Generalized distances for practical optimization methods

Daniel Cederberg

April 30, 2024

Part one - generalized distances

Let's derive gradient descent

$$\text{minimize} \quad f(x)$$

- $f : \mathbf{R}^n \to \mathbf{R}$ differentiable

Let's derive gradient descent

$$\text{minimize} \quad f(x)$$

- $f : \mathbf{R}^n \to \mathbf{R}$ differentiable
- $x_{k+1} = \operatorname{argmin}_x \Big(\underbrace{f(x_k) + \langle \nabla f(x_k), x - x_k \rangle}_{\text{model}}$

Let's derive gradient descent

$$\text{minimize} \quad f(x)$$

- $f : \mathbf{R}^n \to \mathbf{R}$ differentiable
- $x_{k+1} = \operatorname{argmin}_x \left(\underbrace{f(x_k) + \langle \nabla f(x_k), x - x_k \rangle}_{\text{model}} + \underbrace{\frac{1}{2\alpha} \|x - x_k\|_2^2}_{\text{proximity term}} \right)$

Let's derive gradient descent

$$\text{minimize} \quad f(x)$$

- $f : \mathbf{R}^n \to \mathbf{R}$ differentiable
- $$x_{k+1} = \operatorname{argmin}_x \left(\underbrace{f(x_k) + \langle \nabla f(x_k), x - x_k \rangle}_{\text{model}} + \underbrace{\frac{1}{2\alpha} \|x - x_k\|_2^2}_{\text{proximity term}} \right)$$
$$= x_k - \alpha \nabla f(x_k)$$

Let's derive the proximal gradient method

$$\text{minimize} \quad f(x) + g(x)$$

- f differentiable, g possibly non-differentiable

Let's derive the proximal gradient method

$$\text{minimize} \quad f(x) + g(x)$$

- f differentiable, g possibly non-differentiable
- $x_{k+1} = \operatorname{argmin}_x \Big(\underbrace{f(x_k) + \langle \nabla f(x_k), x - x_k \rangle + g(x)}_{\text{model}}$

Let's derive the proximal gradient method

$$\text{minimize} \quad f(x) + g(x)$$

- f differentiable, g possibly non-differentiable
- $x_{k+1} = \operatorname{argmin}_x \Big(\underbrace{f(x_k) + \langle \nabla f(x_k), x - x_k \rangle + g(x)}_{\text{model}} + \underbrace{\frac{1}{2\alpha} \|x - x_k\|_2^2}_{\text{proximity term}} \Big)$

Let's derive the proximal gradient method

$$\text{minimize} \quad f(x) + g(x)$$

- f differentiable, g possibly non-differentiable
- $$x_{k+1} = \operatorname{argmin}_x \Bigg(\underbrace{f(x_k) + \langle \nabla f(x_k), x - x_k \rangle + g(x)}_{\text{model}} + \underbrace{\frac{1}{2\alpha} \|x - x_k\|_2^2}_{\text{proximity term}} \Bigg)$$
$$= \mathbf{prox}_{\alpha g}(x_k - \alpha \nabla f(x_k))$$

Model based optimization

- Gradient descent:

$$x_{k+1} = \operatorname*{argmin}_x \left(\underbrace{f(x_k) + \langle \nabla f(x_k), x - x_k \rangle}_{\text{model}} + \underbrace{\frac{1}{2\alpha} \|x - x_k\|_2^2}_{\text{proximity term}} \right)$$

- Proximal gradient method:

$$x_{k+1} = \operatorname*{argmin}_x \left(\underbrace{f(x_k) + \langle \nabla f(x_k), x - x_k \rangle + g(x)}_{\text{model}} + \underbrace{\frac{1}{2\alpha} \|x - x_k\|_2^2}_{\text{proximity term}} \right)$$

Model based optimization

- Gradient descent:

$$x_{k+1} = \operatorname*{argmin}_x \left(\underbrace{f(x_k) + \langle \nabla f(x_k), x - x_k \rangle}_{\text{model}} + \underbrace{\frac{1}{2\alpha} \|x - x_k\|_2^2}_{\text{proximity term}} \right)$$

- Proximal gradient method:

$$x_{k+1} = \operatorname*{argmin}_x \left(\underbrace{f(x_k) + \langle \nabla f(x_k), x - x_k \rangle + g(x)}_{\text{model}} + \underbrace{\frac{1}{2\alpha} \|x - x_k\|_2^2}_{\text{proximity term}} \right)$$

- why use the Euclidean norm to define proximity?

Generalized distance functions

- the Euclidean norm can be replaced by a *generalized distance function*
- most popular class of generalized distance functions is the class of *Bregman divergences*

Generalized distance functions

- the Euclidean norm can be replaced by a *generalized distance function*
- most popular class of generalized distance functions is the class of *Bregman divergences*

Why would we use a generalized distance?

Generalized distance functions

- the Euclidean norm can be replaced by a *generalized distance function*
- most popular class of generalized distance functions is the class of *Bregman divergences*

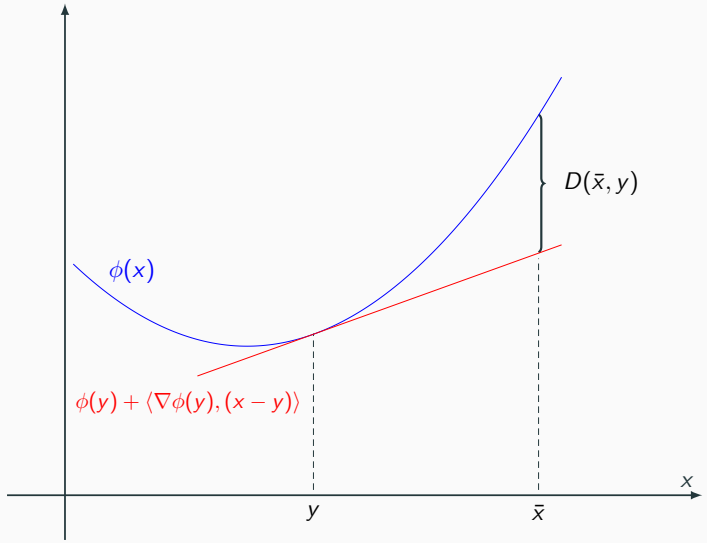
Why would we use a generalized distance?

1. Stronger convergence results for very specific problems.
2. Cheaper iterations (a proximal operator based on a Bregman divergence can be cheaper to evaluate than the standard prox).

Bregman divergences

$$D(x, y) = \phi(x) - \phi(y) - \langle \nabla\phi(y), x - y \rangle$$

- a Bregman divergence is defined through a *kernel* ϕ
- the kernel is convex and differentiable (other mathematical properties are required)



The most well-known textbook example

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & x \in \Delta \end{array}$$

- feasible set $\Delta = \{x \in \mathbf{R}^n \mid 1^T x = 1,\ x \geq 0\}$
- access to a subgradient oracle giving $g_k \in \partial f(x_k)$

The most well-known textbook example

- Choosing the *entropy* kernel $\phi(x) = \sum_{i=1}^{n} x_i \log x_i$ gives us *mirror descent*:

$$x_{k+1} = \operatorname*{argmin}_{x \in \Delta} \left(f(x_k) + \langle g_k, x - x_k \rangle + (1/\alpha) D_\phi(x, x_k) \right)$$

The most well-known textbook example

- Choosing the *entropy* kernel $\phi(x) = \sum_{i=1}^{n} x_i \log x_i$ gives us *mirror descent*:

$$x_{k+1} = \operatorname*{argmin}_{x \in \Delta} \left(f(x_k) + \langle g_k, x - x_k \rangle + (1/\alpha) D_\phi(x, x_k) \right)$$

- Projected subgradient method:

$$\begin{aligned} x_{k+1} &= \operatorname*{argmin}_{x \in \Delta} \left(f(x_k) + \langle g_k, x - x_k \rangle + \frac{1}{2} \|x - x_k\|_2^2 \right) \\ &= \mathbf{Proj}_\Delta(x_k - \alpha g_k). \end{aligned}$$

The most well-known textbook example

- Choosing the *entropy* kernel $\phi(x) = \sum_{i=1}^{n} x_i \log x_i$ gives us *mirror descent*:

$$x_{k+1} = \operatorname*{argmin}_{x \in \Delta} \left(f(x_k) + \langle g_k, x - x_k \rangle + (1/\alpha) D_\phi(x, x_k) \right)$$

- Projected subgradient method:

$$\begin{aligned} x_{k+1} &= \operatorname*{argmin}_{x \in \Delta} \left(f(x_k) + \langle g_k, x - x_k \rangle + \frac{1}{2} \|x - x_k\|_2^2 \right) \\ &= \mathbf{Proj}_\Delta(x_k - \alpha g_k). \end{aligned}$$

- MD update can be worked out analytically, Euclidean projection can be done in $\mathcal{O}(n \log n)$ (requires a sort).

Why do theoretical optimizers love this example?

Assume known bounds on the size of subgradients: for all $x \in \Delta$,

$$\|g\|_\infty \leq G_\infty, \qquad \|g\|_2 \leq G_2, \qquad \forall g \in \partial f(x).$$

- MD with "optimal" constant step size:

$$f_{\text{best}}^k - f^\star \leq G_\infty \sqrt{\frac{2 \log n}{k}}$$

- PSM with "optimal" constant step size:

$$f_{\text{best}}^k - f^\star \leq G_2 \sqrt{\frac{1}{k}}$$

Why do theoretical optimizers love this example?

- $\underbrace{f_{\text{best}}^k - f^\star \leq G_\infty \sqrt{\frac{2\log n}{k}}}_{\text{mirror descent}}, \qquad \underbrace{f_{\text{best}}^k - f^\star \leq G_2 \sqrt{\frac{1}{k}}}_{\text{projected subgradient method}}$

Why do theoretical optimizers love this example?

- $\underbrace{f_{\text{best}}^k - f^\star \leq G_\infty \sqrt{\frac{2\log n}{k}}}_{\text{mirror descent}}, \qquad \underbrace{f_{\text{best}}^k - f^\star \leq G_2 \sqrt{\frac{1}{k}}}_{\text{projected subgradient method}}$

- typical textbook presentation: insert bound $G_2 \leq \sqrt{n} G_\infty$

Why do theoretical optimizers love this example?

- $\underbrace{f_{\text{best}}^k - f^\star \leq G_\infty \sqrt{\frac{2\log n}{k}}}_{\text{mirror descent}}, \qquad \underbrace{f_{\text{best}}^k - f^\star \leq G_2 \sqrt{\frac{1}{k}}}_{\text{projected subgradient method}}$

- typical textbook presentation: insert bound $G_2 \leq \sqrt{n} G_\infty$

- $f_{\text{best}}^k - f^\star \leq G_\infty \sqrt{\frac{2\log n}{k}}, \qquad f_{\text{best}}^k - f^\star \leq G_\infty \sqrt{\frac{n}{k}}$

Why do theoretical optimizers love this example?

- $$\underbrace{f_{\text{best}}^k - f^\star \leq G_\infty \sqrt{\frac{2\log n}{k}}}_{\text{mirror descent}}, \qquad \underbrace{f_{\text{best}}^k - f^\star \leq G_2 \sqrt{\frac{1}{k}}}_{\text{projected subgradient method}}$$

- typical textbook presentation: insert bound $G_2 \leq \sqrt{n} G_\infty$

- $$f_{\text{best}}^k - f^\star \leq G_\infty \sqrt{\frac{2\log n}{k}}, \qquad f_{\text{best}}^k - f^\star \leq G_\infty \sqrt{\frac{n}{k}}$$

- MD converges at a rate $\mathcal{O}(\sqrt{\log n/k})$ and PSM at rate $\mathcal{O}(\sqrt{n/k})$

Why do theoretical optimizers love this example?

- $$\underbrace{f_{\text{best}}^k - f^\star \leq G_\infty \sqrt{\frac{2\log n}{k}}}_{\text{mirror descent}}, \qquad \underbrace{f_{\text{best}}^k - f^\star \leq G_2 \sqrt{\frac{1}{k}}}_{\text{projected subgradient method}}$$

- typical textbook presentation: insert bound $G_2 \leq \sqrt{n} G_\infty$

- $$f_{\text{best}}^k - f^\star \leq G_\infty \sqrt{\frac{2\log n}{k}}, \qquad f_{\text{best}}^k - f^\star \leq G_\infty \sqrt{\frac{n}{k}}$$

- MD converges at a rate $\mathcal{O}(\sqrt{\log n/k})$ and PSM at rate $\mathcal{O}(\sqrt{n/k})$

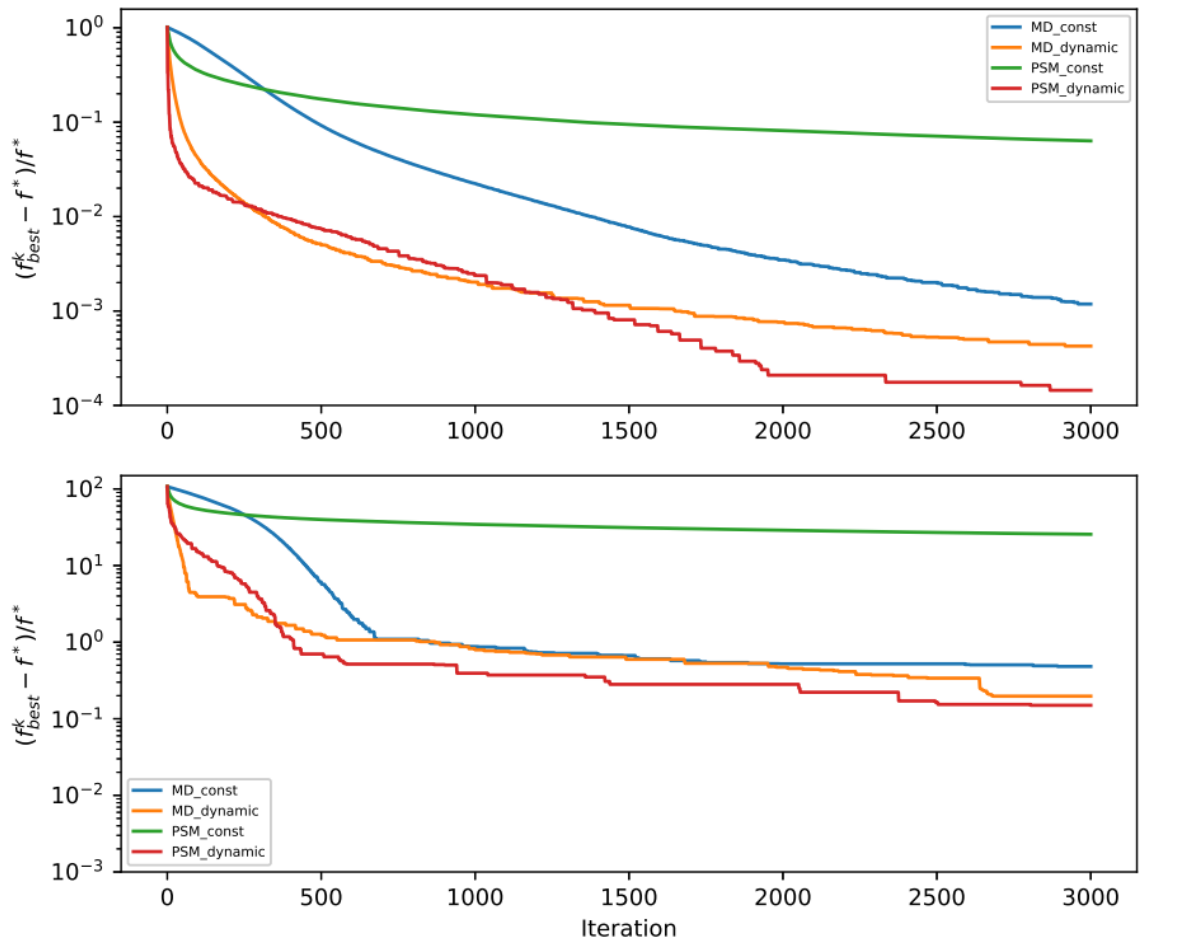
why they love MD!

Mirror descent in practice (example one)

$$\begin{array}{ll} \text{minimize} & f(x) = \|Ax - b\|_1 \\ \text{subject to} & x \in \Delta \end{array}$$

- called robust regression over the simplex

Mirror descent in practice (example one)



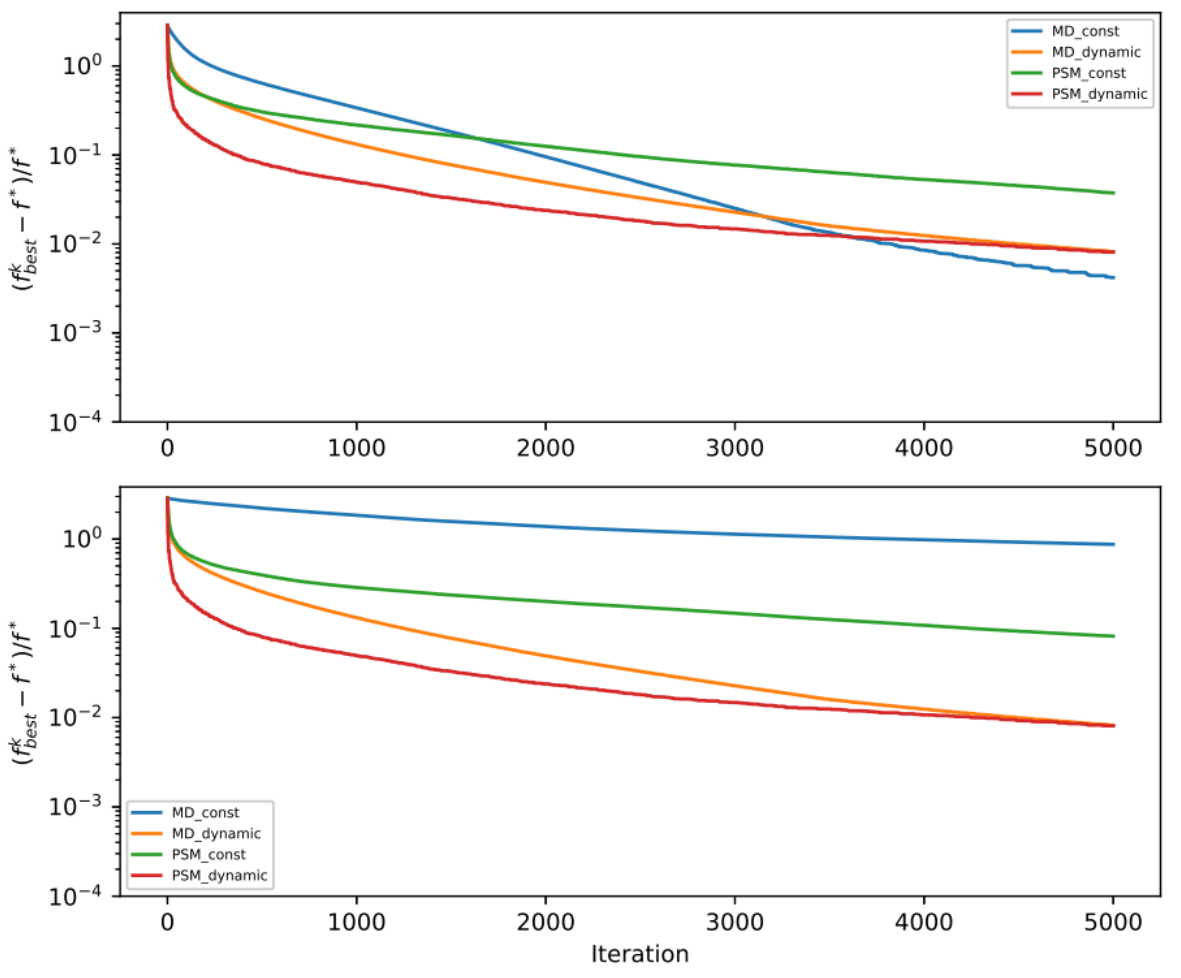
- data generated in two different ways

Mirror descent in practice (example two)

$$\begin{array}{ll} \text{minimize} & c^T x \\ \text{subject to} & Ax \leq b,\ x \geq 0 \end{array}$$

- entries of c, A and b are nonnegative
- called positive linear programming
- can be formulated as a minimization problem over the simplex

Mirror descent in practice (example two)



- data generated in two different ways

Is mirror descent (over the simplex) a scam?

- maybe not...

Is mirror descent (over the simplex) a scam?

- maybe not...
- but it is not as superior as theoretical optimizers claim

Is mirror descent (over the simplex) a scam?

- maybe not...
- but it is not as superior as theoretical optimizers claim
- adaptive step sizes is what matters!

Is mirror descent (over the simplex) a scam?

- maybe not...
- but it is not as superior as theoretical optimizers claim
- adaptive step sizes is what matters!
- disclaimer: I have only talked about *numerically* solving problems

Some important references on generalized distances

1. Three landmark papers: Bregman (1967), Nemirovski & Yudin (1983), Beck & Teboulle (2003)
2. A textbook dedicated to the topic: Censor & Zenios (1997)
3. More recent framework of relative smoothness: Bauschke, Bolte & Teboulle (2016) and Lu, Freund & Nesterov (2016)

Outline

1. Part one: generalized distances. ✓
2. Part two: trigonometric matrix polynomials and semidefinite programming.
3. Part three: new first-order methods

Part two - trigonometric matrix polynomials and semidefinite programming

Trigonometric matrix polynomials

- A *trigonometric matrix polynomial* of degree p is a function $F_{\mathcal{X}} : \mathbf{C} \to \mathbf{C}^{m\times m}$ of the form

$$F_{\mathcal{X}}(z) = X_0 + \sum_{k=1}^{p}(X_k z^k + X_k^T z^{-k})$$

- The matrices $X_0 \in \mathbf{S}^m$ and $X_k \in \mathbf{R}^{m\times m}$, $1 \leq k \leq p$ are called the *coefficients*
- $\mathcal{X} = (X_0, X_1, \ldots, X_p)$

Trigonometric matrix polynomials, crash course

- a zero-mean random *vector* $x \in \mathbf{R}^n$ has a *covariance matrix* $\Sigma = E[xx^T]$.
- a zero-mean random *time series* $\{x(t) \in \mathbf{R}^m,\ t \in \mathbf{Z}\}$ has a *spectral density matrix* $f : \mathbf{R} \to \mathbf{C}^{m\times m}$ given by

$$f(\omega) = \sum_{k=-\infty}^{\infty} R_k e^{-jk\omega},$$

where $j = \sqrt{-1}$. (Here $R_k = \mathrm{E}[x(t+k)x(t)^T]$, $k \geq 0$ are the *autocorrelation coefficients*.)

Trigonometric matrix polynomials, crash course

Roughly speaking, the power spectrum of a time series is analogous to the covariance matrix of a random vector:

- Nonnegativity:
  a) covariance matrix $\Sigma \succeq 0$
  b) spectral density matrix $f(\omega) \succeq 0$ for all $\omega \in \mathbf{R}$
- Graphical models:
  a) sparsity pattern of Σ^{-1} related to conditional independence
  b) sparsity pattern of $f(\omega)^{-1}$ related to conditional independence

SDP parameterization & complexity

$$X_0 + \sum_{k=1}^{p} (X_k z^k + X_k^T z^{-k}) \succeq 0, \qquad \forall z \in \mathbf{C} : |z| = 1.$$

The matrices $X_k,\ k = 0, \ldots, p$ have dimension $m \times m$.

SDP parameterization & complexity

$$X_0 + \sum_{k=1}^{p} (X_k z^k + X_k^T z^{-k}) \succeq 0, \qquad \forall z \in \mathbf{C} : |z| = 1.$$

The matrices $X_k,\ k = 0, \ldots, p$ have dimension $m \times m$.

- can be formulated as semidefinite programs

SDP parameterization & complexity

$$X_0 + \sum_{k=1}^{p} (X_k z^k + X_k^T z^{-k}) \succeq 0, \qquad \forall z \in \mathbf{C} : |z| = 1.$$

The matrices $X_k,\ k = 0, \ldots, p$ have dimension $m \times m$.

- can be formulated as semidefinite programs
- complexity of general-purpose interior-point solver is (at least) $\mathcal{O}(m^6 p^4)$ per iteration

SDP parameterization & complexity

$$X_0 + \sum_{k=1}^{p} (X_k z^k + X_k^T z^{-k}) \succeq 0, \qquad \forall z \in \mathbf{C} : |z| = 1.$$

The matrices $X_k,\ k = 0, \ldots, p$ have dimension $m \times m$.

- can be formulated as semidefinite programs
- complexity of general-purpose interior-point solver is (at least) $\mathcal{O}(m^6 p^4)$ per iteration
- **Example:**

	$m = 10, p = 4$	
Solve time		

SDP parameterization & complexity

$$X_0 + \sum_{k=1}^{p} (X_k z^k + X_k^T z^{-k}) \succeq 0, \qquad \forall z \in \mathbf{C} : |z| = 1.$$

The matrices $X_k,\ k = 0, \ldots, p$ have dimension $m \times m$.

- can be formulated as semidefinite programs
- complexity of general-purpose interior-point solver is (at least) $\mathcal{O}(m^6 p^4)$ per iteration
- **Example:**

	$m = 10, p = 4$	
Solve time	1 sec	

SDP parameterization & complexity

$$X_0 + \sum_{k=1}^{p} (X_k z^k + X_k^T z^{-k}) \succeq 0, \qquad \forall z \in \mathbf{C} : |z| = 1.$$

The matrices $X_k,\ k = 0, \ldots, p$ have dimension $m \times m$.

- can be formulated as semidefinite programs
- complexity of general-purpose interior-point solver is (at least) $\mathcal{O}(m^6 p^4)$ per iteration
- **Example:**

	$m = 10, p = 4$	$m = 40, p = 16$
Solve time	1 sec	

SDP parameterization & complexity

$$X_0 + \sum_{k=1}^{p} (X_k z^k + X_k^T z^{-k}) \succeq 0, \qquad \forall z \in \mathbf{C} : |z| = 1.$$

The matrices $X_k,\ k = 0, \ldots, p$ have dimension $m \times m$.

- can be formulated as semidefinite programs
- complexity of general-purpose interior-point solver is (at least) $\mathcal{O}(m^6 p^4)$ per iteration
- **Example:**

	$m = 10, p = 4$	$m = 40, p = 16$
Solve time	1 sec	12 days

Outline

1. Part one - generalized distances ✓
2. Part two - trigonometric matrix polynomials and semidefinite programming ✓
3. Part three - Bregman first-order methods

Part three - Bregman first-order methods

Cone of nonnegative trigonometric matrix polynomials

- constraint $F_{\mathcal{X}}(z) \succeq 0, \forall z \in \mathbf{C} : |z| = 1$ where

$$F_{\mathcal{X}}(z) = X_0 + \sum_{k=1}^{p} (X_k z^k + X_k^T z^{-k})$$

Cone of nonnegative trigonometric matrix polynomials

- constraint $F_{\mathcal{X}}(z) \succeq 0, \forall z \in \mathbf{C} : |z| = 1$ where

$$F_{\mathcal{X}}(z) = X_0 + \sum_{k=1}^{p} (X_k z^k + X_k^T z^{-k})$$

- let K be the cone of nonnegative matrix polynomials:

$$K = \{\mathcal{X} \mid F_{\mathcal{X}}(z) \succeq 0 \text{ for } |z| = 1\}.$$

Cone of nonnegative trigonometric matrix polynomials

- constraint $F_{\mathcal{X}}(z) \succeq 0, \forall z \in \mathbf{C} : |z| = 1$ where

$$F_{\mathcal{X}}(z) = X_0 + \sum_{k=1}^{p} (X_k z^k + X_k^T z^{-k})$$

- let K be the cone of nonnegative matrix polynomials:

$$K = \{\mathcal{X} \mid F_{\mathcal{X}}(z) \succeq 0 \text{ for } |z| = 1\}.$$

- problems of the form

$$\begin{array}{ll} \text{minimize} & f(\mathcal{X}) \\ \text{subject to} & \mathcal{X} \in K \end{array}$$

A Bregman proximal operator

- For some $a > 0$, define g as

$$g(\mathcal{X}) = \begin{cases} 0 & \text{if } \mathcal{X} \in K,\ \mathbf{Tr}(X_0) = a \\ \infty & \text{otherwise.} \end{cases}$$

A Bregman proximal operator

- For some $a > 0$, define g as

$$g(\mathcal{X}) = \begin{cases} 0 & \text{if } \mathcal{X} \in K,\ \mathbf{Tr}(X_0) = a \\ \infty & \text{otherwise.} \end{cases}$$

- Bregman proximal operator

$$\mathbf{prox}_g^{\phi}(\mathcal{V}, \mathcal{A}) = \operatorname*{argmin}_{\mathcal{X}} \left(g(\mathcal{X}) + \langle \mathcal{A}, \mathcal{X} \rangle + D_{\phi}(\mathcal{X}, \mathcal{V}) \right).$$

A Bregman proximal operator

- For some $a > 0$, define g as

$$g(\mathcal{X}) = \begin{cases} 0 & \text{if } \mathcal{X} \in K,\ \mathbf{Tr}(X_0) = a \\ \infty & \text{otherwise.} \end{cases}$$

- Bregman proximal operator

$$\mathbf{prox}_g^{\phi}(\mathcal{V}, \mathcal{A}) = \operatorname*{argmin}_{\mathcal{X}} \left(g(\mathcal{X}) + \langle \mathcal{A}, \mathcal{X} \rangle + D_{\phi}(\mathcal{X}, \mathcal{V}) \right).$$

- with a clever choice of the kernel the Bregman proximal operator can be evaluated efficiently

A Bregman proximal operator

- **Kernel** (entropy function):

$$\phi(\mathcal{X}) = -\frac{1}{2\pi}\int_0^{2\pi} \textbf{log det } F_{\mathcal{X}}(e^{j\omega})d\omega$$

A Bregman proximal operator

- **Kernel** (entropy function):

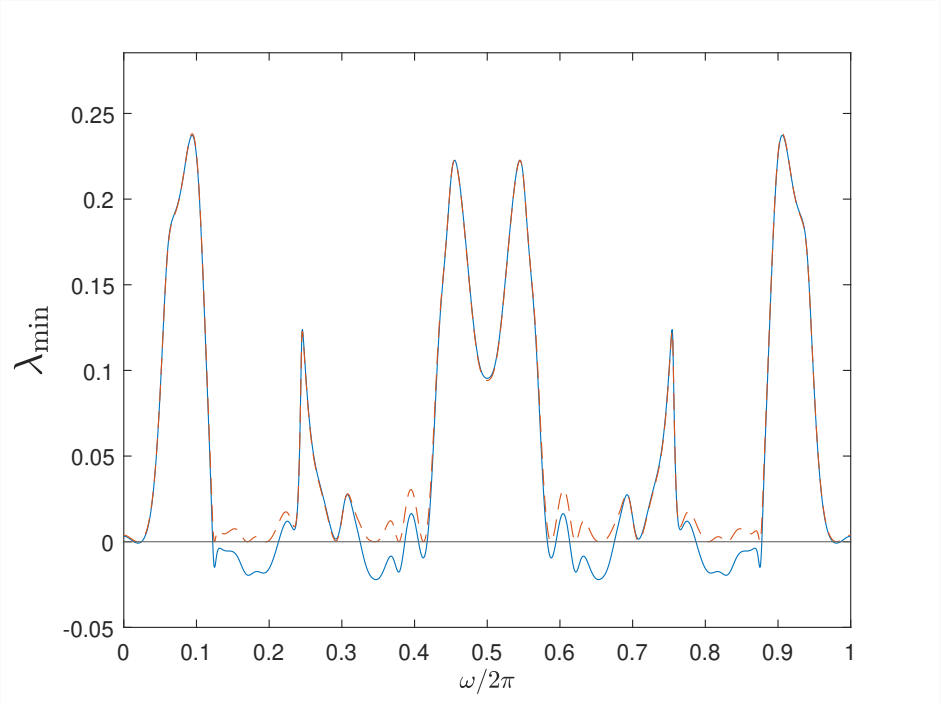
$$\phi(\mathcal{X}) = -\frac{1}{2\pi}\int_0^{2\pi} \textbf{log det } F_{\mathcal{X}}(e^{j\omega})d\omega$$

- we design an efficient method for evaluating $\textbf{prox}_g^{\phi}(\mathcal{V}, \mathcal{A})$:
  1. Main cost is solving positive definite block Toeplitz linear systems.
  2. Complexity is $\mathcal{O}(m^3p^2)$ per iteration.

Application one: nonnegative spectral estimation

$$\begin{array}{ll} \text{minimize} & \|\mathcal{X} - \hat{\mathcal{R}}^T\|^2 \\ \text{subject to} & \mathcal{X} \in K,\ \mathrm{Tr}(X_0) = \mathrm{Tr}(\hat{R}_0) \end{array}$$

- problem data $\hat{\mathcal{R}} = (\hat{R}_0, \ldots, \hat{R}_p)$ corresponds to a *power spectrum* that fails to be nonnegative



Algorithm

- write as

$$\text{minimize } f(\mathcal{X}) + g(\mathcal{X})$$

where $f(\mathcal{X}) = \|\mathcal{X} - \hat{\mathcal{R}}^T\|^2$, and g indicator of feasible set.

Algorithm

- write as

$$\text{minimize } f(\mathcal{X}) + g(\mathcal{X})$$

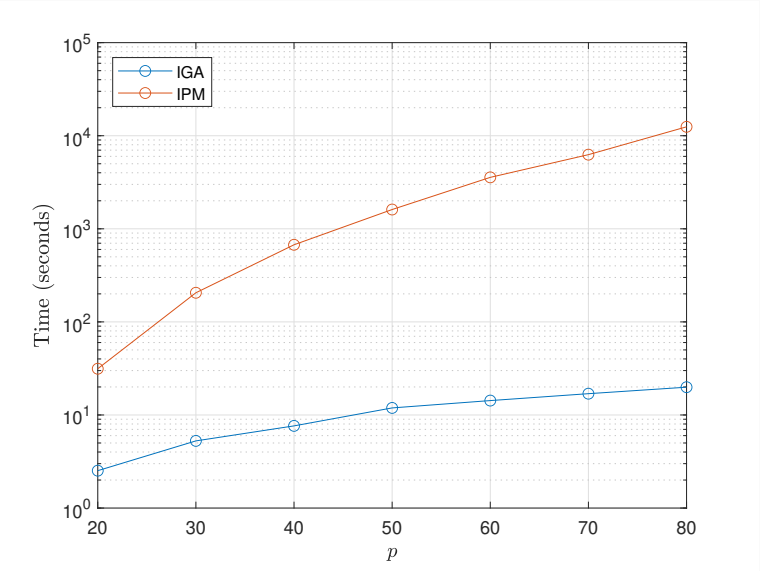
  where $f(\mathcal{X}) = \|\mathcal{X} - \hat{\mathcal{R}}^T\|^2$, and g indicator of feasible set.

- Improved gradient algorithm:

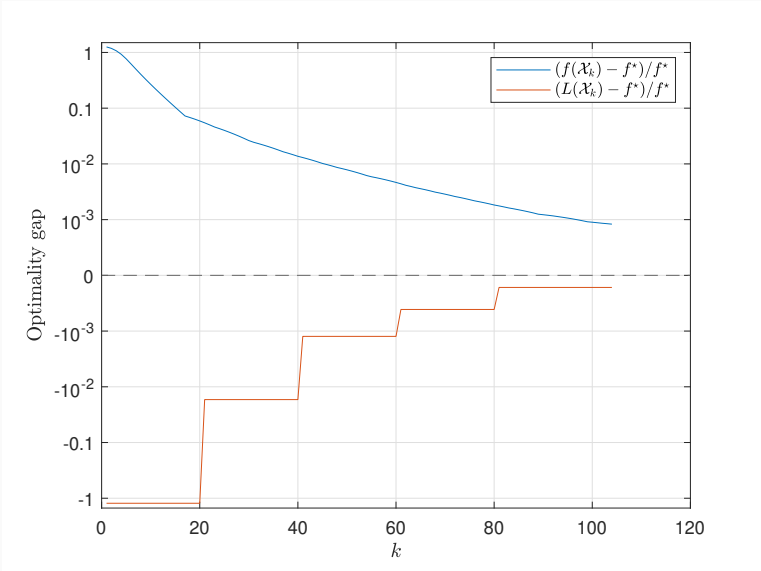
$$\begin{aligned}
\mathcal{Y}_{k+1} &= \mathcal{X}_k + \theta_k(\mathcal{V}_k - \mathcal{X}_k) \\
\mathcal{V}_{k+1} &= \mathbf{prox}^{\phi}_{\tau_k g}(\mathcal{V}_k, \tau_k \nabla f(\mathcal{Y}_{k+1})) \\
\mathcal{X}_{k+1} &= \mathcal{X}_k + \theta_k(\mathcal{V}_{k+1} - \mathcal{X}_k)
\end{aligned}$$

  a) an accelerated proximal gradient method that allows for a Bregman proximal operator
  b) main cost in each iteration is to evaluate the Bregman proximal operator

Numerical results



(a)



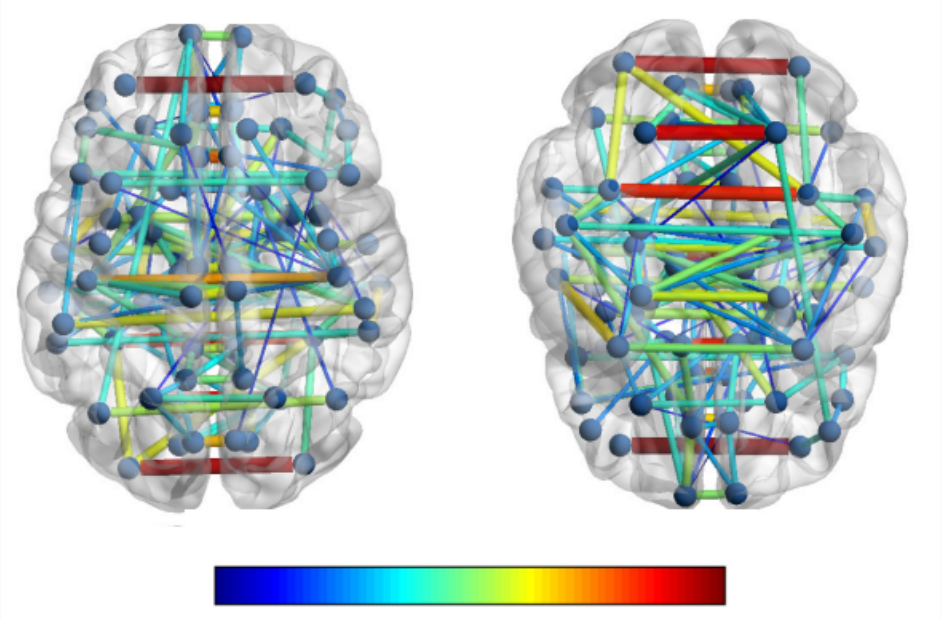
(b)

Application two: graphical models

- sparsity pattern of $f(\omega)^{-1}$ determines a *conditional independence graph*
- for Gaussian time series: $\{x_i(t), t \in \mathrm{Z}\}$ and $\{x_j(t), t \in \mathrm{Z}\}$ are conditionally independent, given $\{x_k(t), t \in \mathrm{Z}\}$ for $k \neq i, j, \iff (f(\omega)^{-1})_{ij} = 0.$
- encouraging sparsity when estimating $f(\omega)^{-1}$ makes sense

Example: functional connectivity analysis

A conditional independence graph visualized:



Problem formulation

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\|\mathcal{X} - \hat{\mathcal{Y}}^T\|^2 + \gamma r(\mathcal{X}) \\ \text{subject to} & \mathcal{X} \in K,\ \mathbf{Tr}(X_0) = \mathbf{Tr}(\hat{Y}_0) \end{array}$$

- problem data $\hat{\mathcal{Y}} = (\hat{Y}_0, \ldots, \hat{Y}_p)$ corresponds to a *measured* inverse power spectrum
- the variable $\mathcal{X}$ represents a *sparsified* inverse power spectrum

Problem formulation

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\|\mathcal{X} - \hat{\mathcal{Y}}^T\|^2 + \gamma r(\mathcal{X}) \\ \text{subject to} & \mathcal{X} \in K,\ \mathbf{Tr}(X_0) = \mathbf{Tr}(\hat{Y}_0) \end{array}$$

- problem data $\hat{\mathcal{Y}} = (\hat{Y}_0, \ldots, \hat{Y}_p)$ corresponds to a *measured* inverse power spectrum
- the variable $\mathcal{X}$ represents a *sparsified* inverse power spectrum
- the regularizer r encourages a common sparsity pattern

Problem formulation

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\|\mathcal{X} - \hat{\mathcal{Y}}^T\|^2 + \gamma r(\mathcal{X}) \\ \text{subject to} & \mathcal{X} \in K,\ \mathbf{Tr}(X_0) = \mathbf{Tr}(\hat{Y}_0) \end{array}$$

- problem data $\hat{\mathcal{Y}} = (\hat{Y}_0, \ldots, \hat{Y}_p)$ corresponds to a *measured* inverse power spectrum
- the variable $\mathcal{X}$ represents a *sparsified* inverse power spectrum
- the regularizer r encourages a common sparsity pattern
- write as

$$\text{minimize } f(\mathcal{X}) + g(\mathcal{X})$$

where $g(\mathcal{X}) = \frac{1}{2}\|\mathcal{X} - \hat{\mathcal{R}}^T\|^2 + \gamma r(\mathcal{X})$ and

$$f(\mathcal{X}) = \begin{cases} 0 & \text{if } \mathcal{X} \in K,\ \mathbf{Tr}(X_0) = \mathbf{Tr}(R_0) \\ \infty & \text{otherwise,} \end{cases}.$$

Algorithm

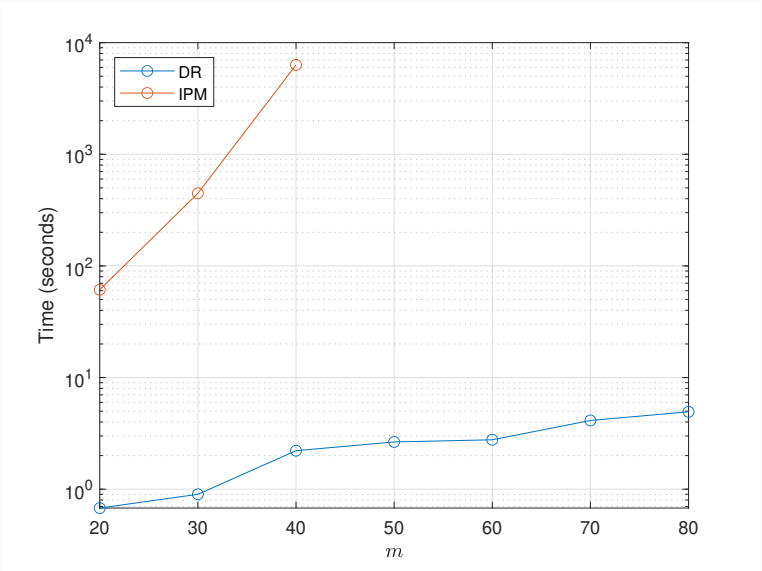
$$\text{minimize } f(\mathcal{X}) + g(\mathcal{X})$$

Bregman Douglas–Rachford:

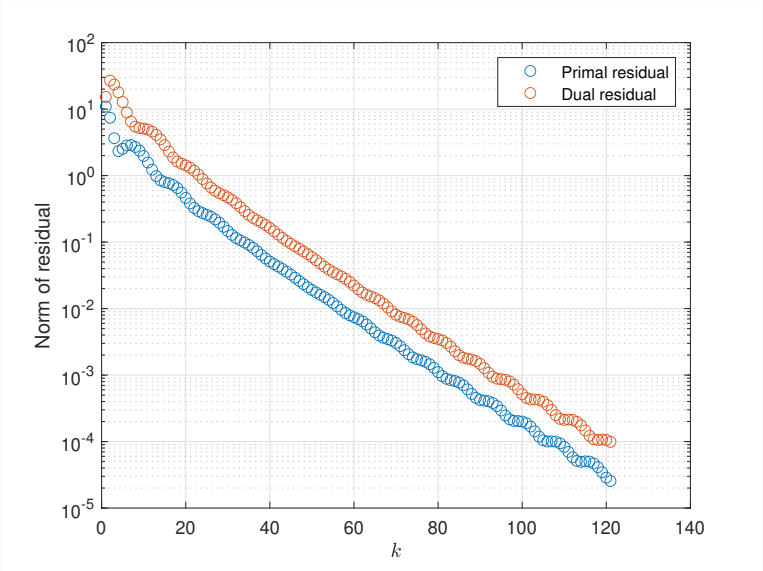
$$\mathcal{Y}_{k+1} = \mathbf{prox}_{\sigma g^*}(\mathcal{Y}_k + \sigma(2\mathcal{X}_k - \mathcal{X}_{k-1}))$$
$$\mathcal{X}_{k+1} = \mathbf{prox}^{\phi}_{\tau f}(\mathcal{X}_k, \tau\mathcal{Y}_{k+1})$$

Super simple algorithm!

Numerical results



(c)



(d)

Outline

1. Part one - generalized distances ✓
2. Part two - trigonometric matrix polynomials and semidefinite programming ✓
3. Part three - Bregman first-order methods ✓

Summary

- most work on generalized distances is theoretical
- useful in practice only for very specific problems
- when they apply they can be really fast

Thanks! Questions?