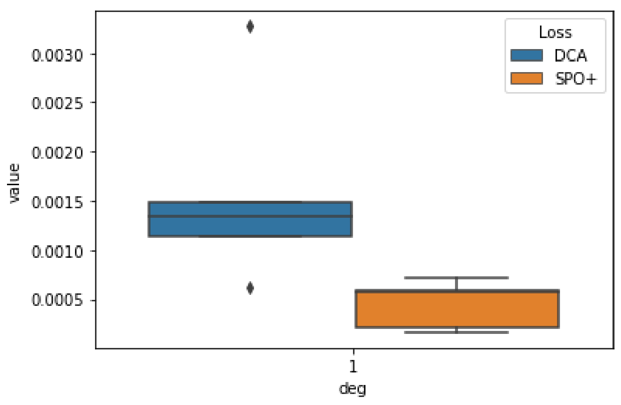


(a) Strong Convexity Trick

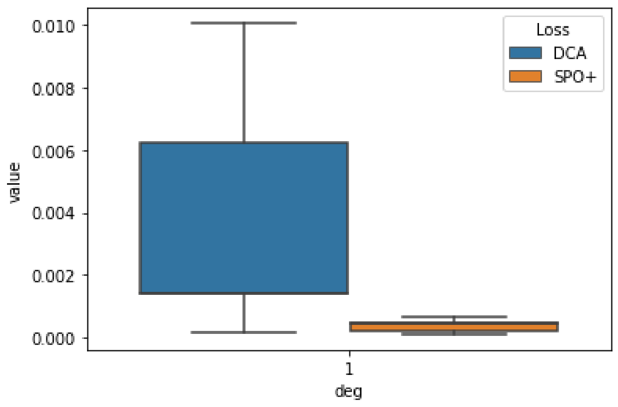


(b) Proximal Trick

Figure 2: $deg = 1$, $\epsilon = 0.5$

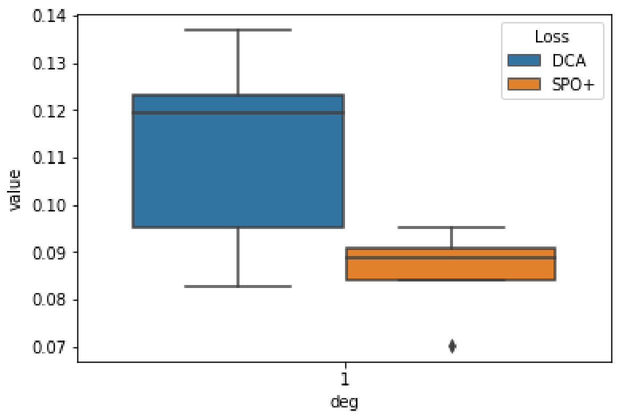


(a) Strong Convexity Trick

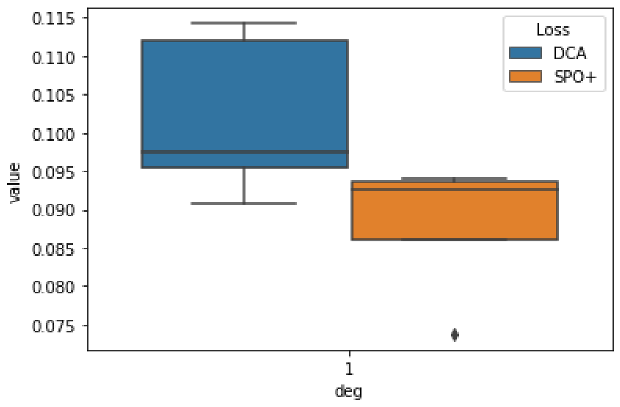


(b) Proximal Trick

Figure 3: $deg = 2$, $\epsilon = 0$

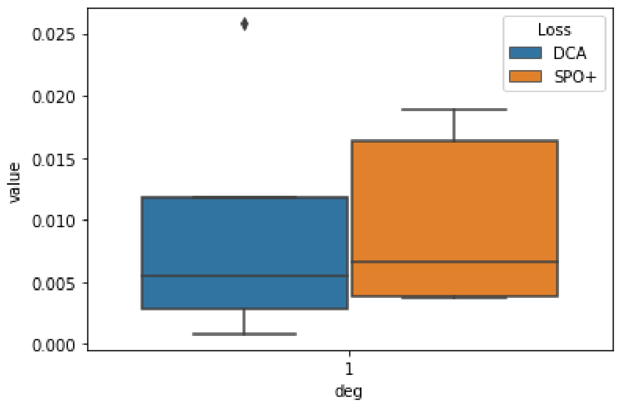


(a) Strong Convexity Trick

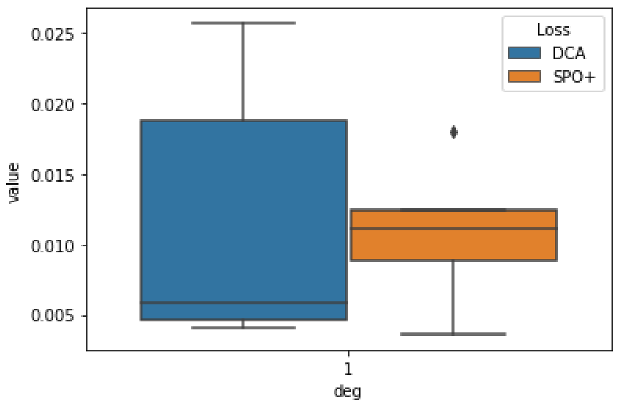


(b) Proximal Trick

Figure 4: $deg = 2$, $\epsilon = 0.5$

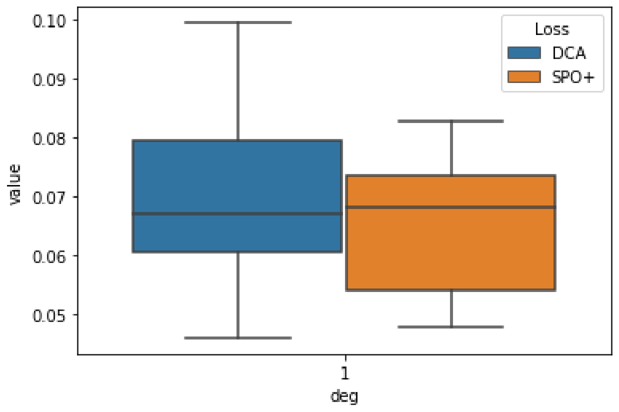


(a) Strong Convexity Trick

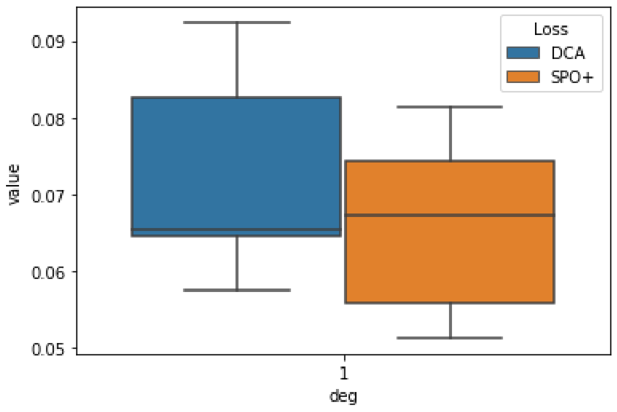


(b) Proximal Trick

Figure 5: $deg = 4$, $\epsilon = 0$



(a) Strong Convexity Trick



(b) Proximal Trick

Figure 6: $deg = 4$, $\epsilon = 0.5$

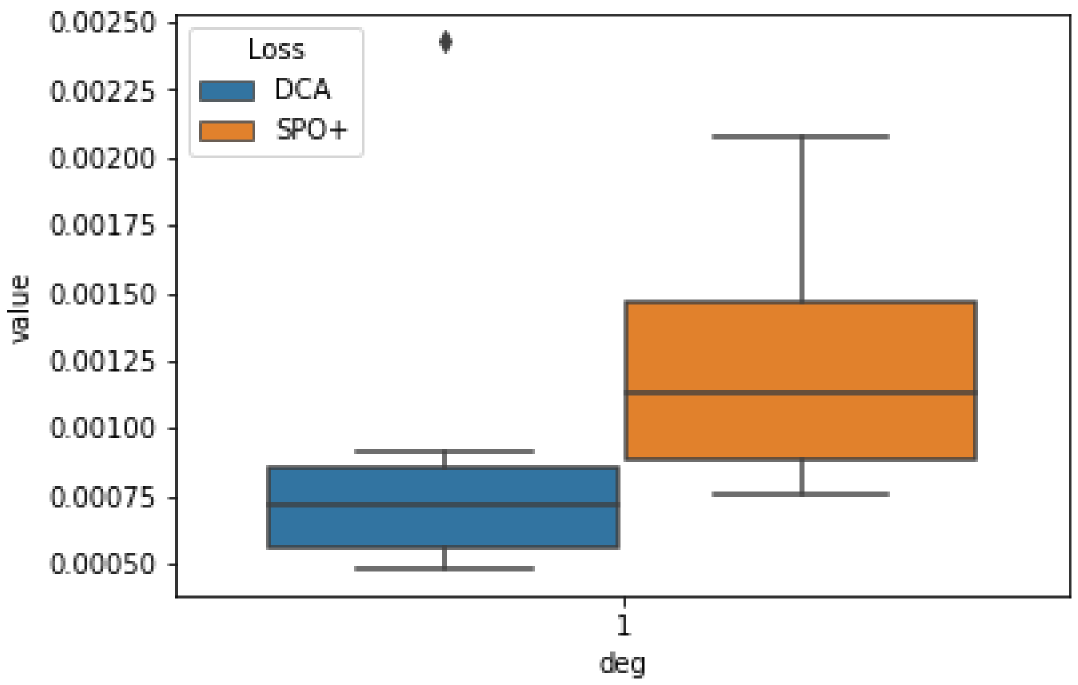


Figure 7: DCA can beat SPO+ when $\alpha = 2$ (10 simulations, $\gamma_{DCA} = 0.1$, $\gamma_{SPO+} = 0.5$)

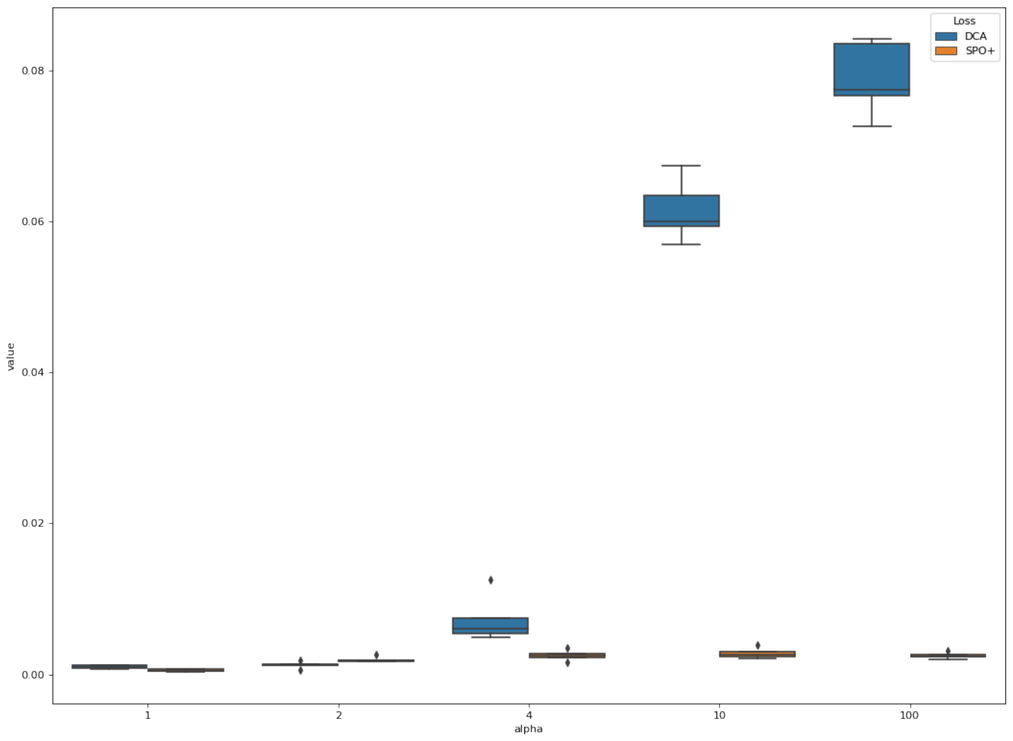


Figure 8: DCA loss increases when α increases (5 simulations)

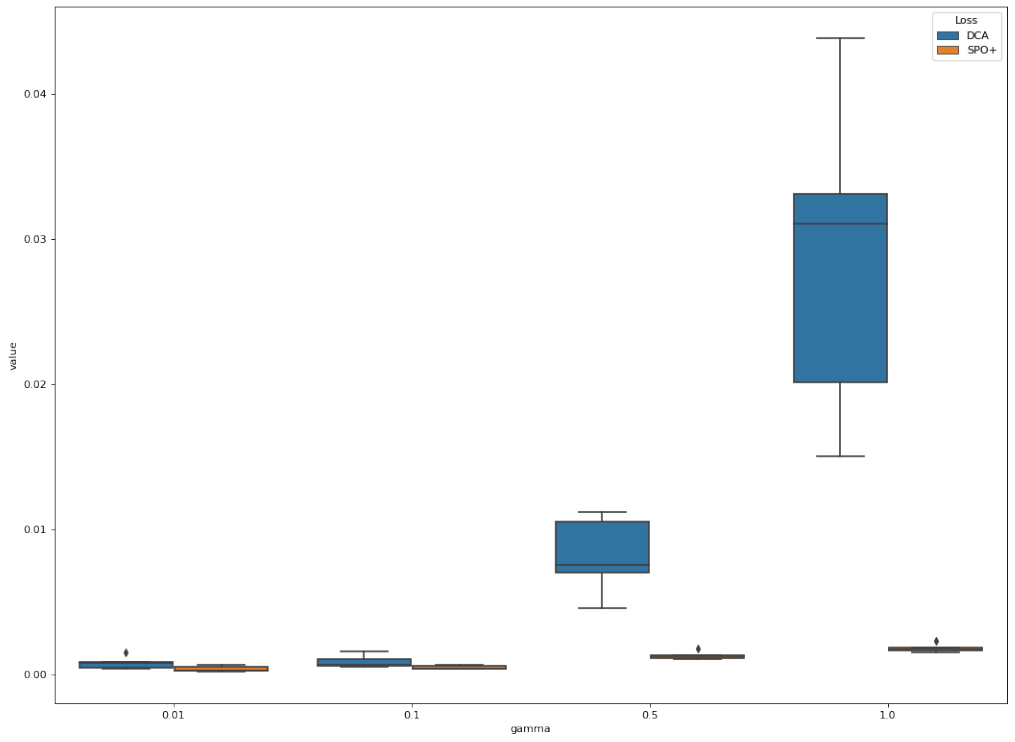


Figure 9: Different DCA loss with different γs (5 simulations)

5 Nonlinear Objective $f(c, w)$

Goal: Come up with general SPO-type arguments for non-linear objective functions

$$\ell_{SPO}(\hat{c}, c) = \max_{w \in W^*(\hat{c})} \{f(c, w)\} - z^*(c)$$

where $z^*(\hat{c}) = \min_{w \in S} f(\hat{c}, w)$

- To use DCA, we want to construct a surrogate loss function that has a DC structure.
- Dual approach from the paper:

$$q(\alpha) = \max_{w \in S} \{f(c, w) - \alpha f(\hat{c}, w)\} + \alpha z^*(\hat{c}) =: \phi(\hat{c}) - g(\hat{c})$$

  We can take

  - $\phi(\hat{c}) = \max_{w \in S}\{f(c, w) - \alpha f(\hat{c}, w)\}$
  - $g(\hat{c}) = \max_{w \in S}\{-\alpha f(\hat{c}, w)\}$

  Assuming $-f(\hat{c}, w)$ is convex w.r.t $\hat{c}$ (i.e., $f(\hat{c}, w)$ is concave w.r.t. $\hat{c}$), we have the desired DC structure

  Computing subgradients:

  - Let $\phi(x) = \max_{w \in S}\{f(c, w) - \alpha f(x, w)\}$ and let $\bar{w}_x = \arg\max_{w \in S}\{f(c, w) - \alpha f(x, w)\}$. Then,

$$\begin{aligned}
\phi(x) - \phi(x') &= \max_{w \in S}\{f(c, w) - \alpha f(x, w)\} - \max_{w \in S}\{f(c, w) - \alpha f(x', w)\} \\
&= \max_{w \in S}\{f(c, w) - \alpha f(x, w)\} - \{f(c, w) - \alpha f(x', \bar{w}_{x'})\} \\
&\geq f(c, w) - \alpha f(x, \bar{w}_{x'}) - (f(c, w) - \alpha f(x', \bar{w}_{x'})) \\
&\geq \alpha (x - x')^T \partial(-f(x', \bar{w}_{x'}))
\end{aligned}$$

Thus, $\partial\phi(x) = \alpha\partial(-f(x, \bar{w}_x))$

– Similarly, let $g(x) = \max_{w\in S}\{-\alpha f(x,w)\}$ and let $\bar{w}_x = \arg\max_{w\in S}\{-\alpha f(x,w)\}$. Then,

$$\begin{aligned} g(x) - g(x') &= \max_{w\in S}\{-\alpha f(x,w)\} - \max_{w\in S}\{-\alpha f(x',w)\} \\ &= \max_{w\in S}\{-\alpha f(x,w)\} - \{-\alpha f(x',\bar{w}_{x'})\} \\ &\geq -\alpha f(x,\bar{w}_{x'}) - (-\alpha f(x',\bar{w}_{x'})) \\ &\geq \alpha(x-x')^T\partial(-f(x',\bar{w}_{x'})) \end{aligned}$$

Thus, $\partial g(x) = \alpha\partial(-f(x, \bar{w}_x))$

The subroutines should solve:

$$\max_{w\in S}\{f(c,w) - \alpha f(x,w)\} \quad , \quad \max_{w\in S}\{-\alpha f(x,w)\}$$

Reduces to non-trivial convex maximization problems (if f is linear, we have simple LPs as in the original SPO example)

- Different surrogate loss function of

$$\ell(c,\hat{c}) = f(c, w^*(\hat{c})) - f(c, w^*(c))$$

where $w^*(c) := \arg\min_{w\in S} f(c,w)$

Unravel arg min into difference of convex functions (or some convex surrogate)

- Bi-level optimization:

A bi-level optimization problem consists of an upper problem and a lower problem. The former defines an objective over two sets of variables, say x and y. The latter binds y as a function of x, typically by solving a minimization problem. Formally, we can write the problem as

$$\begin{aligned} &\min_x f^U(x,y) \\ &\text{s.t. } y \in \arg\min_{y'} f^L(x,y') \end{aligned}$$

Minimizing SPO loss (or any regret loss) can be viewed as a simple instance of bi-level optimization?

$$\begin{aligned} &\min_{\hat{c}} f(\hat{c},w) \\ &\text{s.t. } w \in \arg\min_{w\in S} f(\hat{c},w) \end{aligned}$$

6 Dual Approach with First-Order Condition

Assume f is differentiable w.r.t w:

$$\begin{aligned} w \in \arg\min_{w\in S} f(\hat{c},w) &\implies \nabla_w f(\hat{c},w)^T(v-w) \geq 0 \ \ \forall v \in S \\ &\implies h(w) := \min_{v\in S}\{\nabla_w f(\hat{c},w)^T(v-w)\} \geq 0 \end{aligned}$$

Take Lagrangian

$$L(w,\alpha) := f(c,w) + \alpha h(w)$$

and we obtain

$$\begin{aligned} \max_{w\in S} L(w,\alpha) &= \max_{w\in S}\left\{f(c,w) + \alpha\left(\min_{v\in S}\{\nabla_w f(\hat{c},w)^T v\} - \nabla_w f(\hat{c},w)^T w\right)\right\} \\ &\leq \max_{w\in S}\left\{f(c,w) - \alpha\nabla_w f(\hat{c},w)^T w\right\} + \max_{w\in S}\left\{\alpha\min_{v\in S}\{\nabla_w f(\hat{c},w)^T v\}\right\} \\ &=: \phi(\hat{c}) - g(\hat{c})?? \end{aligned}$$

Can we reformulate it with DC structure? (Convexify max functions)

7 Variational Inequality Problem

arg min (cf. arg max) constraints can be refomulated with variational inequality problem defined as follows: Find $x \in S$ s.t.

$$VI(F,S) := \langle F(x), y-x\rangle \geq 0,\ \forall y \in S$$

If S is closed and convex, then $x \in S$ in the solution of $VI(F,S)$ iff for any $\gamma > 0$, x is a fixed point of the map

$$\mathrm{Proj}_S(I-\gamma F) : S \mapsto S$$

that is,

$$x = \mathrm{Proj}_S(x-\gamma F(x))$$

- Ingredient 1: Regularized gap function:

  $$G_\gamma(x) = \max_{y\in S}\left\{\langle F(x), x-y\rangle - \frac{\gamma}{2}\|x-y\|^2\right\}$$

  x is the solution of $VI(F,S)$ iff x is global minimizer of function $G_\gamma(x)$ in S and $G_\gamma(x) = 0$

- Ingredient 2: "Fixed Point" Idea and DC Programs

  $$\begin{aligned}\min_x\ & f(x) := u(x) - v(x) \\ \text{s.t. } & x \in \Omega\end{aligned} \tag{1}$$

  and

  $$\mathcal{A}_{cccp}(y) = \arg\min\{u(x) - \nabla v(y)^T x : x \in \Omega\}$$

  Suppose x^* is a generalized fixed point of $\mathcal{A}_{cccp}$ and assume that constraints are qualified at x. Then, x is a stationary point of the program in (1).

8 Idea: Bilevel Optimization

- Reformulation 1. KKT Reduction:

  When the lower level problem adheres to certain convexity and regularity conditions, it is possible to replace the lower level optimization task with its KKT conditions.

  $$\begin{aligned}\min_{\hat{c}} & f(\hat{c}, w) \\ \text{s.t. } & I_S(w) \leq 0 \\ & \lambda I_S(w) = 0 \\ & \lambda \geq 0 \\ & \nabla_w L(\hat{c}, w, \lambda) = 0\end{aligned}$$

  where $L(\hat{c}, w, \lambda) = f(\hat{c}, w) + \lambda I_S(w)$.

  - Problem 1) Solving non-trivial problem
  - Problem 2) Gradient of f and indicator function

- Reformulation 2. Lower-level optimal value function Approach

  Let $z^* : \mathbb{R}^d \to \mathbb{R}$ ("lower level optimal value function mapping") denote

  $$z^*(c) = \min_w\{f(c,w) : w \in S\}$$

Then, the bilevel optimization problem can be expressed as follows:

$$\begin{aligned}&\min_{\hat{c}} f(\hat{c}, w)\\ &\text{s.t. } I_S(w) \le 0\\ &\quad\;\; f(\hat{c}, w) \le z^*(\hat{c})\end{aligned}$$

  - Approximate the z^*-mapping: similar to approximation schemes in ICEO (i) uniform approximations or (ii) high-probability approximations.(some paper presents evolutionary algorithm for this)

- Deep Bilevel Learning: Setting
  - m sample pairs: $(x^{(k)}, y^{(k)})_{k=1,\dots m}$, $x^{(k)} \in \mathcal{X}$, $y^{(k)} \in \mathcal{Y}$
  - $\phi_\theta : \mathcal{X} \to \mathcal{Y}$ (model depends on parameter $\theta \in \mathbb{R}^d$)
  - Loss function $\mathcal{L} : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$
  - Consider evaluating loss $\mathcal{L}$ on b mini-batches $\mathcal{B}_i \subset \{1, \dots, m\}$, $i = 1, \dots, b$ with $\mathcal{B}_i \cap \mathcal{B}_j = \phi$ for $i \neq j$. We redefine the loss as

$$\ell_i(\theta) := \sum_{k \in \mathbb{B}_i} \mathcal{L}(\phi_\theta(x^{(k)}), y^{(k)})$$

  - At every iteration, collect a subset of the mini-batches $\mathcal{U}^t \subset \{1, \dots, b\}$, which we partition into $\mathcal{T}^t$ (training set) and $\mathcal{V}^t$ (validation set)
  - SGD uses only one mini-batch to update the parameters (at t-th iteration): $\theta^{t+1} = \theta^t - \epsilon \nabla \ell_i(\theta^t)$ where $i \in \mathcal{U}^t$. Instead, consider the subset $\mathcal{T}^t \subset \mathcal{U}^t$ of mini-batches and look for the linear combination of the losses that best approximates the validation error. We introduce an additional coefficient ω_i per mini-batch in $\mathcal{T}^t$, which we estimate during training. Our task is then to find parameters θ of our model by using exclusively mini-batches in the training set $\mathcal{T}^t \subset \mathcal{U}^t$, and to identify coefficients (hyper-parameters) ω_i so that the model performs well on the validation set $\mathcal{V}^t \subset \mathcal{U}^t$

$$\begin{aligned}\hat{\theta}, \hat{\omega} = \arg\min_{\theta, \omega} \sum_{j \in V^t} \ell_j(\theta(\omega)) + \frac{\mu}{2} |\omega|^2\\ \text{s.t. } \theta(\omega) = \arg\min_{\bar{\theta}} \sum_{i \in T^t} \omega_i \ell_i(\bar{\theta})\end{aligned}$$

Similar to ICEO?

$$\begin{aligned}&\min_{f \in \mathcal{H}} \;\frac{1}{n} \sum_{i=1}^{n} c(w_i, \xi_i) + \rho \phi(w_i)\\ &\text{s.t. } w_i = w_\rho(f(x_i))\end{aligned}$$

- Algorithm: The Bilevel Approximation Methods

  Consider

$$\begin{aligned}&\min_{x \in X} f(x, y^*(x))\\ &\text{s.t. } y^*(x) = \arg\min_{y \in \mathbb{R}^m} g(x, y)\end{aligned}$$

where f and g are continuously differentiable functions and $X \subseteq \mathbb{R}^n$ is a closed convex set.

Gradient Approximation (Ghadimi and Wang, 2018):

$$\bar{\nabla} f(\bar{x}, \bar{y}) := \nabla_x f(\bar{x}, \bar{y}) - M(\bar{x}, \bar{y}) \nabla_y f(\bar{x}, \bar{y})$$

where

$$M(\bar{x}, \bar{y}) := \nabla^2_{xy} g(\bar{x}, \bar{y}) \left[\nabla^2_{yy} g(\bar{x}, \bar{y})\right]^{-1}$$

Algorithm 8 The Bilevel Approximation (BA) Method

```
1: Input: $x_0 \in X$, $y_0 \in \mathbb{R}^m$, nonnegative sequences $\{\alpha_k\}_{k\geq 0}$, $\{\beta\}_{k\geq 0}$, and integer sequence $\{t_k\}_{k\geq 0}$
2: for $k = 0, 1, \ldots$: do
3:     for $t = 0, 1, \ldots, t_k - 1$ do
4:         Set $y_{k+1} = y_k - \beta_t \nabla_y g(x_k, y_t)$
5:     end for
6:     Set $\bar{y}_k = y_{t_k}$ and
           $x_{k+1} = \arg\min_{u \in X} \left\{ \langle \bar{\nabla} f(x_k, \bar{y_k}), u \rangle + \frac{1}{2\alpha_k} \|u - x_k\|^2 \right\}$
7: end for
```

- The outer loop indexed by k, counts the number of inexact projected gradient performed on function f over the feasible set of outer variable $x \in X$
- The inner one indexed by t shows steps of the gradient method with respect to the inner variable and function y and g, respectively.
- The number of iterations of the inner loop determines the tradeoff between the accuracy of inner minimization problem and the total complexity

Algorithm 9 Two-Timescale Stochastic Approximation (TTSA)

```
1: Input: $x_0 \in X$, $y_0 \in \mathbb{R}^m$, nonnegative sequences $\{\alpha_k\}_{k\geq 0}$, $\{\beta\}_{k\geq 0}$
2: for $k = 0, 1, \ldots K$: do
3:     for $t = 0, 1, \ldots, t_k - 1$ do
           $y^{k+1} = y^k - \beta_k \cdot h_g^k$
           $x^{k+1} = P_X(x^k - \alpha_k \cdot h_f^k)$
4:     end for
```

where h_g^k, h_f^k are stochastic estimates of $\nabla_y g(x^k, yk)$, $\bar{\nabla}_x f(x^k, y^{k+1})$, respectively, and $P_X(\cdot)$ is the Euclidean projection operator onto the convex set X. (Hong et al. 2020)

Other minimax, maximin, minimin formulations and algorithms?

Let (P) be our downstream optimization problem:

$$\begin{aligned} \min \quad & -x \\ \text{s.t.} \quad & -w \leq 0 \\ & w \leq b \end{aligned} \tag{P}$$

Note that we have a nonnegativity constraint in our toy example which is intrinsic to the problem (denote it S_0, the "ground" constraint given) We can think of it as a flow preservation constraint at each node in the max flow problem when we are trying to predict the flow constraint values.

Setting: Observe feature vector $x \in \mathcal{X}$ and the real b is defined by the true mapping $f^*(x) : \mathcal{X} \to \mathbb{R}$.

Suppose further that $f^*(x) = 2^{-x}$ with $x \geq 0$ and we have data $(x_1, b_1) = (0, 1)$ and $(x_2, b_2) = (1, 0.5)$. In predict-then-optimize, for instance, we first predict b using linear regression and obtain $\hat{f}(x) = -0.5x + 1$. If the new observation is $x = 3$, we get $\hat{b} = -0.5$ which makes (P) infeasible.

Obs. Hypothesis class of linear functions does not work if apply ERM (even with Tikhonov-type regularization)

$$\min_{f \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} \ell(f(x_i), b_i) + \lambda \|f\|_{\mathcal{H}}^2$$

Should come up with $\mathcal{H}$ in the way that the ERM problem well-posed AND ensures feasibility

- Idea 1. Using point estimate of upstream prediction

  Given new observation $\breve{x}$, we predict $\breve{b}$. In the previouse example, if the new observation is $\breve{x} = 3$, we get the predicted value $\breve{b} = -0.5$, which is not feasible at the moment. Consider $\breve{b} \pm \epsilon$ in the way that makes (P) feasible. (i.e., take $\epsilon = 0.5$)

  In general, we can consider a ball centered at $\breve{b}$. Let

  $$B_\epsilon(\breve{b}) := \{b : \|b - \breve{b}\| \leq \epsilon\}$$
  $$S_\epsilon(\breve{b}) := \{w : a^\top w \leq b, b \in B_\epsilon(\breve{b})\}$$

  We can take ϵ s.t. $S_0 \cup S_\epsilon(\breve{b}) \neq \emptyset$. Note that if S_0 is compact, there exists ϵ s.t. the intersection is a singleton (i.e., lies on the boundary of S_0)

  JERO type argument to set value of ϵ??

  This is more like predict then robust optimize.

- Idea 2. One-shot way (regret-based loss function approach) Want to capture the discrepancy of estimated cost (or estimated feasible set) and the optimal cost (or optimal feasible set)

  What should be the appropriate regret?

  Types of costs:

  (a) Naïve cost: $\min_{w \in S_0} c^\top w$
  Note: This may be unbounded.

  (b) Good feasible cost (from the feasible set formulation above): $\min_{w \in S_0 \cap S_\epsilon(\breve{b})} c^\top w$

  (c) Closest feasible cost: $\min_{w \in S_0 \cap S_{i^*}} c^\top w$
  Going back to the previous example. We can just choose the closest feasible set from the data. That is, choose b_i s.t. $\|\breve{x} - x_i\|$ is minimized (w.r.t certain metric) where $\breve{x}$ is the new observation.

  If $S(\hat{b})$ is well-defined and using (b),

  $$\begin{aligned} \ell(\hat{b}, b) :&= \left| \min_{w \in S(\hat{b})} c^\top w - \min_{w \in S(b)} c^\top w \right| \\ &= \left| \min_{w \in S_0 \cap S_\epsilon(\hat{b})} c^\top w - \min_{w \in S(b)} c^\top w \right| \end{aligned}$$

  ERM:

  $$\frac{1}{n} \sum_{i=1}^{n} \sum_{i=1}^{n} \ell(f(x_i), b_i) = \frac{1}{n} \sum_{i=1}^{n} \left| \min_{w \in S_0 \cap S_\epsilon(f(x_i))} c^\top w - \min_{w \in S(b_i)} c^\top w \right|$$

  Choosing the right cost with plausible feasible set (first term in RHS) and right hypothesis class $\mathcal{H}$ remains funky...

- Idea 3. Contextual Chanced Constrained Optimization

  Assume $\{(x_i, b_i)\}_{i=1}^n$ is i.i.d. from a joint distribution $p(x, b)$.

$$\begin{aligned} \min_w \quad & c^\top w \\ \text{s.t.} \quad & p_x(w) := P_{B|X}(h(w, b) \leq 0 \mid X = x) \geq 1 - \alpha \\ & w \in S_0 \end{aligned}$$

  where $P_{B|X}$ is the conditional probability of B given X. For each $X = x$, we have a solution $w^*(x)$ to CCCO problem.

  Possible DRO extension? Define a empirical distribution $\hat{\mathbb{P}}_n$ and consider an ambiguity set $\mathcal{P}_r := \{\mathbb{P} : d(\mathbb{P}, \hat{\mathbb{P}}_n) \leq r\}$ along with contextual information $X = x$

- Idea 4. Algorithmic approach?

$$\begin{aligned} c(w, \xi) &\geq \min_w c(w, \xi) \\ \mathbb{E}_\xi[c(w, \xi)] &\geq \mathbb{E}_\xi[\min_w c(w, \xi)] \\ \min_w \mathbb{E}_\xi[c(w, \xi)] &\geq \mathbb{E}_\xi[\min_w c(w, \xi)] \end{aligned}$$