Chapter 5

Proximal methods

Contents (class version)

5.0 Introduction **5.3**

5.1 Proximal basics **5.4**

Proximal operator 5.4

Complex cases 5.14

Proximal point algorithm 5.15

5.2 Proximal gradient method (PGM) **5.18**

Convergence rate of PGM 5.23

PGM with line search 5.27

Iterative hard thresholding 5.28

5.3 Accelerated proximal methods **5.30**

Fast proximal gradient method (FPGM) 5.30

Proximal optimized gradient method (POGM) 5.33

Inexact computation of proximal operators 5.36

Other proximal methods 5.37

5.4 Examples **5.38**

Machine learning: Binary classifier with 1-norm regularizer 5.38

Example: MRI compressed sensing 5.40
5.5 Proximal distance algorithms 5.44
5.6 Summary 5.47

5.0 Introduction

Consider again the **LASSO / sparse regression / compressed sensing** optimization problem:

$$\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}} \frac{1}{2} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \beta \|\boldsymbol{x}\|_1 . \tag{5.1}$$

The previous chapter developed the following **majorize-minimize** (**MM**) algorithm for this optimization problem based on a separable quadratic majorizer of the data term:

$$\boldsymbol{x}_{k+1} = \mathsf{soft}\left(\boldsymbol{x}_k - \boldsymbol{D}^{-1} \nabla \Psi(\boldsymbol{x}_k), \beta \oslash \boldsymbol{d}\right)$$

where $\boldsymbol{A}'\boldsymbol{A} \preceq \boldsymbol{D} = \mathsf{Diag}\{\boldsymbol{d}\}$. Another name for it is the **iterative soft thresholding algorithm** (**ISTA**).

This chapter develops extensions that are faster and/or more general.
Particularly important is the (recent: 2017!) **proximal optimized gradient method** (**POGM**) [1].

The **proximal methods** in this chapter are especially useful for **composite** cost functions:

$$\Psi(\boldsymbol{x}) = \underbrace{f(\boldsymbol{x})}_{\text{smooth}} + \underbrace{g(\boldsymbol{x})}_{\hookrightarrow \text{ prox friendly}}.$$

5.1 Proximal basics

Proximal operator

We begin with an operation appearing in many optimization algorithms.

Define. Let $\mathcal{X}$ denote a vector space with norm $\|\cdot\|_{\mathcal{X}}$. The **proximal operator** [2, 3] (also called the **proximal mapping** [4, Ch. 6]) associated with a (typically **convex**) function $f : \mathcal{X} \mapsto \mathbb{R}$, is defined, for every $\boldsymbol{v} \in \mathcal{X}$, as the following minimizer:

$$\operatorname{prox}_f(\boldsymbol{v}) \triangleq \arg\min_{\boldsymbol{x} \in \mathcal{X}} \frac{1}{2} \|\boldsymbol{x} - \boldsymbol{v}\|_{\mathcal{X}}^2 + f(\boldsymbol{x}). \tag{5.2}$$

- The usual case is $\mathcal{X} = \mathbb{R}^N$ or $\mathcal{X} = \mathbb{C}^N$ and $\|\cdot\|_2$.
- Usually the norm squared is strictly convex and f is convex so the "arg min" is unique for any $\boldsymbol{v} \in \mathcal{X}$.
- Often both the norm squared and the function f are additively separable, and $\mathcal{X}$ is a **Cartesian product space** like $\mathbb{F}^N$, in which case the "arg min" separates into N individual 1D minimization problems.
- The proximal operator transforms one function $f(\cdot)$ into another function $\operatorname{prox}_f(\cdot)$.

In general, the functions f and prox_f have the same (domain?, **codomain**?)

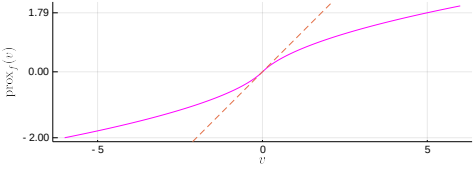
A: true,true B: true,false C: false,true D: false,false **??**

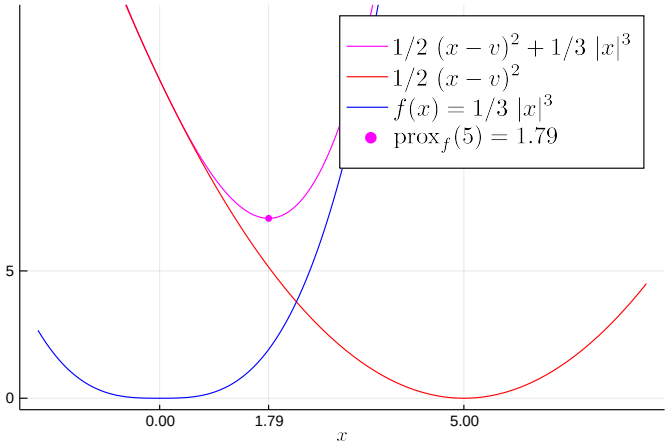
??

Example. For $\mathcal{X} = \mathbb{R}$ and $f(x) = \frac{1}{3}|x|^3$ we have

$$\begin{aligned}\mathrm{prox}_f(v) &= \arg\min_{x \in \mathbb{R}} \frac{1}{2}(x-v)^2 + \frac{1}{3}|x|^3 \\ &= \mathrm{sign}(v)(-1 + \sqrt{1 + 4\,|v|})/2,\end{aligned}$$

because for $v > 0$ the minimizer will be where $x > 0$, for which the derivative is $x - v + x^2$.





Here we need not treat the case $v = 0$ or $x = 0$ separately, because $|x|^3$ is continuously differentiable.

Example. Now we consider a more general case where the vector space is not $\mathbb{F}^N$ and the norm is not $\|\cdot\|_2$.

Let $\mathcal{X} = \mathbb{F}^{M \times N}$ and $\mathcal{X}_K = \left\{ \boldsymbol{X} \in \mathbb{F}^{M \times N} \ : \ \mathsf{rank}(\boldsymbol{X}) \le K \right\}$ and $f(\boldsymbol{X}) = \chi_{\mathcal{X}_K}(\boldsymbol{X})$ for $K \ge 0$, and consider the **Frobenius norm** $\|\cdot\|_{\mathrm{F}}$.
Then (from EECS 551) the **proximal operator** for this f is **singular value hard thresholding**:

$$\begin{aligned} \mathrm{prox}_f(\boldsymbol{Y}) &= \arg\min_{\boldsymbol{X} \in \mathbb{F}^{M \times N}} \frac{1}{2} \|\boldsymbol{X} - \boldsymbol{Y}\|_{\mathrm{F}}^2 + \chi_{\mathcal{X}_K}(\boldsymbol{X}) \\ &= \arg\min_{\boldsymbol{X} \in \mathcal{X}_K} \frac{1}{2} \|\boldsymbol{X} - \boldsymbol{Y}\|_{\mathrm{F}}^2 = \sum_{k=1}^{\min(K,M,N)} \sigma_k \boldsymbol{u}_k \boldsymbol{v}_k', \quad \boldsymbol{Y} = \sum_{k=1}^{\min(M,N)} \sigma_k \boldsymbol{u}_k \boldsymbol{v}_k'. \end{aligned}$$

The function f here is nonconvex, because $\mathcal{X}_K$ is a nonconvex set, and $\mathcal{X} = \mathbb{F}^{M \times N}$ is the vector space of $M \times N$ matrices, so this example illustrates the generality of the proximal operator.

In this case the minimizer is not unique if the SVD of $\boldsymbol{Y}$ has $\sigma_K = \sigma_{K+1}$ so the proximal operator is not quite well defined for such inputs, but in practical use the lack of uniqueness does not matter.

If $0 \le K \le \min(M,N)$, then $\left\| \boldsymbol{Y} - \mathrm{prox}_f(\boldsymbol{Y}) \right\|_2^2 = \sum_{k=K+1}^{\min(M,N)} \sigma_k^2$. (?)
A: True B: False **??**

For non-convex functions f we could define:

$$\operatorname{prox}_f(\boldsymbol{v}) \in \arg\min_{\boldsymbol{x}} \frac{1}{2} \|\boldsymbol{x} - \boldsymbol{v}\|_{\mathcal{X}}^2 + f(\boldsymbol{x}),$$

implying that the proximal operator may return any of the (global) minimizers.

Example.

$$\operatorname{prox}_{\mathcal{X}_1}\left(\begin{bmatrix} 1 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 4 \end{bmatrix} \right) \in \left\{ \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 4 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 0 \end{bmatrix} \right\}.$$

Example. For $f(\boldsymbol{x}) = \beta \frac{1}{2} \|\boldsymbol{x}\|_2^2$, the proximal operator is

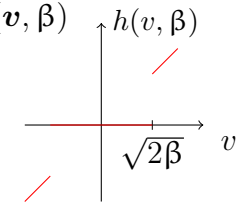
$$\operatorname{prox}_{\frac{\beta}{2}\|\cdot\|_2}(\boldsymbol{v}) = \frac{1}{1+\beta} \boldsymbol{v}.$$

Challenge. Determine the proximal operator of $f(\boldsymbol{x}) = \beta \|\boldsymbol{x}\|_2$.

Example. For $\mathcal{X} = \mathbb{F}^N$ and $\|\cdot\|_2$ and the (nonconvex) 0-norm $f(\boldsymbol{x}) = \beta \|\boldsymbol{x}\|_0$, the **proximal operator** is (element-wise) **hard thresholding**:

$$\operatorname{prox}_f(\boldsymbol{v}) = \arg\min_{\boldsymbol{x} \in \mathbb{F}^N} \frac{1}{2} \|\boldsymbol{x} - \boldsymbol{v}\|_2^2 + \beta \|\boldsymbol{x}\|_0 = \arg\min_{\boldsymbol{x} \in \mathbb{F}^N} \sum_{n=1}^{N} \frac{1}{2} |v_n - x_n|^2 + \beta \mathbb{I}_{\{x_n \neq 0\}} = h.(\boldsymbol{v}, \beta)$$

$$h(v, \beta) = \arg\min_{x \in \mathbb{F}} \frac{1}{2} |x - v|^2 + \beta \mathbb{I}_{\{x \neq 0\}} = \begin{cases} v, & \frac{1}{2}|v|^2 > \beta \\ 0, & \text{otherwise.} \end{cases}$$



Note that the threshold is $\sqrt{2\beta}$ due to the $1/2$ in front of the norm in the proximal operator definition. Again the minimizer is not unique if $|v| = \sqrt{2\beta}$; in such cases, both v and 0 are global minimizers.

This example illustrates the very common case where the norm squared and the function f are both **additively separable** so the minimization problem simplifies.

In JULIA code: `hard = (v,beta) -> v .* (abs.(v).^2 .> 2beta)`

If we replace $\beta \|\boldsymbol{x}\|_0$ with a *weighted* 0-norm: $f(\boldsymbol{x}) = \sum_{n=1}^{N} \beta_n \mathbb{I}_{\{x_n \neq 0\}}$, then the JULIA code for the proximal operator of this f is still the same as above, where `beta` is now a `Vector`. (?)

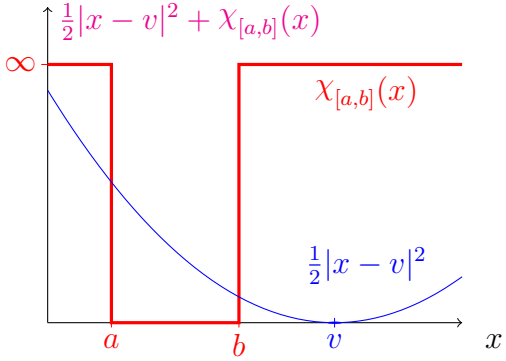
A: True B: False **??**

Example. Consider $\mathcal{X} = \mathbb{R}$ and $f(x) = \chi_{[a,b]}(x)$ with $-\infty < a < b < \infty$.

$$\text{prox}_f(v) = \arg\min_{x \in \mathbb{R}} \frac{1}{2}|x-v|^2 + \chi_{[a,b]}(x)$$

$$= \arg\min_{a \le x \le b} \frac{1}{2}|x-v|^2 = \begin{cases} a, & v < a \\ b, & b < v \\ v, & \text{otherwise} \end{cases}$$

$$= \texttt{clamp(v,a,b)}$$



Dictionary: https://en.wiktionary.org/wiki/proximal
proximal: "Closer to the point of attachment or observation."
"The proximal end of the arm connects to the shoulder."
From Latin "proximus" meaning "nearest."

The value of $\text{prox}_{\chi_{[-3,-2]}(\cdot)}(-4) = ?$

A: 0 B: -1 C: -2 D: -3 E: -4 **??**

Example. If $\mathcal{C} \subset \mathbb{F}^N$ is a **closed convex set** and $f(\boldsymbol{x}) = \chi_{\mathcal{C}}(\boldsymbol{x})$ is its **characteristic function**, then:

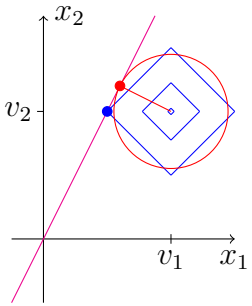
$$\operatorname{prox}_f(\boldsymbol{v}) = \arg\min_{\boldsymbol{x} \in \mathbb{F}^N} \frac{1}{2} \|\boldsymbol{x} - \boldsymbol{v}\|^2 + \chi_{\mathcal{C}}(\boldsymbol{x}) =$$

Intuitively, this example might be how the term **proximal** originated!

In this example, the choice of norm $\|\cdot\|$ matters, *i.e.*, affects the value of $\operatorname{prox}_f(\cdot)$ (?)

A: True B: False **??**

Example. Consider the half plane:
$\mathcal{C} = \{(x_1, x_2) \ : \ x_2 \geq 2x_1\} \subset \mathbb{R}^2$.

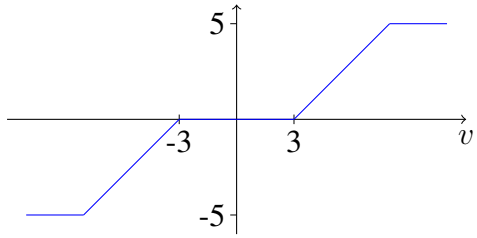


Some combinations of functions are still amenable to easy proximal operations.

Example. Consider $f(\boldsymbol{x}) = f_1(\boldsymbol{x}) + f_2(\boldsymbol{x}),\ f_1(\boldsymbol{x}) = 3\left\|\boldsymbol{x}\right\|_1,\ f_2(\boldsymbol{x}) = \chi_{\mathcal{C}}(\boldsymbol{x}),\ \mathcal{C} = \left\{\boldsymbol{x} \in \mathbb{R}^N \ : \ \left\|\boldsymbol{x}\right\|_\infty \le 5\right\}.$

Exercise. Verify that

$$\text{prox}_f(\boldsymbol{v}) = \text{softclamp.}(\boldsymbol{v}), \quad \text{softclamp.}(v) =$$

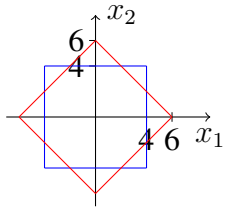


This case is easy because the box constraint is **separable**:

$$\mathcal{C} = \bigcap_{n=1}^{N} \left\{\boldsymbol{x} \in \mathbb{R}^N \ : \ |x_n| \le 5\right\} \implies f_2(\boldsymbol{x}) = \sum_{n=1}^{N} \chi_{[-5,5]}(x_n).$$

Other combinations are not amenable to easy proximal operations.

Example. Consider $f(\boldsymbol{x}) = \chi_{\mathcal{C}_1}(\boldsymbol{x}) + \chi_{\mathcal{C}_2}(\boldsymbol{x}) = \chi_{\mathcal{C}_1 \cap \mathcal{C}_2}(\boldsymbol{x}) = \chi_{\mathcal{C}}(\boldsymbol{x}),\ \mathcal{C} = \mathcal{C}_1 \cap \mathcal{C}_2$
$\mathcal{C}_1 = \left\{\boldsymbol{x} \in \mathbb{R}^N \ : \ \left\|\boldsymbol{x}\right\|_\infty \le 4\right\},\ \mathcal{C}_2 = \left\{\boldsymbol{x} \in \mathbb{R}^N \ : \ \left\|\boldsymbol{x}\right\|_1 \le 6\right\}.$
Here prox_f is complicated in $\mathbb{R}^N$, because $\mathcal{C}_2$ is not separable.



Properties of proximal operators

There seem to be relatively few general properties of the **proximal operator**. Even something as simple looking as $\operatorname{prox}_{f+g}$ is complicated to analyze [5].

Define. A "**prox friendly**" function f is one where $\operatorname{prox}_f(\cdot)$ is "easy" to compute.

Example. $f(\boldsymbol{x}) = \beta \|\boldsymbol{T}\boldsymbol{x}\|_1$ is prox friendly where $\boldsymbol{T}$ is **unitary**, but it is not prox friendly for general $\boldsymbol{T}$:

$$\boldsymbol{T} = \boldsymbol{T}^{-1} \implies \operatorname{prox}_{\beta\|\boldsymbol{T}\cdot\|_1}(\boldsymbol{v}) = \arg\min_{\boldsymbol{x}} \frac{1}{2}\|\boldsymbol{x}-\boldsymbol{v}\|_2^2 + \beta\|\boldsymbol{T}\boldsymbol{x}\|_1 = \boldsymbol{T}' \operatorname{soft}(\boldsymbol{T}\boldsymbol{v}, \beta). \tag{5.3}$$

The derivation uses the change of variables $\boldsymbol{z} = \boldsymbol{T}\boldsymbol{x}$ so $\boldsymbol{x} = \boldsymbol{T}^{-1}\boldsymbol{z} = \boldsymbol{T}'\boldsymbol{z}$. Thus

$$\operatorname{prox}_{\beta\|\boldsymbol{T}\cdot\|_1}(\boldsymbol{v}) = \boldsymbol{T}'\hat{\boldsymbol{z}}$$

$$\hat{\boldsymbol{z}} = \arg\min_{\boldsymbol{z}} \frac{1}{2}\|\boldsymbol{T}'\boldsymbol{z}-\boldsymbol{v}\|_2^2 + \beta\|\boldsymbol{z}\|_1 = \arg\min_{\boldsymbol{z}} \frac{1}{2}\|\boldsymbol{z}-\boldsymbol{T}\boldsymbol{v}\|_2^2 + \beta\|\boldsymbol{z}\|_1 = \operatorname{soft}(\boldsymbol{T}\boldsymbol{v}, \beta).$$

Exercise. Generalize this example to the case $f(\boldsymbol{x}) = \|\boldsymbol{W}\boldsymbol{T}\boldsymbol{x}\|_1$, where $\boldsymbol{W} = \operatorname{Diag}\{\boldsymbol{w}\}$ with $\boldsymbol{w} \geq \boldsymbol{0}$.

In general, suppose we have $\operatorname{prox}_f(\cdot)$ for some function $f : \mathbb{F}^N \mapsto \mathbb{F}^N$, and then we define $g(\boldsymbol{x}) = f(\boldsymbol{T}\boldsymbol{x})$ for a $N \times N$ unitary matrix $\boldsymbol{T}$; then $\operatorname{prox}_g(\boldsymbol{v}) = \boldsymbol{T}' \operatorname{prox}_f(\boldsymbol{T}\boldsymbol{v})$. (?)

A: True B: False **??**

Complex cases (Read)

When $\mathcal{X} = \mathbb{C}$, often we need the proximal operator of a function $f(\cdot)$ that depends only on the magnitude of its argument, *i.e.*, $f(z) = g(|z|)$ where $f : \mathbb{C} \mapsto \mathbb{R}$ and $g : \mathbb{R} \mapsto \mathbb{R}$.
Applying the definition (5.2) to this case and simplifying:

$$\begin{aligned}
\operatorname{prox}_f(v) &= \arg\min_{z \in \mathbb{C}} \frac{1}{2} |v - z|^2 + f(z) = \arg\min_{m e^{\imath\phi} : m \in [0,\infty),\ \phi \in \mathbb{R}} \frac{1}{2} \left| v - m\, e^{\imath\phi} \right|^2 + f(m\, e^{\imath\phi}) \\
&= \arg\min_{m e^{\imath\phi} : m \in [0,\infty),\ \phi \in \mathbb{R}} \frac{1}{2} \left| |v|\, e^{\imath \angle v} - m\, e^{\imath\phi} \right|^2 + g(m) \\
&= m_* e^{\imath\phi_*}, \quad \phi_* = \angle v, \quad m_* = \arg\min_{m \in [0,\infty)} \frac{1}{2} \left| |v| - m \right|^2 + g(m) = \operatorname{prox}_g(|v|),
\end{aligned}$$

because $\left| |v|\, e^{\imath \angle v} - m\, e^{\imath\phi} \right|^2 = |v|^2 + m^2 - 2\cos(\phi - \angle v)$ is minimized at $\phi_* = \angle v$.

Thus, to determine the proximal operator for a function that depends only on the magnitude of its argument, we simply determine the proximal operator for the magnitude and apply the sign of the argument at the end.

Example. For $f(\boldsymbol{x}) = c \|\boldsymbol{x}\|_1 = c \sum_{n=1}^N |x_n|$ the proximal operator in JULIA code is

```
soft = (z,c) -> sign(z) * max(abs(z) - c, 0)
prox = (v,c) -> soft.(v)
```

The `sign` function in JULIA (and MATLAB) returns either $e^{\imath \angle v}$ or 0, as desired here.

Proximal point algorithm

The **proximal point algorithm** for minimizing Ψ has the following deceptively simple looking form:

$$\boldsymbol{x}_{k+1} = \arg\min_{\boldsymbol{x}} \left(\frac{1}{2} \left\| \boldsymbol{x} - \boldsymbol{x}_k \right\|^2 + \alpha_k \, \Psi(\boldsymbol{x}) \right) = \tag{5.4}$$

for any sequence of positive values $\{\alpha_k\}$.

Taylor et al. establish the following tight worst-case bound on the cost function decrease [1, p. 1301]:

$$\Psi(\boldsymbol{x}_N) - \Psi(\hat{\boldsymbol{x}}) \leq \frac{\left\| \boldsymbol{x}_0 - \hat{\boldsymbol{x}} \right\|^2}{4 \sum_{k=1}^{N} \alpha_k}. \tag{5.5}$$

To show the bound is tight, they provide a simple 1D function $\Psi(x) = c\,|x|$ for an appropriate c that depends on N, $\|\boldsymbol{x}_0 - \hat{\boldsymbol{x}}\|$ and $\{\alpha_k\}$.

At first glance this method seems useless because the minimization problem in (5.4) looks to be as hard as minimizing Ψ, if not harder. However, the norms in (5.4) and (5.5) must be the same, but they need not be the standard Euclidean norm! That norm choice does not affect the final value of $\Psi(\hat{\boldsymbol{x}})$.

Example. Consider the LASSO / compressed sensing cost function (5.1) and suppose we use the usual Euclidean norm in (5.4). Then the update would be the following difficult minimization problem:

$$\boldsymbol{x}_{k+1} = \arg\min_{\boldsymbol{x}} \frac{1}{2} \left\| \boldsymbol{x} - \boldsymbol{x}_k \right\|_2^2 + \alpha_k \left(\frac{1}{2} \left\| \boldsymbol{A}\boldsymbol{x} - \boldsymbol{y} \right\|_2^2 + \beta \left\| \boldsymbol{x} \right\|_1 \right).$$

Suppose instead though that, drawing inspiration from Ch. 4, we assume that the α_k values are a constant: $\alpha_k = \alpha, \ \forall k$, and we choose the weighted norm

$$\|\boldsymbol{v}\|^2 = \|\boldsymbol{v}\|^2_{\boldsymbol{D} - \alpha \boldsymbol{A}'\boldsymbol{A}} = \boldsymbol{v}' \left(\boldsymbol{D} - \alpha \boldsymbol{A}'\boldsymbol{A} \right) \boldsymbol{v},$$

where we choose $\boldsymbol{D} \succ \boldsymbol{0}$ to satisfy the majorization condition: $\alpha \boldsymbol{A}'\boldsymbol{A} \preceq \boldsymbol{D} = \mathsf{Diag}\{\boldsymbol{d}\}$.

With this choice of norm, the proximal point algorithm update is

$$\begin{aligned}
\boldsymbol{x}_{k+1} &= \arg\min_{\boldsymbol{x}} \frac{1}{2} \|\boldsymbol{x} - \boldsymbol{x}_k\|^2_{\boldsymbol{D} - \alpha \boldsymbol{A}'\boldsymbol{A}} + \alpha \left(\frac{1}{2} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \beta \|\boldsymbol{x}\|_1 \right) \\
&= \arg\min_{\boldsymbol{x}} \frac{1}{2} (\boldsymbol{x} - \boldsymbol{x}_k)' \left(\boldsymbol{D} - \alpha \boldsymbol{A}'\boldsymbol{A} \right) (\boldsymbol{x} - \boldsymbol{x}_k) + \alpha \frac{1}{2} (\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y})' (\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}) + \alpha\beta \|\boldsymbol{x}\|_1 \\
&= \ldots \arg\min_{\boldsymbol{x}} \frac{1}{2} \left\| \boldsymbol{x} - \left(\boldsymbol{x}_k - \alpha \boldsymbol{D}^{-1} \boldsymbol{A}' (\boldsymbol{A}\boldsymbol{x}_k - \boldsymbol{y}) \right) \right\|^2_{\boldsymbol{D}} + \alpha\beta \|\boldsymbol{x}\|_1 + c \\
&= \mathsf{soft} \left(\boldsymbol{x}_k - \alpha \boldsymbol{D}^{-1} \boldsymbol{A}' (\boldsymbol{A}\boldsymbol{x}_k - \boldsymbol{y}), \alpha\beta \oslash \boldsymbol{d} \right).
\end{aligned}$$

Choosing larger α accelerates the convergence rate (?)

A: True B: False **??**

When we choose constant $\alpha_k = \alpha$, the cost function worst-case convergence rate is $O(1/k)$ (?)

A: True B: False **??**

You might think at this point that the **proximal point algorithm** looks basically like a **MM method**.
I agree! If you find any example that is not a MM method, please let me know!

I include it here for these reasons:

- To see a "simple" use of a **proximal operator** in an optimization algorithm.
- To relate two terms you will see in the literature (proximal point and MM).
- To distinguish the **proximal point algorithm** from the **proximal gradient method** (**PGM**) coming up, because they have similar sounding names but are different.
  When someone says "I used a proximal method" it is ambiguous; probably they mean a proximal gradient method, but you might need to seek clarification.

A majorizer of Ψ related to (5.4) is the following function: $\phi_k(\boldsymbol{x}) \triangleq \frac{1}{2\alpha_k} \|\boldsymbol{x} - \boldsymbol{x}_k\|^2 + \Psi(\boldsymbol{x})$?

A: True B: False **??**

An interesting recent extension uses the Moreau-Yosida regularization of a cost function and OGM-type momentum to address convex but nonsmooth problems [6, 7].

An alternating minimization approach is perhaps more compelling [8].

5.2 Proximal gradient method (PGM)

Consider the **composite** cost function:

$$\Psi(\boldsymbol{x}) = \underbrace{f(\boldsymbol{x})}_{\text{smooth}} + \underbrace{g(\boldsymbol{x})}_{\hookrightarrow \text{ prox friendly}}, \tag{5.6}$$

where here by **smooth** we mean: f is differentiable and $\nabla f(\boldsymbol{x})$ is $\boldsymbol{S}$-Lipschitz continuous.

Then recall from Ch. 4 that a **quadratic majorizer** for $f(\boldsymbol{x})$ is

$$f(\boldsymbol{x}) \le q_1(\boldsymbol{x}; \boldsymbol{x}_k) \triangleq f(\boldsymbol{x}_k) + \text{real}\{\langle \nabla f(\boldsymbol{x}_k), \, \boldsymbol{x} - \boldsymbol{x}_k \rangle\} + \frac{1}{2} \left\| \boldsymbol{S}'(\boldsymbol{x} - \boldsymbol{x}_k) \right\|_2^2 .$$

Slightly more generally, for any $0 < \gamma \le 1$ the following function is also a **quadratic majorizer** for f:

$$\begin{aligned} f(\boldsymbol{x}) \le q_\gamma(\boldsymbol{x}; \boldsymbol{x}_k) &\triangleq f(\boldsymbol{x}_k) + \text{real}\{\langle \nabla f(\boldsymbol{x}_k), \, \boldsymbol{x} - \boldsymbol{x}_k \rangle\} + \frac{1}{2\gamma} \left\| \boldsymbol{S}'(\boldsymbol{x} - \boldsymbol{x}_k) \right\|_2^2 \\ &= c(\boldsymbol{x}_k) + \frac{1}{2\gamma} \left\| \boldsymbol{S}' \left(\boldsymbol{x} - \left(\boldsymbol{x}_k - \gamma \left(\boldsymbol{S}\boldsymbol{S}' \right)^{-1} \nabla f(\boldsymbol{x}_k) \right) \right) \right\|_2^2 , \end{aligned}$$

after completing the square, where $c(\boldsymbol{x}_k)$ is an irrelevant constant independent of $\boldsymbol{x}$.
Note $f \le q_1 \le q_\gamma$ and $f(\boldsymbol{x}_k) = q_1(\boldsymbol{x}_k; \boldsymbol{x}_k) = q_\gamma(\boldsymbol{x}_k; \boldsymbol{x}_k)$.

Then an **MM algorithm** for minimizing Ψ using the majorizer $\phi_k(\boldsymbol{x}) \triangleq q_\gamma(\boldsymbol{x}; \boldsymbol{x}_k) + g(\boldsymbol{x})$ is:

$$\begin{aligned} \boldsymbol{x}_{k+1} &= \arg\min_{\boldsymbol{x}} q_\gamma(\boldsymbol{x}; \boldsymbol{x}_k) + g(\boldsymbol{x}) = \arg\min_{\boldsymbol{x}} \frac{1}{2\gamma} \left\| \boldsymbol{S}' \left(\boldsymbol{x} - \left(\boldsymbol{x}_k - \gamma \left(\boldsymbol{S}\boldsymbol{S}' \right)^{-1} \nabla f(\boldsymbol{x}_k) \right) \right) \right\|_2^2 + g(\boldsymbol{x}) \\ &= \operatorname{prox}_{\gamma g}\left(\boldsymbol{x}_k - \gamma \left(\boldsymbol{S}\boldsymbol{S}' \right)^{-1} \nabla f(\boldsymbol{x}_k) \right). \end{aligned} \tag{5.7}$$

With the notation convention $\|\boldsymbol{x}\|_{\boldsymbol{W}}^2 = \boldsymbol{x}'\boldsymbol{W}\boldsymbol{x}$, the proximal operator in (5.7) uses norm:

A: $\|\cdot\|_{\boldsymbol{S}}$ B: $\|\cdot\|_{\boldsymbol{S}'}$ C: $\|\cdot\|_{\boldsymbol{S}'\boldsymbol{S}}$ D: $\|\cdot\|_{\boldsymbol{S}\boldsymbol{S}'}$ E: $\|\cdot\|_{(\boldsymbol{S}\boldsymbol{S}')^{-1}}$ **??**

Usually the proximal operator is simple to compute only if $\boldsymbol{S}$ is diagonal.

Because of the $\boldsymbol{S}$, the MM algorithm (5.7) is a generalization of the classical **proximal gradient method** (**PGM**) that appears in most textbooks. Here we define **PGM** using this generality.

Define. For a composite cost function (5.6) where ∇f has a $\boldsymbol{S}$-Lipschitz gradient, the **proximal gradient method** (**PGM**) with step size factor γ is

$$\begin{aligned} \boldsymbol{x}_{k+1} &= \arg\min_{\boldsymbol{x}} \frac{1}{2\gamma} \left\| \boldsymbol{S}' \left(\boldsymbol{x} - \left(\boldsymbol{x}_k - \gamma \left(\boldsymbol{S}\boldsymbol{S}' \right)^{-1} \nabla f(\boldsymbol{x}_k) \right) \right) \right\|_2^2 + g(\boldsymbol{x}) \\ &= \operatorname{prox}_{\gamma g}\left(\boldsymbol{x}_k - \gamma \left(\boldsymbol{S}\boldsymbol{S}' \right)^{-1} \nabla f(\boldsymbol{x}_k) \right), \end{aligned} \tag{5.8}$$

where this prox uses the weighted norm from the preceding clicker question.

The classical or "textbook" choice for the PGM is when $\boldsymbol{S} = \sqrt{L}\boldsymbol{I}$, where L denotes a **Lipschitz constant** of ∇f. In this case the PGM simplifies to

$$\boldsymbol{x}_{k+1} = \arg\min_{\boldsymbol{x}} \frac{L}{2\gamma} \left\| \boldsymbol{x} - \left(\boldsymbol{x}_k - \frac{\gamma}{L} \nabla f(\boldsymbol{x}_k) \right) \right\|_2^2 + g(\boldsymbol{x}) \tag{5.9}$$

$$= \arg\min_{\boldsymbol{x}} \frac{1}{2\alpha} \left\| \boldsymbol{x} - (\boldsymbol{x}_k - \alpha \nabla f(\boldsymbol{x}_k)) \right\|_2^2 + g(\boldsymbol{x})$$

$$= \operatorname{prox}_{?g}(\boldsymbol{x}_k - \alpha \nabla f(\boldsymbol{x}_k)), \tag{5.10}$$

where this expression uses the **proximal operator** for the usual Euclidean norm and we let $\alpha \triangleq \gamma / L$.

The constant that goes in front of g in (5.10) is

A: 1 B: $1/\alpha$ C: $1/\alpha^2$ D: α E: α^2 **??**

Note that the argument inside the prox is the usual gradient method update, hence the PGM name.

If the regularizer g is "prox friendly" then PGM is a very simple **first-order algorithm**.

Example. If $g(\boldsymbol{x}) = \beta \|\boldsymbol{x}\|_1$ then the PGM update (5.10) is simply a gradient update with step size $\alpha = \gamma/L$ followed by soft thresholding, with threshold . So strictly speaking, **PGM** is a generalization of **ISTA**. However, in practice people often use the terms ISTA and PGM interchangeably.

A possible use of the γ factor is in large-scale regularized least-squares problems where $L = \|\boldsymbol{A}\|_2^2$ is impractical to compute. In such problems we can use

$$\gamma = \frac{\|\boldsymbol{A}\|_2^2}{\|\boldsymbol{A}\|_1 \|\boldsymbol{A}\|_\infty} \le 1 \implies \alpha_1 = \frac{\gamma}{L} = \frac{1}{\|\boldsymbol{A}\|_1 \|\boldsymbol{A}\|_\infty},$$

which is a practical step size.
Alternatively, we could arrive at the same update by choosing $\gamma = 1$ and $\boldsymbol{S} = \sqrt{\|\boldsymbol{A}\|_1 \|\boldsymbol{A}\|_\infty}\, \boldsymbol{I} = (1/\sqrt{\alpha_1})\boldsymbol{I}$.

When $g = 0$, for what range of value of γ does the **PGM** update (5.8) ensure a monotonic decrease: $\Psi(\boldsymbol{x}_{k+1}) < \Psi(\boldsymbol{x}_k)$, when $\boldsymbol{x}_k$ is not a minimizer? (Choose most general correct answer.)
A: $0 \le \gamma$ B: $0 < \gamma$ C: $0 < \gamma \le 1$ D: $0 < \gamma < 2$ E: $\gamma = 1$ **??**

Because q_1 is a majorizer for Ψ when $g = 0$, it suffices to verify that $q_1(\boldsymbol{x}_{k+1}; \boldsymbol{x}_k) < q_1(\boldsymbol{x}_k; \boldsymbol{x}_k)$, because that in turn ensures descent of Ψ by the **sandwich inequality** in Ch. 4. Let $\boldsymbol{g}_k = \nabla f(\boldsymbol{x}_k)$ and use (5.8):

$$q_1(\boldsymbol{x}_k; \boldsymbol{x}_k) - q_1(\boldsymbol{x}_{k+1}; \boldsymbol{x}_k) = f(\boldsymbol{x}_k) - \left(f(\boldsymbol{x}_k) + \mathrm{real}\{\langle \nabla f(\boldsymbol{x}_k),\, \boldsymbol{x}_{k+1} - \boldsymbol{x}_k \rangle\} + \frac{1}{2} \|\boldsymbol{S}'(\boldsymbol{x}_{k+1} - \boldsymbol{x}_k)\|_2^2 \right)$$

$$= \mathrm{real}\left\{ \langle \boldsymbol{g}_k,\, \gamma(\boldsymbol{S}\boldsymbol{S}')^{-1} \boldsymbol{g}_k \rangle \right\} - \frac{1}{2} \left\| \boldsymbol{S}' \gamma (\boldsymbol{S}\boldsymbol{S}')^{-1} \boldsymbol{g}_k \right\|_2^2 = \gamma\,(1 - \gamma/2)\, \boldsymbol{g}_k' (\boldsymbol{S}\boldsymbol{S}')^{-1} \boldsymbol{g}_k > 0,$$

if $0 < \gamma < 2$ and $\boldsymbol{g}_k \neq \boldsymbol{0}$, *i.e.*, if $\boldsymbol{x}_k$ is not a minimizer.

https://www.google.com/search?q=plot+x*(1-x/2)

For a general convex g and $\boldsymbol{S}$, for what range of value of γ does the **PGM** update (5.8) ensure a monotonic decrease: $\Psi(\boldsymbol{x}_{k+1}) \le \Psi(\boldsymbol{x}_k)$? (Choose most general correct answer.)
Assume the norm for "prox" is $\|\cdot\|_{\boldsymbol{S}\boldsymbol{S}'}$.

A: $0 \le \gamma$ B: $0 < \gamma$ C: $0 < \gamma \le 1$ D: $0 < \gamma < 2$ E: $\gamma = 1$ **??**

It certainly holds for $0 < \gamma \le 1$ because q_γ is a majorizer for that range of γ values.

When $g = \chi_{\mathcal{C}}$ for a convex set $\mathcal{C}$, then **PGM** is the same as the **gradient projection method**, and one can prove [9, p. 207] convergence of $\{\boldsymbol{x}_k\}$ in (5.10) to a constrained minimizer for $0 < \alpha < 2/L$.
However, that proof does not show monotonicity! [9, Pr. 4 on p. 210] implies the cost is non-decreasing.

Challenge. Examine the case where $1 < \gamma < 2$ for other g functions.

Convergence rate of PGM

Recall from Ch. 3, specifically (3.31), that there is a (convex, smooth) Huber-like function for which the convergence rate of GD is exactly [10]:

$$\Psi(\boldsymbol{x}_k) - \Psi(\hat{\boldsymbol{x}}) = \frac{\left\| \boldsymbol{S}' (\boldsymbol{x}_0 - \hat{\boldsymbol{x}}) \right\|^2}{4k+2}, \quad k \geq 1. \tag{5.11}$$

Thus the worst-case rate of PGM with $\alpha = 1/L$ can be no better than $O(1/k)$.
Simply take $f(\boldsymbol{x})$ to be that Huber function and $g(\boldsymbol{x}) = 0$, then PGM with $\alpha = 1$ is the same as GD.

Could the worst-case rate be worse than $O(1/k)$ when g is nonzero?
Fortunately, the answer is no, when f is **convex** and smooth with L-Lipschitz gradient.
For $\alpha = 1/L$ there is the following classic bound for PGM [11]:

$$\Psi(\boldsymbol{x}_k) - \Psi(\hat{\boldsymbol{x}}) \leq \frac{\left\| \boldsymbol{S}' (\boldsymbol{x}_0 - \hat{\boldsymbol{x}}) \right\|^2}{2k}, \quad k \geq 1.$$

I do not know if this is a tight bound. (Probably not.)
I do not know what the tight bound is for $\alpha \neq 1/L$.
In summary, the worst-case rate of PGM for composite cost functions where ∇f is Lipschitz is $O(1/k)$.

Linear convergence rate for POGM (Read)

More recent analysis has shown that under a certain condition called the proximal-PL property, where PL stands for Polyak-Łojasiewicz, PGM converges at a **linear rate**, meaning there exists $\mu \in (0, L)$ such that [12, Thm. 5]: ♦♦

$$\Psi(\boldsymbol{x}_k) - \Psi_* \leq \left(1 - \frac{\mu}{L}\right)^k (\Psi(\boldsymbol{x}_0) - \Psi_*) \,. \tag{5.12}$$

In the early iterations, the $O(1/k)$ bound is probably more useful, but the $O(\rho^k)$ bound becomes more applicable asymptotically.

The PL property for a differentiable function f is

$$\frac{1}{2} \|\nabla f(\boldsymbol{x})\|_2^2 \geq \mu \left(f(\boldsymbol{x}) - f_*\right).$$

The proximal-PL property for composite cost function $\Psi = f + g$, where ∇f is L-Lipschitz, is

$$\frac{1}{2} D_g(\boldsymbol{x}, L) \geq \mu \left(\Psi(\boldsymbol{x}) - \Psi_*\right), \quad D_g(\boldsymbol{x}, a) \triangleq -2a \min_{\boldsymbol{z}} \left(\langle \nabla f(\boldsymbol{x}), \, \boldsymbol{z} - \boldsymbol{x} \rangle + \frac{a}{2} \|\boldsymbol{z} - \boldsymbol{x}\|_2^2 + g(\boldsymbol{z}) - g(\boldsymbol{x}) \right).$$

The bound (5.12) will eventually lie below the equality in (5.11), which at first glance might seem contradictory. The reason is that the function that leads to the worst-case bound in (5.11) depends on how many iterations one plans to run, *i.e.*, the value of k. In contrast, the bound (5.12) holds for all k, for any Ψ that satisfies the proximal-PL property. Having to specify the number of iterations when proving worst-case bounds seems annoying but apparently necessary.

Often the μ in (5.12) is very small relative to L.

PGM for composite cost with strongly convex term

If $f(\boldsymbol{x})$ is **strongly convex** with strong convexity parameter μ and Lipschitz constant L, then there are nice very recent results showing *linear convergence* rates for PGM [13]:

$$\|\boldsymbol{x}_k - \hat{\boldsymbol{x}}\| \le \rho^{2k} \|\boldsymbol{x}_0 - \hat{\boldsymbol{x}}\|$$
$$\Psi(\boldsymbol{x}_k) - \Psi(\hat{\boldsymbol{x}}) \le \rho^{2k} \left(\Psi(\boldsymbol{x}_0) - \Psi(\hat{\boldsymbol{x}})\right),$$

where $0 \le \alpha < 2/L$ and

$$\rho \triangleq \rho(\alpha) = \max\left\{L\alpha - 1, 1 - \mu\alpha\right\}.$$

In light of these rates, the optimal step size α_* that provides the fastest convergence rate is:
A: 1 B: $1/L$ C: $(L-\mu)/L$ D: $2/(L+\mu)$ E: None of these **??**

Example. This analysis is useful for under-determined problems with **elastic net** regularization:

$$\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}} \frac{1}{2} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \beta \|\boldsymbol{x}\|_1 + \epsilon \frac{1}{2} \|\boldsymbol{x}\|_2^2.$$

Here we choose f and g appropriately , then $L =$ and, when $\boldsymbol{A}$ is wide, $\mu =$.

For the optimal α_*, the convergence factor is $\rho_* = \dfrac{L-\mu}{L+\mu}$ which can be very close to 1 when $\mu = \epsilon$ is small, so the linear convergence rate can still be slow.

PGM with line search (Read)

So far we considered PGM with a fixed step size α.
That approach requires that we know a Lipschitz constant L (or matrix $\boldsymbol{S}$).

An alternative algorithm is the **PGM with a line search** [13]:

$$\alpha_k = \arg\min_{\alpha \in \mathbb{R}} h_k(\alpha), \quad h_k(\alpha) \triangleq \Psi\left(\operatorname{prox}_{\alpha g}(\boldsymbol{x}_k - \alpha \nabla f(\boldsymbol{x}_k))\right)$$
$$\boldsymbol{x}_{k+1} = \operatorname{prox}_{\alpha_k g}(\boldsymbol{x}_k - \alpha_k \nabla f(\boldsymbol{x}_k)).$$

This line search is more complicated because of the proximal operator.

Open question:
If $f(\boldsymbol{x}) = \sum_{j=1}^{J} f_j(\boldsymbol{B}_j \boldsymbol{x})$ where each f_j has a Lipschitz gradient, is there an efficient line search procedure?
(The answer is yes when $g = 0$, and we have used that case in `pgm_inv_mm` etc.)

For the (somewhat hypothetical) case of an *exact* line search, as written above, if f is strongly convex with parameter μ, then PGM with a line search achieves the same rate as PGM with the optimal step size:

$$\Psi(\boldsymbol{x}_k) - \Psi(\hat{\boldsymbol{x}}) \leq \rho_*^{2k} \left(\Psi(\boldsymbol{x}_0) - \Psi(\hat{\boldsymbol{x}})\right).$$

This is a tight bound [13].

I am unsure about the rate for the case where f is smooth but not strongly convex;
I am confident it is $O(1/k)$ but with what constant?

Iterative hard thresholding

The preceding convergence rate theory is for **convex** composite cost functions, yet PGM has also been applied to **non-convex** cost functions.

A representative example is the sparsity regularized least-squares problem using a 0-norm:

$$\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x} \in \mathbb{F}^N} \Psi(\boldsymbol{x}), \quad \Psi(\boldsymbol{x}) = \frac{1}{2} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \beta \|\boldsymbol{x}\|_0 .$$

The proximal operator associated with $g(\boldsymbol{x}) = \beta \|\boldsymbol{x}\|_0$ is **hard thresholding**, as derived on p. 5.8.
Combining that proximal operator with the classic version of PGM (5.10) yields the iterative algorithm

$$\boldsymbol{x}_{k+1} = \mathsf{hard}(\boldsymbol{x}_k - (\boldsymbol{x}_k - \alpha \nabla f(\boldsymbol{x}_k)), \beta),$$

where as usual $f(\boldsymbol{x}) = \frac{1}{2} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2$. This algorithm is called **iterative hard thresholding** (**IHT**) [14].

The sequence generated by IHT decreases the cost function monotonically, *i.e.*, $\Psi(\boldsymbol{x}_{k+1}) \le \Psi(\boldsymbol{x}_k)$ for any initial $\boldsymbol{x}_0 \in \mathbb{F}^N$, if $0 < \alpha \le 1/\|\boldsymbol{A}\|_2^2$. (?)

A: True B: False **??**

Fact. The sequence $\{\boldsymbol{x}_k\}$ converges to a **local minimizer** of Ψ [14, Thm. 3].

There is also constrained version: (Read)

$$\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x} \in \mathbb{F}^N} \Psi(\boldsymbol{x}), \quad \Psi(\boldsymbol{x}) = \frac{1}{2} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \chi_{\mathcal{C}_K}(\boldsymbol{x}), \quad \mathcal{C}_K = \left\{ \boldsymbol{x} \in \mathbb{F}^N \ : \ \|\boldsymbol{x}\|_0 \le K \right\}.$$

Exercise. Determine the **proximal operator** for this case. **??**

- There are performance guarantees under certain **restricted isometry property** (**RIP**) conditions on $\boldsymbol{A}$.
- For example, if $\boldsymbol{x}$ is K-sparse and $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{\varepsilon}$ with $\|\boldsymbol{A}\|_2 \le 1$, then [15, eqn. (17)]:

$$\|\boldsymbol{x}_k - \boldsymbol{x}\|_2 \le 2^{-k} \|\boldsymbol{x}\|_2 + 5 \|\boldsymbol{\varepsilon}\|_2 .$$

- For comparison, if an oracle told us which elements of $\boldsymbol{x}$ are nonzero, then we could solve the corresponding LS problem directly as $\hat{\boldsymbol{x}}_K = \boldsymbol{A}_K^+ \boldsymbol{y}$, where $\boldsymbol{A}_K$ denotes the $M \times K$ matrix with the appropriate K columns of $\boldsymbol{A}$, for which $\|\hat{\boldsymbol{x}}_K - \boldsymbol{x}_K\|_2 = \left\|\boldsymbol{A}_K^+ \boldsymbol{\varepsilon}\right\|_2 \le \left\|\boldsymbol{A}_K^+\right\|_2 \|\boldsymbol{\varepsilon}\|_2$.
- There are similar bounds when $\boldsymbol{x}$ is approximately sparse [15, eqn. (13)].
- There are performance guarantees even for an accelerated version [16].
- The IHT algorithm above requires the Lipschitz constant $L = \|\boldsymbol{A}\|_2^2$ that is impractical for large-scale problems. An adaptive choice of the step size α is described in [17].
- After a finite number of iterations, the support set (set of nonzero elements of $\boldsymbol{x}_k$) remains fixed and the convergence rate is like that of GD using the corresponding columns of $\boldsymbol{A}$ [14, eqn. (2.8)].
- For generalizations beyond regularized LS see [18].
- For further generalizations see [19].

5.3 Accelerated proximal methods

Fast proximal gradient method (FPGM)

For minimizing a composite cost function $\Psi(\boldsymbol{x}) = f(\boldsymbol{x}) + g(\boldsymbol{x})$ where $\nabla f(\boldsymbol{x})$ is L-Lipschitz and g is prox friendly, the **Fast proximal gradient method** (**FPGM**), also known as the **fast iterative soft thresholding algorithm** (**FISTA**) is the following famous method:

Initialize: $\boldsymbol{z}_0 = \boldsymbol{x}_0$, then `for` $k = 1 : N$

$$\begin{aligned} \boldsymbol{z}_k &= \operatorname{prox}_{g/L}\left(\boldsymbol{x}_{k-1} - \frac{1}{L}\nabla f(\boldsymbol{x}_{k-1})\right) && \text{“PGM step”} \\ \boldsymbol{x}_k &= \boldsymbol{z}_k + \alpha_k\left(\boldsymbol{z}_k - \boldsymbol{z}_{k-1}\right) && \text{“momentum step”}, \end{aligned} \tag{5.13}$$

where the **momentum parameters** or **initial parameters** $\{\alpha_k\}$ have two standard choices:

- $\alpha_k = \frac{k-1}{k+2}$, and
- $\alpha_k = \dfrac{t_{k-1} - 1}{t_k}$, where $t_0 = 1,\ t_k = \frac{1}{2}\left(1 + \sqrt{1 + 4t_{k-1}^2}\right)$.

Both choices lead to this convergence bound on $\boldsymbol{z}_N$ [1]:

$$\Psi(\boldsymbol{z}_N) - \Psi_* \le \frac{2L\left\|\boldsymbol{x}_0 - \hat{\boldsymbol{x}}\right\|_2^2}{(N+1)^2},$$

where we assume $\Psi_* \triangleq \min_{\boldsymbol{x}} \Psi(\boldsymbol{x})$ exists and is finite.

The bound is on the "primary" iterate $\boldsymbol{z}_N$, not on the secondary iterate $\boldsymbol{x}_N$, because if g is enforcing a constraint (like nonnegativity) then we want the final output to satisfy that constraint.

For the choice $\alpha_k = \frac{k-1}{k+2}$, Taylor et al. found numerically a tight bound for finite N and conjectured that it holds for all N [1, Table 1]:

$$\Psi(\boldsymbol{z}_N) - \Psi(\hat{\boldsymbol{x}}) \leq \frac{2L \left\| \boldsymbol{x}_0 - \hat{\boldsymbol{x}} \right\|_2^2}{N^2 + 5N + 6}.$$

A worst-case function is $f(x) = cx$ with $g(x) = 2c \max(-x, 0)$, for some c that depends on L.

- The $O(1/N^2)$ worst-case rate makes FPGM/FISTA much more attractive than PGM/ISTA.
  It requires more storage (see below), but this is a minor drawback compared to the acceleration benefit.
- FPGM is not guaranteed to decrease Ψ monotonically, unlike MM methods, even though the worst-case bound decreases monotonically.
- There are variations with a line search.
- Other improvements continue to surface, like "faster FISTA" [20] and methods for decreasing the gradient norm [21].
- Open questions include proving tight bounds analytically.
- Adaptive restart is helpful when f is locally strongly convex [22] [23].
- For a bound that also considers cases where $f(x)$ is **strongly convex**, see [24].
- Convergence of the iterates $\{\boldsymbol{z}_k\}$ is discussed in [25] [26].

Memory requirements

Implementing GD efficiently requires memory for two vectors of length N:
`g = grad(x)` (generally cannot be done in-place)
`x .-= step * g` (can do in-place!)

Example. Consider $f(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2 \implies \nabla f(\boldsymbol{x}) = \boldsymbol{A}'(\boldsymbol{x} - \boldsymbol{y})$. A memory efficient implementation uses the `mul!` in-place multiplication function in the `LinearAlgebra` package:

```
grad!(g, x) = mul!(g, A', A*x-y)
```

How many vectors of length N does PGM (5.10) need when $g(\boldsymbol{x}) = \beta \|\boldsymbol{x}\|_1$?
A: 1 B: 2 C: 3 D: 4 E: 5 **??**

Use above two steps of GD, then in-place soft threshold: `x .= soft.(x,beta)`

How many vectors of length N does FPGM (5.13) need when $g(\boldsymbol{x}) = \beta \|\boldsymbol{x}\|_1$?
A: 1 B: 2 C: 3 D: 4 E: 5 **??**

`g .= soft.(x,beta)` (reuse `g` to store $\boldsymbol{z}_k$)
`x .= g + alpha * (g - zold)` (*cf.* $\boldsymbol{x}_{k+1} = \boldsymbol{z}_k + \alpha(\boldsymbol{z}_k - \boldsymbol{z}_{k-1})$)
`zold .= g` (store $\boldsymbol{z}_k$ for next iteration where it will be $\boldsymbol{z}_{k-1}$)

Proximal optimized gradient method (POGM)

The **proximal optimized gradient method** (**POGM**) is an extension of the OGM [27] to the case of a **composite** cost function $f(\boldsymbol{x}) + g(\boldsymbol{x})$, where f is convex and smooth (L-Lipschitz gradient) and g is convex and prox friendly [1].

Initialize $\boldsymbol{w}_0 = \boldsymbol{z}_0 = \boldsymbol{x}_0, \theta_0 = \gamma_0 = 1$. Then `for` $k = 1 : N$:

$$\theta_k = \begin{cases} \frac{1}{2}\left(1 + \sqrt{4\theta_{k-1}^2 + 1}\right), & 2 \leq k < N \\ \frac{1}{2}\left(1 + \sqrt{8\theta_{k-1}^2 + 1}\right), & k = N \end{cases}$$

$$\gamma_k = \frac{1}{L} \frac{2\theta_{k-1} + \theta_k - 1}{\theta_k}$$

$$\boldsymbol{w}_k = \boldsymbol{x}_{k-1} - \frac{1}{L} \nabla f(\boldsymbol{x}_{k-1})$$

$$\boldsymbol{z}_k = \boldsymbol{w}_k + \underbrace{\frac{\theta_{k-1} - 1}{\theta_k}(\boldsymbol{w}_k - \boldsymbol{w}_{k-1})}_{\text{Nesterov}} + \underbrace{\frac{\theta_{k-1}}{\theta_k}(\boldsymbol{w}_k - \boldsymbol{x}_{k-1})}_{\text{OGM}} + \underbrace{\frac{\theta_{k-1} - 1}{L\gamma_{k-1}\theta_k}(\boldsymbol{z}_{k-1} - \boldsymbol{x}_{k-1})}_{\text{POGM}}$$

$$\boldsymbol{x}_k = \text{prox}_{\gamma_k g}(\boldsymbol{z}_k) = \arg\min_{\boldsymbol{x}} \frac{1}{2} \|\boldsymbol{x} - \boldsymbol{z}_k\|_2^2 + \gamma_k g(\boldsymbol{x})$$

- See [23] for an adaptive restart version.
- JULIA code is available in MIRT.jl in the function `pogm_restart`
- The convergence rate worst-case bound is about $2\times$ better than that of FPGM/FISTA
- The corresponding $\sqrt{2}$ decrease in number of iterations is often observed in practical applications like LASSO, matrix completion, and MRI [23, 28–30].
- Finding an analytical, tight, worst-case convergence bound is an open problem.
- Recall that OGM was $2\times$ better than Nesterov's FGM and it turned out to be optimal for smooth convex problems.
  Here POGM is $2\times$ better than FPGM/FISTA, so one can conjecture that POGM might be optimal or near optimal for composite convex problems.
  Proving that conjecture, or finding a provably optimal algorithm in the composite case is an open problem.
- Finding a line-search version that does not require knowing L is an open problem.
- A recent approach focuses on gradient norm decrease [31].
- Extension to the case where ∇f is $\boldsymbol{S}$-Lipschitz is possible using a change of variables.

Which sequence in the POGM algorithm is the one we should analyze for convergence bounds?
A: $\{\boldsymbol{w}_k\}$ B: $\{\boldsymbol{x}_k\}$ C: $\{\boldsymbol{z}_k\}$ D: $\{\theta_k\}$ E: $\{\gamma_k\}$ **??**

Despite the many open theoretical questions, the practical summary is simple:
use **POGM** (instead of FISTA or ISTA) to solve **composite** optimization problems!

Example. Consider the LASSO problem where $\Psi(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2 + b\|\boldsymbol{x}\|_1$ for $b > 0$.

The following code shows how to call POGM in MIRT for this optimization problem.

```
grad = x -> A' * (A*x - y) # gradient of smooth term
Lf = opnorm(A)^2 # best Lipschitz constant for gradient
soft = (x,t) -> sign(x) * max(abs(x) - t, 0) # soft threshold at t
prox1 = (z,c) -> soft.(z, c * b) # proximal operator for c*b*|x|_1
xh, out = pogm_restart(x0, x->0, grad, Lf ; g_prox=prox1, niter=niter)
```

Here the nonsmooth term is $g(\boldsymbol{x}) = b\|\boldsymbol{x}\|_1$.

A subtle point is that POGM needs

$$\text{prox}_{\gamma_k g}(\boldsymbol{v}) = \text{prox}_{\gamma_k b\|\cdot\|_1}(\boldsymbol{v}) = \text{soft}\,.(\boldsymbol{v}, \gamma_k b),$$

where the user provides the regularization parameter b but the POGM code provides the multiplier γ_k as the second argument "`c`" of the `g_prox` function.

Inexact computation of proximal operators (Read)

Many functions $g(\boldsymbol{x})$ are not prox friendly, *i.e.*, do not have a simple expression for $\operatorname{prox}_g(\cdot)$.

Important examples include:

- $\|\boldsymbol{T}\boldsymbol{x}\|_1$ when $\boldsymbol{T}$ is not diagonal or unitary,
  although cases where $\boldsymbol{T}$ is a **tight frame** are tractable using **projected FISTA** (**pFISTA**) [32],
- the characteristic function of an intersection of multiple convex sets,
- nuclear norm for large-scale problems (requires SVD),
- some group sparsity models,
- ...

Applying proximal methods like PGM / FPGM / POGM to such problems requires inner iterations to compute the proximal operator, and with a finite number of iterations it will be **inexact**.

Provided the inexactness errors diminish to zero fast enough in k, one can still guarantee $O(1/k)$ and $O(1/k^2)$ convergence rates of PGM and FPGM [33]. The "only" change is to the constant.

Recent work provides algorithms with optimal rates under inexactness [34].

Other proximal methods (Read)

The are many other proximal methods, especially in the recent literature.

- **proximal Newton algorithm** [35]
- Analysis of local (linear) convergence rates [36, 37]
- **inertial forward-backward** methods [37]
- **projected Nesterov's proximal-gradient** (**PNPG**) algorithm [38]
- Nesterov's **universal gradient** method that adapts to cost function properties (convex, strongly convex, etc.) [39]
- An inexact accelerated high-order proximal-point method [40] with $O(1/k^4)$ rate!

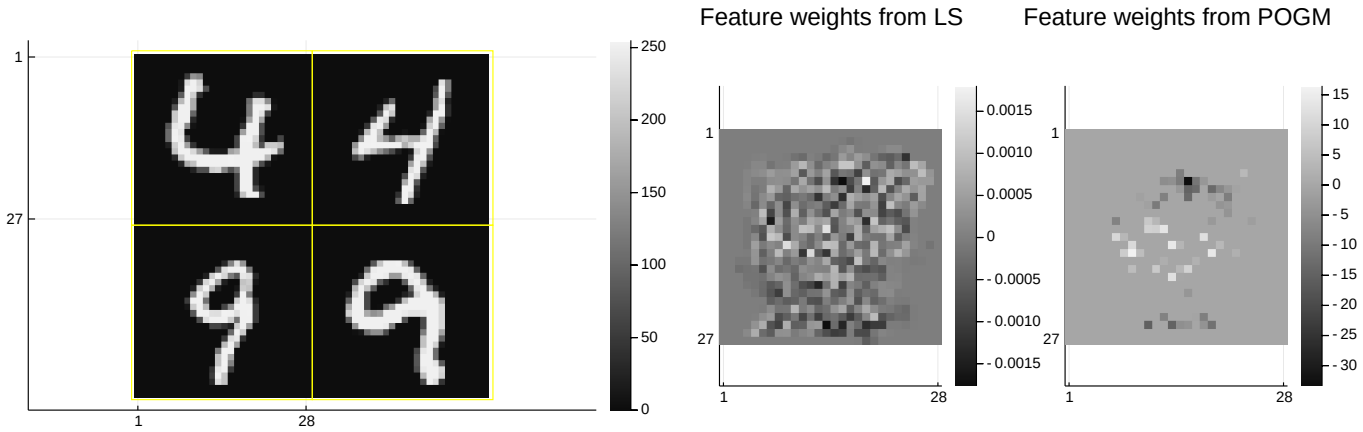
5.4 Examples

Machine learning: Binary classifier with 1-norm regularizer

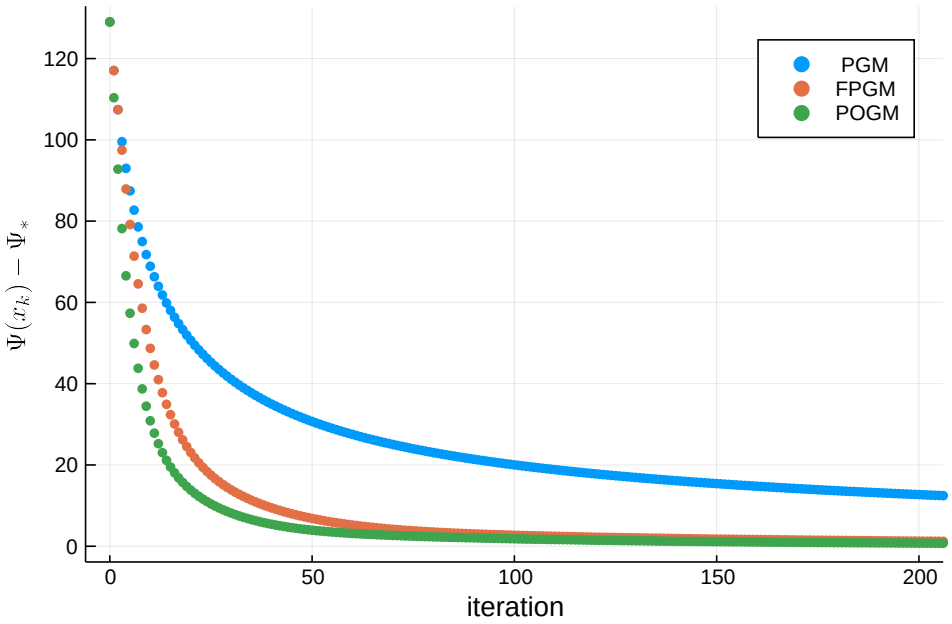
To encourage sparse features for classification using logistic loss:

$$\Psi(\boldsymbol{x}) = \mathbf{1}' h.(\boldsymbol{A}\boldsymbol{x}) + \beta \left\|\boldsymbol{x}\right\|_1, \quad h(z) = \log\left(1 + \mathrm{e}^{-z}\right).$$

Trained with 100 cases each of digits ‘4’ (label +1) and ‘9’ (label -1), and tested with 700 cases each.



Clearly the accelerated methods (using momentum) converge far faster than PGM, and POGM converges faster than FPGM.

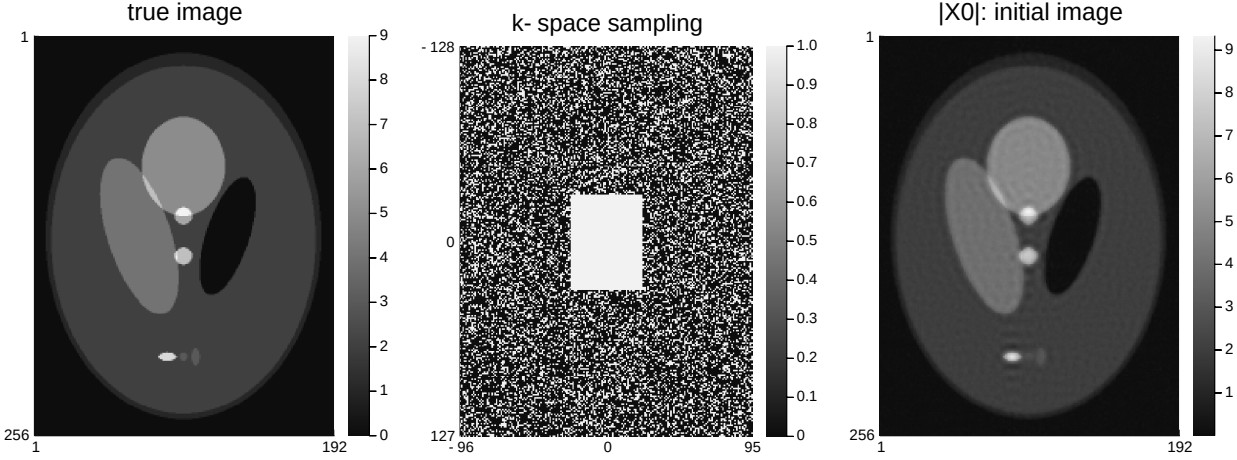


- Replacing the logistic loss with the Huber hinge loss is explored in HW .
- An alternative formulation is $\arg\min_{\boldsymbol{x}\,:\,\|\boldsymbol{x}\|_0 \le s} \mathbf{1}' h_.(\boldsymbol{A}\boldsymbol{x})$, for which a projected gradient-Newton method is discussed in [41].

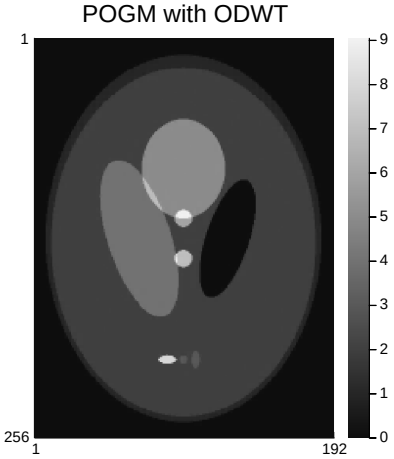
Example: MRI compressed sensing

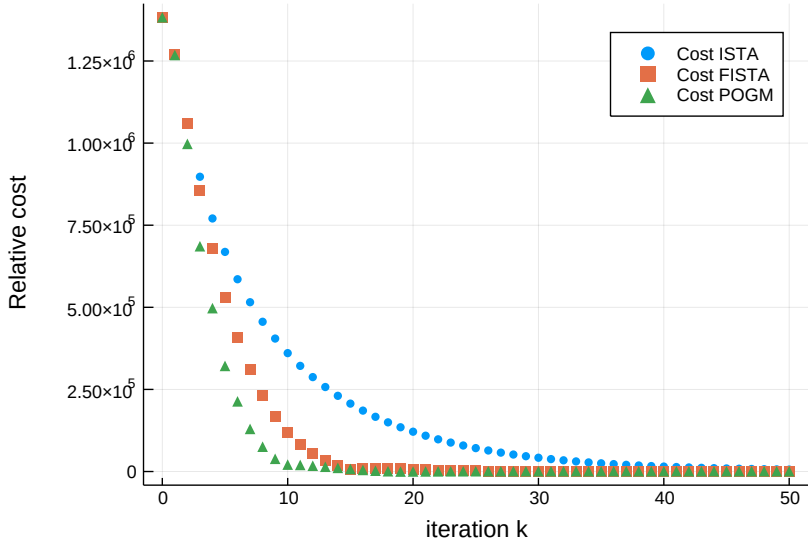
$$\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}} \frac{1}{2} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \beta \|\boldsymbol{T}\boldsymbol{x}\|_1$$

where $\boldsymbol{T}$ is **orthogonal discrete wavelet transform** (**ODWT**), $\boldsymbol{A}$ is wide matrix $(M < N)$ that models MRI physics and k-space sampling, and $\boldsymbol{y}$ is the raw "k-space" data measured by the MRI scanner.
This type of cost function is used in recent FDA-approved MRI scanners [42–44], so a real-world example!
Because $\boldsymbol{T}$ is unitary, this is a prox-friendly composite cost function, per (5.3).



As predicted by the worst-case convergence rate bounds, POGM converges about $\sqrt{2}\times$ faster than FPGM/-FISTA, which in turn is much faster than PGM/ISTA.





PGM:

$$\tilde{\boldsymbol{x}}_k = \boldsymbol{x}_k - \frac{1}{L}\boldsymbol{A}'(\boldsymbol{A}\boldsymbol{x}_k - \boldsymbol{y}), \quad \text{gradient step}$$

$$\boldsymbol{x}_{k+1} = \boldsymbol{T}' \,\text{soft}\,.(\boldsymbol{T}\tilde{\boldsymbol{x}}_k, \beta/L), \quad \text{denoising step, via wavelet coefficient soft thresholding (5.3).}$$

The Lipschitz constant in single-coil Cartesian MRI

The proximal methods shown in the preceding figure all require a value for the Lipschitz constant $L = \|\boldsymbol{A}\|_2^2$. In general this spectral norm can be expensive to compute, but for single-coil MRI with Cartesian k-space sampling, it is easy to evaluate. In this case, the form of the system matrix $\boldsymbol{A}$ is

$$\boldsymbol{A} = \underbrace{\boldsymbol{D}}_{\text{sampling}} \underbrace{\boldsymbol{Q}}_{\hookrightarrow \text{DFT}} \implies \boldsymbol{A}\boldsymbol{A}' =$$

where $\boldsymbol{Q}$ is an unitary DFT matrix, $\boldsymbol{D}\boldsymbol{D}' = \boldsymbol{I}_M$, and $\boldsymbol{D}'\boldsymbol{D}$ is a diagonal matrix with 1's where k-space is sampled and 0's where it is not (like in the middle picture on a previous page). Thus

$$L = \|\boldsymbol{A}\|_2^2 =$$

In practice, people often use the ordinary DFT instead of the unitary DFT, *i.e.*, $\boldsymbol{A} = \boldsymbol{D}\boldsymbol{F}$, where $\boldsymbol{F}^{-1} = \frac{1}{N}\boldsymbol{F}'$. For this model, what is $\|\boldsymbol{A}\|_2^2$?

A: $1/N^2$ B: N^2 C: 1 D: $1/N$ E: N **??**

Wavelets and Ingrid Daubechies (Read)

Probably the most important sparsifying transform ever developed for images is **wavelets**. They were pioneered by Stéphane Mallat [45] and Ingrid Daubechies [46].



Ingrid Daubechies was the first to invent **orthogonal wavelet bases** with **compact support**, and her 1988 paper on that topic [47] (that is 88 pages long) has been cited over 11,000 times on google scholar. In 1993, she became the first female mathematics professor to earn tenure at Princeton; about which she said: "In my view, this was more something they should have been ashamed of than a reason for celebration."
The **Daubechies wavelets** that she invented, especially her "D4" wavelet, are used widely in signal and image processing. Wikipedia says the **Daubechies wavelets** are "the most commonly used."

A key property of orthogonal wavelets is that one can compute an **orthogonal discrete wavelet transform** of a length-N signal in $O(N)$ time, making it even faster than the $O(N \log N)$ time of the FFT.

5.5 Proximal distance algorithms (Read)

The methods in this chapter have focused primarily on unconstrained optimization, though proximal methods are also well-suited to many constrained problems if the projection onto the constraint set $\mathcal{P}_{\mathcal{X}}(\cdot)$ is easy to evaluate.

Another way of handling constrained problems of the form

$$\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x} \in \mathcal{X}} \Psi(\boldsymbol{x})$$

is to use a regularizer with a large (or increasing with iteration) parameter:

$$\hat{\boldsymbol{x}} \approx \hat{\boldsymbol{x}}_\beta \triangleq \arg\min_{\boldsymbol{x}} \Psi(\boldsymbol{x}) + \beta \operatorname{dis}(\boldsymbol{x}, \mathcal{X}) = \arg\min_{\boldsymbol{x}} \Psi(\boldsymbol{x}) + \beta \left\| \boldsymbol{x} - \mathcal{P}_{\mathcal{X}}(\boldsymbol{x}) \right\|_2^2 .$$

The **proximal distance algorithm** of [48] provides a flexible family of methods for working with such optimization problems. A HW problem may explore such methods further.

Machine learning application: Sparse PCA

An **sparse PCA** example in [48] nicely illustrates the benefits of this method. If $\boldsymbol{X}$ is a $M \times N$ data matrix with each column having zero mean, then ordinary PCA is

$$\hat{\boldsymbol{U}} = \underset{\boldsymbol{U} \in \mathcal{V}_K(\mathbb{F}^N)}{\arg\max} \, \mathsf{trace}\{\boldsymbol{U}' \boldsymbol{\Pi} \boldsymbol{U}\} = \underset{\boldsymbol{U} \in \mathcal{V}_K(\mathbb{F}^N)}{\arg\min} \underbrace{- \mathsf{trace}\{\boldsymbol{U}' \boldsymbol{\Pi} \boldsymbol{U}\}}_{\text{concave}}, \quad \boldsymbol{\Pi} \triangleq \boldsymbol{X}' \boldsymbol{X}.$$

Define. The **Stiefel manifold** $\mathcal{V}_K(\mathbb{F}^N)$ is the set of $N \times K$ matrices having orthonormal columns

$$\mathcal{V}_K(\mathbb{F}^N) = \left\{ \boldsymbol{Q} \in \mathbb{F}^{N \times K} \, : \, \boldsymbol{Q}' \boldsymbol{Q} = \boldsymbol{I}_K \right\}.$$

The solution is the first K singular vectors of the $N \times N$ covariance matrix $\boldsymbol{\Pi}$, *i.e.*, the first K right singular vectors of $\boldsymbol{X}$ [49].

In some applications we also want $\boldsymbol{U}$ to be **sparse**, *e.g.*, each column of $\boldsymbol{U}$ has at most r nonzero elements:

$$\mathcal{S}_r \triangleq \left\{ \boldsymbol{U} \in \mathbb{F}^{N \times K} \, : \, \left\| U_{[:,k]} \right\|_0 \leq r, \; k = 1, \ldots, K \right\},$$

leading to the harder optimization problem with two non-convex sets and a non-convex cost function:

$$\hat{\boldsymbol{U}} = \underset{\boldsymbol{U} \in \mathcal{V}_K(\mathbb{F}^N) \cap \mathcal{S}_r}{\arg\min} \; - \mathsf{trace}\{\boldsymbol{U}' \boldsymbol{\Pi} \boldsymbol{U}\}.$$

It is easy to project onto $\mathcal{V}_K(\mathbb{F}^N)$ or $\mathcal{S}_r$ individually, but not jointly, so [48] considers

$$\hat{\boldsymbol{U}}_\beta = \underset{\boldsymbol{U} \in \mathcal{V}_K(\mathbb{F}^N)}{\arg\min} \; -\mathsf{trace}\{\boldsymbol{U}' \boldsymbol{\Pi} \boldsymbol{U}\} + \beta \|\boldsymbol{U} - \mathcal{P}_{\mathcal{S}_r}(\boldsymbol{U})\|_{\mathrm{F}}^2,$$

and develops a majorizer leading to a proximal update of the following form:

$$\boldsymbol{U}_{k+1} = \mathcal{P}_{\mathcal{V}_K(\mathbb{F}^N)}(\boldsymbol{\Pi} \boldsymbol{U}_k + \beta \mathcal{P}_{\mathcal{S}_r}(\boldsymbol{U}_k)).$$

Empirical results in [48] for breast cancer data, with $M = 19,672$ RNA measurements for $N = 89$ patients, demonstrate faster convergence than a previous method [50].

The above form is akin to the PGM, and invites the question of whether there is an accelerated version. The paper [48] reports that they found Nesterov-type acceleration is useful, citing [51], even for non-convex problems.

5.6 Summary

This chapter has described algorithms for minimizing composite cost functions.

Composite cost functions are rampant in modern signal processing and machine learning applications. Thus there are numerous applications of these methods, such as sparse regression, compressed sensing (with unitary sparsifying transformations), matrix completion and matrix sensing, binary classifier learning with sparsity regularization, etc.

Because of this wide applicability, developing proximal methods is an active research area. Some of the methods discussed, notably POGM, are quite recent.

Consideration of **non-convex** problems is especially interesting [51].

Bibliography

[1] A. B. Taylor, J. M. Hendrickx, and Francois Glineur. "Exact worst-case performance of first-order methods for composite convex optimization". In: *SIAM J. Optim.* 27.3 (Jan. 2017), 1283–313 (cit. on pp. 5.3, 5.15, 5.30, 5.31, 5.33).

[2] J. J. Moreau. "Proximité et dualité dans un espace hilbertien". In: *Bulletin de la Société Mathématique de France* 93 (1965), 273–99 (cit. on p. 5.4).

[3] N. Parikh and S. Boyd. "Proximal algorithms". In: *Found. Trends in Optimization* 1.3 (2013), 123–231 (cit. on p. 5.4).

[4] A. Beck. *First-order methods in optimization*. Soc. Indust. Appl. Math., 2017 (cit. on p. 5.4).

[5] S. Adly, Loic Bourdin, and F. Caubet. "On a decomposition formula for the proximal operator of the sum of two convex functions". In: *J. Convex Analysis* 26.3 (2019), 699–718 (cit. on p. 5.12).

[6] A. Ouorou. *Proximal bundle algorithms for nonsmooth convex optimization via fast gradient smooth methods*. 2020 (cit. on p. 5.17).

[7] A. Ouorou. *Fast proximal algorithms for nonsmooth convex optimization*. 2020 (cit. on p. 5.17).

[8] H. Attouch, Jerome Bolte, P. Redont, and A. Soubeyran. “Proximal alternating minimization and projection methods for nonconvex problems: an approach based on the Kurdyka-Łojasiewicz inequality”. In: *Mathematics of Operations Research* 35.2 (May 2010), 438–57 (cit. on p. 5.17).

[9] B. T. Polyak. *Introduction to optimization*. New York: Optimization Software Inc, 1987 (cit. on p. 5.22).

[10] Y. Drori and M. Teboulle. “Performance of first-order methods for smooth convex minimization: A novel approach”. In: *Mathematical Programming* 145.1-2 (June 2014), 451–82 (cit. on p. 5.23).

[11] D. H. Gutman and J. F. Pena. “Convergence rates of proximal gradient methods via the convex conjugate”. In: *SIAM J. Optim.* 29.1 (2019), 162–74 (cit. on p. 5.23).

[12] H. Karimi, J. Nutini, and M. Schmidt. “Linear convergence of gradient and proximal-gradient methods under the Polyak-Łojasiewicz condition”. In: *Machine Learning and Knowledge Discovery in Databases*. 2016, 795–811 (cit. on p. 5.24).

[13] A. B. Taylor, J. M. Hendrickx, and Francois Glineur. “Exact worst-case convergence rates of the proximal gradient method for composite convex minimization”. In: *J. Optim. Theory Appl.* 178.2 (Aug. 2018), 455–76 (cit. on pp. 5.26, 5.27).

[14] T. Blumensath and M. E. Davies. “Iterative thresholding for sparse approximations”. In: *J. Fourier Anal. and Appl.* 14.5 (2008), 629–54 (cit. on pp. 5.28, 5.29).

[15] T. Blumensath and M. Davies. “Iterative hard thresholding for compressed sensing”. In: *Applied and Computational Harmonic Analysis* 27.3 (Nov. 2009), 265–74 (cit. on p. 5.29).

[16] T. Blumensath. “Accelerated iterative hard thresholding”. In: *Signal Processing* 92.3 (Mar. 2012), 752–6 (cit. on p. 5.29).

[17] T. Blumensath and M. E. Davies. “Normalized iterative hard thresholding: guaranteed stability and performance”. In: *IEEE J. Sel. Top. Sig. Proc.* 4.2 (Apr. 2010), 298–309 (cit. on p. 5.29).

[18] A. Beck and Y. Eldar. “Sparsity constrained nonlinear optimization: optimality conditions and algorithms”. In: *SIAM J. Optim.* 23.3 (2013), 1480–509 (cit. on p. 5.29).

[19] H. Liu and R. F. Barber. *Between hard and soft thresholding: optimal iterative thresholding algorithms*. 2019 (cit. on p. 5.29).

[20] J. Liang and C-B. Schonlieb. *Improving FISTA: Faster, smarter and greedier*. 2018 (cit. on p. 5.31).

[21] D. Kim and J. A. Fessler. “Another look at the Fast Iterative Shrinkage/Thresholding Algorithm (FISTA)”. In: *SIAM J. Optim.* 28.1 (2018), 223–50 (cit. on p. 5.31).

[22] B. O’Donoghue and E. Candes. “Adaptive restart for accelerated gradient schemes”. In: *Found. Comp. Math.* 15.3 (June 2015), 715–32 (cit. on p. 5.31).

[23] D. Kim and J. A. Fessler. "Adaptive restart of the optimized gradient method for convex optimization". In: *J. Optim. Theory Appl.* 178.1 (July 2018), 240–63 (cit. on pp. 5.31, 5.34).

[24] L. Calatroni and A. Chambolle. "Backtracking strategies for accelerated descent methods with smooth composite objectives". In: *SIAM J. Optim.* 29.3 (Jan. 2019), 1772–98 (cit. on p. 5.31).

[25] H. Attouch, Z. Chbani, and H. Riahi. "Convergence rate of inertial proximal algorithms with general extrapolation and proximal coefficients". In: *Vietnam J. Math* 48.2 (2020), 247–76 (cit. on p. 5.31).

[26] H. Liu, T. Wang, and Z. Liu. *Convergence rate of inertial forward-backward algorithms based on the local error bound condition*. 2020 (cit. on p. 5.31).

[27] D. Kim and J. A. Fessler. "Optimized first-order methods for smooth convex minimization". In: *Mathematical Programming* 159.1 (Sept. 2016), 81–107 (cit. on p. 5.33).

[28] L. El Gueddari, C. Lazarus, Hanae Carrie, A. Vignaud, and P. Ciuciu. "Self-calibrating nonlinear reconstruction algorithms for variable density sampling and parallel reception MRI". In: *Proc. IEEE SAM*. 2018, 415–9 (cit. on p. 5.34).

[29] C. Y. Lin and J. A. Fessler. "Accelerated methods for low-rank plus sparse image reconstruction". In: *Proc. IEEE Intl. Symp. Biomed. Imag.* 2018, 48–51 (cit. on p. 5.34).

[30] C. Y. Lin and J. A. Fessler. "Efficient dynamic parallel MRI reconstruction for the low-rank plus sparse model". In: *IEEE Trans. Computational Imaging* 5.1 (Mar. 2019), 17–26 (cit. on p. 5.34).

[31] M. Ito and M. Fukuda. *Nearly optimal first-order methods for convex optimization under gradient norm measure: An adaptive regularization approach*. 2019 (cit. on p. 5.34).

[32] Y. Liu, Z. Zhan, J-F. Cai, D. Guo, Z. Chen, and X. Qu. "Projected iterative soft-thresholding algorithm for tight frames in compressed sensing magnetic resonance imaging". In: *IEEE Trans. Med. Imag.* 35.9 (Sept. 2016), 2130–40 (cit. on p. 5.36).

[33] M. Schmidt, N. L. Roux, and F. Bach. "Convergence rates of inexact proximal-gradient methods for convex optimization". In: *Neural Info. Proc. Sys.* 2011, 1458–66 (cit. on p. 5.36).

[34] M. Barre, A. Taylor, and F. Bach. *Principled analyses and design of first-order methods with inexact proximal operators*. 2020 (cit. on p. 5.36).

[35] X. Li, L. Yang, J. Ge, J. Haupt, T. Zhang, and T. Zhao. "On quadratic convergence of DC proximal Newton algorithm in nonconvex sparse learning". In: *Neural Info. Proc. Sys.* 2017, 2742–52 (cit. on p. 5.37).

[36] I. E-H. Yen, C-J. Hsieh, P. K. Ravikumar, and I. S. Dhillon. "Constant nullspace strong convexity and fast convergence of proximal methods under high-dimensional settings". In: *Neural Info. Proc. Sys.* 2014, 1008–16 (cit. on p. 5.37).

[37] J. Liang, J. Fadili, and G. Peyre. *Activity identification and local linear convergence of forward–backward-type methods*. 2018 (cit. on p. 5.37).

[38] R. Gu and A. Dogandzic. "Projected Nesterov's proximal-gradient algorithm for sparse signal recovery". In: *IEEE Trans. Sig. Proc.* 65.13 (July 2017), 3510–25 (cit. on p. 5.37).

[39] Y. Nesterov. "Universal gradient methods for convex optimization problems". In: *Mathematical Programming* 152.1-2 (Aug. 2015), 381–404 (cit. on p. 5.37).

[40] Y. Nesterov. *Inexact accelerated high-order proximal-point methods*. 2020 (cit. on p. 5.37).

[41] R. Wang, N. Xiu, and C. Zhang. "Greedy projected gradient-Newton method for sparse logistic regression". In: *IEEE Trans. Neural Net. Learn. Sys.* 31.2 (Feb. 2020), 527–38 (cit. on p. 5.39).

[42] FDA. *510k premarket notification of HyperSense (GE Medical Systems)*. 2017 (cit. on p. 5.40).

[43] FDA. *510k premarket notification of Compressed Sensing Cardiac Cine (Siemens)*. 2017 (cit. on p. 5.40).

[44] FDA. *510k premarket notification of Compressed SENSE*. 2018 (cit. on p. 5.40).

[45] S. G. Mallat. "A theory for multiresolution signal decomposition: the wavelet representation". In: *IEEE Trans. Patt. Anal. Mach. Int.* 11.7 (July 1989), 674–89 (cit. on p. 5.43).

[46] I. Daubechies. *Ten lectures on wavelets*. Philadelphia: Soc. Indust. Appl. Math., 1992 (cit. on p. 5.43).

[47] I. Daubechies. "Orthonormal bases of compactly supported wavelets". In: *Comm. Pure Appl. Math.* 41.7 (Oct. 1988), 909–96 (cit. on p. 5.43).

[48] K. L. Keys, H. Zhou, and K. Lange. "Proximal distance algorithms: Theory and practice". In: *J. Mach. Learning Res.* 20.66 (2019), 1–38 (cit. on pp. 5.44, 5.45, 5.46).

[49] K. Fan. "On a theorem of Weyl concerning eigenvalues of linear transformations I". In: *Proc. Natl. Acad. Sci.* 35.11 (Nov. 1949), 652–5 (cit. on p. 5.45).

[50] D. M. Witten, R. Tibshirani, and T. Hastie. "A penalized matrix decomposition, with applications to sparse principal components and canonical correlation analysis". In: *Biostatistics* 10.3 (July 2009), 515–534 (cit. on p. 5.46).

[51] S. Ghadimi and G. Lan. "Accelerated gradient methods for nonconvex nonlinear and stochastic programming". In: *Mathematical Programming* 156.1 (Mar. 2016), 59–99 (cit. on pp. 5.46, 5.47).