# CSCI 6960/4140: Homework 3

Assigned Tuesday, 2/24/2026. Due by 11:59pm ET 3/17/2026.

Objective of this assignment. This assignment develops two fundamental families of first-order methods for constrained optimization: projected gradient descent (PGD) and the Frank–Wolfe algorithm (also known as the conditional gradient method). Both are designed for minimizing a smooth function over a convex constraint set, but they handle the constraint very differently. PGD enforces feasibility at every step via a projection; Frank–Wolfe linearizes the objective and solves a linear minimization problem over the constraint set, which is often much cheaper. You will explore both on the low-rank matrix completion problem. Along the way you will:

- Understand projections onto norm balls from first principles via dual norms and normal cones.
- Implement PGD for both convex (nuclear norm ball) and nonconvex (fixed-rank) formulations, and understand concretely why nonconvexity is dangerous.
- Build Frank–Wolfe from scratch: linear minimization oracles, extreme points, and the update rule.
- Learn natural and robust stopping conditions used in practice for PGD and Frank–Wolfe.
- Compare PGD and Frank–Wolfe empirically and reason about their relative strengths.
- (CSCI 6960) Prove the $O(1/T)$ convergence of Frank–Wolfe.

Instructions. Create a Jupyter notebook, use Python 3+. Write documented, well-structured, readable code. Submit with outputs shown; the TA will not re-run your notebook to produce them. Label all plots: axes (with units if applicable), legends, titles; use differentiating styles for different data series; choose scales and limits appropriately.

Environment. I tested my solution on a machine with Python 3.8.8, NumPy 1.24.3, SciPy 1.6.2, scikit-learn 0.24.1, Matplotlib 3.3.4. Before running on the full dataset, test on a small synthetic case (e.g., $m = n = 20$, and a few dozen observed entries) to catch bugs quickly.

Academic Honesty and AI Usage. Your submitted work must reflect your own understanding and effort. Use of AI systems to generate or debug code, derivations, or analysis defeats the learning objectives of this assignment, undermines assessment integrity, is unfair to other students, and harms your learning.

Authorship Statement (Required). At the top of your notebook, include:

> "I completed this work without assistance from AI tools. All code, derivations, analysis, and discussion are my own."

Submissions omitting this statement will be treated as late.

Verification of Authorship. The instructor and TA may conduct oral spot-checks. Random, non-random, or triggered by discrepancies with in-class performance. Significant gaps in understanding may reduce the assignment grade or, in severe cases, constitute an academic honesty violation.




Figure 1: The power of the low-rank prior: Even when the observation set $\Omega$ is highly incomplete (left), the assumption that $\mathbf{X}^\star$ has low rank (e.g., $r = 1$) allows us to propagate information from a few entries to the entire matrix (right).

Setup: low-rank matrix completion (Problems 2 and 3). Let $\mathbf{X}^\star \in \mathbb{R}^{m\times n}$ be an unknown rank-$r$ matrix. We observe $\mathbf{y} = \mathcal{A}(\mathbf{X}^\star)$, where the linear operator $\mathcal{A} : \mathbb{R}^{m\times n} \to \mathbb{R}^{|\Omega|}$ extracts entries at index set $\Omega \subseteq [m] \times [n]$:

$$\mathcal{A}(\mathbf{X})_k = X_{i_k j_k}, \quad (i_k, j_k) \in \Omega.$$

The adjoint $\mathcal{A}^*$ places a measurement vector back at positions $\Omega$ with zeros elsewhere, satisfying $\langle \mathcal{A}(\mathbf{X}), \mathbf{y}\rangle = \langle \mathbf{X}, \mathcal{A}^*(\mathbf{y})\rangle$. Figure 1 illustrates the recovery task. We encode the low-rank prior in two ways:

(a) Nonconvex (rank-constrained):

$$\underset{\text{rank}(\mathbf{X})\le r}{\arg\min} \ \frac{1}{2}\|\mathcal{A}(\mathbf{X}) - \mathbf{y}\|_2^2. \tag{RC}$$

(b) Convex (nuclear norm ball): The nuclear norm $\|\mathbf{X}\|_* = \sum_i \sigma_i(\mathbf{X})$ is the $\ell_1$ norm of the singular values, and serves as a convex surrogate for rank (analogous to $\ell_1$ for sparsity):

$$\underset{\|\mathbf{X}\|_*\le\tau}{\arg\min} \ \frac{1}{2}\|\mathcal{A}(\mathbf{X}) - \mathbf{y}\|_2^2, \tag{NNB}$$

where $\tau = \|\mathbf{X}^\star\|_*$ is assumed known.

Data. Download `hw3recoverydata.npz` from `http://www.cs.rpi.edu/~gittea/old-site/teaching/spring2026/files/hw3recoverydata.npz`, which contains $\mathbf{y}$, $m$, $n$, $\Omega$, $r$, and $\tau$. The array `Omega` has shape $(|\Omega|, 2)$: row $k$ is the pair $(i_k, j_k)$. Load the data and implement the utility functions below.

Visual baseline (required). Before running any optimization algorithm, use `mat2im` to visualize $\mathcal{A}^*(\mathbf{y})$. This image—the observed entries placed back in their positions with zeros elsewhere—serves as your visual baseline: it is what the data "looks like" before any algorithm has been applied. You will compare all subsequent recovered images against this baseline to judge whether your algorithms are producing sensible output.

```
import numpy as np
import matplotlib.pyplot as plt

data = np.load('hw3recoverydata.npz', allow_pickle=True)
y, m, n = data['y'], int(data['m']), int(data['n'])
Omega, r, tau = data['Omega'], int(data['rank']), float(data['tau'])
```

```python
def A(X, Omega):
    """Apply A: extract entries of m x n matrix X at index pairs Omega.
    Returns 1-d array of length |Omega|."""
    ...

def Astar(y, Omega, m, n):
    """Apply A*: place 1-d array y at positions Omega in an m x n zero matrix.
    Returns m x n array."""
    ...

def mat2im(X):
    """Display float array as a grayscale image using matplotlib."""
    plt.imshow(X, cmap='gray', vmin=0, vmax=1)
    plt.axis('off')
    plt.show()
```

Note: Some observed values may lie slightly outside $[0, 1]$ (e.g., due to the ground truth image). This is expected; use `vmin=0, vmax=1` in `plt.imshow` (as shown above) to clip the display range consistently.

1. [30 pts] Projecting onto the $\ell_1$ ball.

   Projection onto the $\ell_1$ ball underlies the nuclear norm projection needed in Problem 2, since $\|\mathbf{X}\|_* = \|\boldsymbol{\sigma}(\mathbf{X})\|_1$, where $\boldsymbol{\sigma}(\mathbf{X})$ is the vector of singular values of $\mathbf{X}$. It also appears in LASSO via proximal gradient descent: the proximal operator of $\lambda\|\cdot\|_1$ is the soft-threshold, which is equivalent to projecting onto a scaled $\ell_1$ ball. In this problem you will derive the projection from first principles using convex analysis.

   We seek

   $$P_{\mathcal{B}_1(\tau)}(\mathbf{v}) = \operatorname*{arg\,min}_{\|\mathbf{x}\|_1 \le \tau} \tfrac{1}{2}\|\mathbf{x} - \mathbf{v}\|_2^2, \qquad \text{(Proj-}\ell_1\text{)}$$

   the Euclidean projection of $\mathbf{v} \in \mathbb{R}^d$ onto $\mathcal{B}_1(\tau) = \{\mathbf{x} : \|\mathbf{x}\|_1 \le \tau\}$.

   The key geometric insight (Figure 2) is that $\mathbf{v} - P_{\mathcal{B}_1(\tau)}(\mathbf{v})$ points in the direction of the outward normal cone of $\mathcal{B}_1(\tau)$ at the projection point. When $\mathbf{v}$ is already inside the ball, the projection is $\mathbf{v}$ itself (shown as $\mathbf{v}'$ above). The following exercises make this precise.

   (a) Dual norms.

   The dual norm of an arbitrary norm $\|\cdot\|$ on $\mathbb{R}^d$ is defined as

   $$\|\mathbf{g}\|_* = \sup_{\|\mathbf{x}\| \le 1} \langle \mathbf{g}, \mathbf{x} \rangle.$$

   It measures the largest value the linear function $\mathbf{x} \mapsto \langle \mathbf{g}, \mathbf{x} \rangle$ can take on the unit ball of $\|\cdot\|$. Geometrically: the unit ball $\{\mathbf{x} : \|\mathbf{x}\| \le 1\}$ extends farther in some directions than others, and $\|\mathbf{g}\|_*$ measures how far it reaches in the direction of $\mathbf{g}$. A large dual norm means the unit ball is "thin" in that direction; a small dual norm means the ball extends far.

   Dual norms appear ubiquitously; of particular interest to us is the fact that the subdifferential of a norm is expressed through the dual norm.

   (i) Show $\|\cdot\|_*$ is a norm. That is, it is: i) positively homogeneous, or $\|\alpha\mathbf{x}\|_* = |\alpha|\|\mathbf{x}\|_*$ for any scalar $\alpha$, ii) positive-definite, or $\|\mathbf{x}\|_* = 0$ if and only if $\mathbf{x} = 0$, and iii) it satisfies the triangle inequality $\|\mathbf{x} + \mathbf{y}\|_* \le \|\mathbf{x}\|_* + \|\mathbf{y}\|_*$.

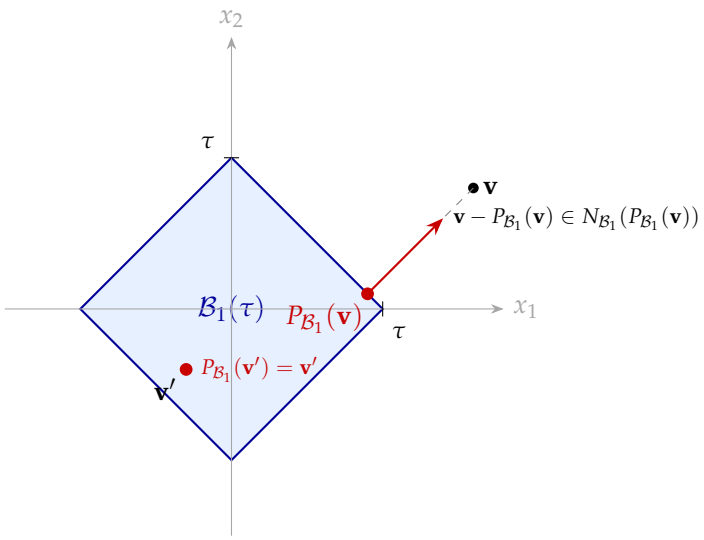



Figure 2: Euclidean Projection onto the $\ell_1$-ball. If $\mathbf{v}$ is outside the ball, it is mapped to the nearest point on the boundary; if inside, it remains unchanged.

(ii) Show the dual norm of $\|\cdot\|_1$ is $\|\cdot\|_\infty$. That is, show that: i) $\|\mathbf{x}\|_\infty \geq \langle \mathbf{y}, \mathbf{x} \rangle$ for any $\mathbf{y}$ in the unit $\ell_1$ ball, and ii) there exists a $\mathbf{y}$ in the unit $\ell_1$ ball for which $\|\mathbf{x}\|_\infty = \langle \mathbf{y}, \mathbf{x} \rangle$.

(b) Normal cone of a norm ball.

For a convex set $\mathcal{C}$ and $\mathbf{z} \in \mathcal{C}$, the normal cone is

$$N_{\mathcal{C}}(\mathbf{z}) = \{\mathbf{g} : \langle \mathbf{g}, \mathbf{x} - \mathbf{z} \rangle \leq 0 \ \forall \mathbf{x} \in \mathcal{C}\}.$$

These are directions making obtuse angles with every feasible direction from $\mathbf{z}$: they point "outward."

(i) Let $\mathcal{B} = \{\mathbf{x} : \|\mathbf{x}\| \leq \tau\}$ and $\|\mathbf{z}\| = \tau$. Show

$$N_{\mathcal{B}}(\mathbf{z}) = \{\lambda \mathbf{g} : \lambda \geq 0, \ \|\mathbf{g}\|_* \leq 1, \ \langle \mathbf{g}, \mathbf{z} \rangle = \|\mathbf{z}\|\}.$$

(For interior points $\|\mathbf{z}\| < \tau$, $N_{\mathcal{B}}(\mathbf{z}) = \{\mathbf{0}\}$; you need only prove the boundary case.) Hint: Use the definition of the dual norm to characterize when $\langle \mathbf{g}, \mathbf{x} \rangle \leq \langle \mathbf{g}, \mathbf{z} \rangle$ for all $\mathbf{x} \in \mathcal{B}$.

(ii) Specialize to $\|\cdot\|_1$. Describe explicitly what $N_{\mathcal{B}_1(\tau)}(\mathbf{z})$ is at a boundary point $\mathbf{z}$ with exactly $k$ nonzero entries. Work out the cases $k = 1$ and $k = d$ explicitly to build intuition.

(c) Characterizing the projection via Fermat's condition.

The optimality condition for (Proj-$\ell_1$) is: $\mathbf{z}^\star$ is optimal iff $\mathbf{v} - \mathbf{z}^\star \in N_{\mathcal{B}_1(\tau)}(\mathbf{z}^\star)$ when $\mathbf{v} \notin \mathcal{B}_1(\tau)$ (the residual points into the normal cone at the projection); if $\mathbf{v} \in \mathcal{B}_1(\tau)$ then $\mathbf{z}^\star = \mathbf{v}$.

(i) Translate this condition into explicit componentwise conditions on $\mathbf{z}^\star$ in terms of $\mathbf{v}$. Show that there exists $\theta \geq 0$ such that

$$z_i^\star = \text{sign}(v_i) \max(|v_i| - \theta, 0)$$

and $\sum_i |z_i^\star| = \tau$. This is the soft-threshold operator.

(ii) Show that the constraint $\sum_i |z_i^\star| = \tau$ determines $\theta$ uniquely, and derive the closed-form sorting algorithm: let $\mu_1 \geq \cdots \geq \mu_d \geq 0$ be the sorted absolute values of $\mathbf{v}$; then

$$\theta = \frac{1}{\rho}\left(\sum_{j=1}^{\rho} \mu_j - \tau\right), \quad \rho = \max\left\{k : \mu_k > \tfrac{1}{k}\left(\textstyle\sum_{j=1}^{k} \mu_j - \tau\right)\right\}.$$

(d) Implementation and testing.

Implement `projectL1` (Table 1): given a 1-d array `v` of length $d$ and positive float `tau`, returns the 1-d array $P_{\mathcal{B}_1(\tau)}(\mathtt{v})$ using the sorted-threshold formula from (c)(ii). Use no loops; use NumPy array operations only.

Table 1: Arguments for `projectL1(v, tau)`.

| Argument | Type | Description |
|---|---|---|
| `v` | 1-d `ndarray`, length $d$ | Vector to project |
| `tau` | `float` $> 0$ | Radius of the $\ell_1$ ball |

Returns: 1-d `ndarray` of length $d$, the projection $P_{\mathcal{B}_1(\tau)}(\mathtt{v})$.

Test your implementation:

(i) Project $\mathbf{v} = (3, 1, -2, 0.5)^T$ onto $\mathcal{B}_1(2)$. Verify $\|\mathbf{z}^\star\|_1 = 2$ and that the soft-threshold condition from (c)(i) is satisfied for each component.

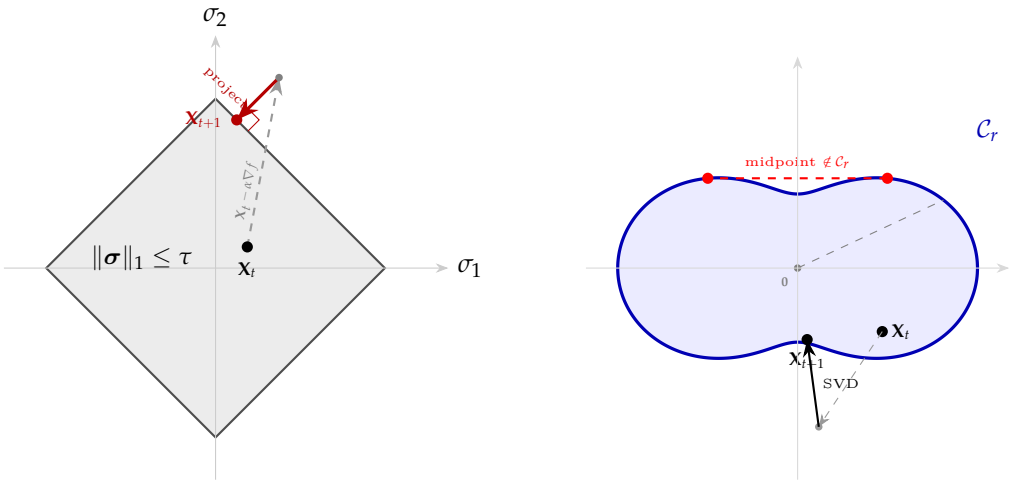
(ii) Let $d = 500$, draw $\mathbf{v} \sim \mathcal{N}(0, I)$, and project onto $\mathcal{B}_1(\tau)$ for $\tau \in \{1, 5, 10, 50\}$. For each $\tau$, produce a single plot with two curves: the absolute values $|v_i|$ sorted in decreasing order, and the absolute values $|z_i^\star|$ of the corresponding projection, plotted in the same sorting order (i.e., sort indices by $|v_i|$ and apply that permutation to both $|\mathbf{v}|$ and $|\mathbf{z}^\star|$). Report the number of nonzero entries (count $|z_i^\star| > 10^{-10}$) in $\mathbf{z}^\star$ for each $\tau$. What pattern do you observe as $\tau$ increases?

2. [40 pts] Projected gradient descent for matrix completion.

The smooth objective is $f(\mathbf{X}) = \frac{1}{2}\|\mathcal{A}(\mathbf{X}) - \mathbf{y}\|_2^2$ with gradient $\nabla f(\mathbf{X}) = \mathcal{A}^*(\mathcal{A}(\mathbf{X}) - \mathbf{y})$. The PGD update is

$$\mathbf{X}_{t+1} = P_{\mathcal{C}}(\mathbf{X}_t - \alpha \nabla f(\mathbf{X}_t)), \tag{PGD}$$

where $P_{\mathcal{C}}$ is Euclidean projection. The two constraint sets lead to qualitatively different behavior, illustrated in Figure 3.




Convex Projection
(Singular value space)

Nonconvex Rank Constraint
(schematic)

Figure 3: Left: The nuclear norm ball (in singular value space) is convex; PGD projects onto a convex set and enjoys standard convergence guarantees. Right (schematic): The set $\mathcal{C}_r = \{\mathbf{X} : \mathsf{rank}(\mathbf{X}) \leq r\}$ is nonconvex but path-connected: for any $\mathbf{X} \in \mathcal{C}_r$, the path $t \mapsto t\mathbf{X}$ stays in $\mathcal{C}_r$ (the set is star-shaped with respect to the origin). However, midpoints of two rank-$r$ matrices can have rank up to $2r$. This visualization is NOT accurate, just suggestive of the star-shaped nonconvex nature of $\mathcal{C}_r$.

(a) Why nonconvexity is dangerous.

The feasible set of (RC), $\mathcal{C}_r = \{\mathbf{X} \in \mathbb{R}^{m\times n} : \mathsf{rank}(\mathbf{X}) \leq r\}$, is not convex. Give a concrete $2 \times 2$ example: find two rank-1 matrices whose midpoint has rank 2. What guarantee, if any, does PGD provide when applied to (RC)? (State the strongest convergence guarantee you can justify for PGD on a nonconvex feasible set.)

(b) Projection onto rank-$r$ matrices.

By the Eckart–Young theorem, if $\mathbf{X} = \mathbf{U}\Sigma\mathbf{V}^T$, then $P_{\mathcal{C}_r}(\mathbf{X}) = \mathbf{U}_r\Sigma_r\mathbf{V}_r^T$, retaining only the top $r$ singular triplets. Use `scipy.sparse.linalg.svds` with `k=r` and `which='LM'` (largest magnitude; this is the default but worth specifying explicitly).

Implement the functions described in Tables 2 and 3.

Table 2: Arguments for `projectRank(X, r)`.

| Argument | Type | Description |
|---|---|---|
| `X` | $m \times n$ `ndarray` | Matrix to project |
| `r` | `int` | Target rank |

Returns: $m \times n$ `ndarray`, the rank-$r$ projection $P_{\mathcal{C}_r}(\mathtt{X})$.

Table 3: Arguments for `pgdRank(y, Omega, m, n, r, alpha, X_init, max_iter, tol)`.

| Argument | Type | Description |
|---|---|---|
| `y` | 1-d `ndarray`, length $\lvert\Omega\rvert$ | Observed measurements |
| `Omega` | $(\lvert\Omega\rvert, 2)$ `ndarray` | Observation index set |
| `m`, `n` | `int` | Matrix dimensions |
| `r` | `int` | Target rank |
| `alpha` | `float` | Step size (use $\alpha = 1$; see part (f)) |
| `X_init` | $m \times n$ `ndarray` | Initial iterate $\mathbf{X}_0$ |
| `max_iter` | `int` | Maximum iterations (use 1000) |
| `tol` | `float` | Tolerance $\epsilon$ for all three stopping criteria |

Returns: `(X, obj_hist, stop_info)` — $m \times n$ final iterate, 1-d array $[f(\mathbf{X}_0), \ldots, f(\mathbf{X}_T)]$, and string naming the stopping criterion triggered (or `'max_iter'`).

Stopping criteria for PGD. Stop when any of the following is satisfied, and also set a maximum iteration count[1]:

(i) Gradient mapping:

$$G_\alpha(\mathbf{X}_t) \;=\; \frac{1}{\alpha}\big\|\mathbf{X}_t - P_{\mathcal{C}}(\mathbf{X}_t - \alpha\nabla f(\mathbf{X}_t))\big\|_F \;\le\; \epsilon_g G_\alpha(\mathbf{X}_0)$$

In unconstrained optimization, stationarity means $\nabla f(\mathbf{X}^\star) = 0$. For constrained problems the gradient is generally nonzero at the optimum; the correct first-order condition is instead $\mathbf{X}^\star = P_{\mathcal{C}}(\mathbf{X}^\star - \alpha\nabla f(\mathbf{X}^\star))$, i.e., a projected gradient step leaves the optimal point fixed. The gradient mapping $G_\alpha$ measures deviation from this fixed-point condition and vanishes exactly at first-order stationary points—whether the constraint set is convex or not.

(ii) Iterate change: $\|\mathbf{X}_t - \mathbf{X}_{t-1}\|_F \le \epsilon_x(1 + \|\mathbf{X}_t\|_F)$. Catches stalling even when the gradient mapping is not small—common near saddle points in the nonconvex setting.

(iii) Objective change: $|f(\mathbf{X}_t) - f(\mathbf{X}_{t-1})| \le \epsilon_f(1 + |f(\mathbf{X}_t)|)$. Catches flat objective regions complementary to the above.

Use $\epsilon_g = \epsilon_x = \epsilon_f = \texttt{tol} = 10^{-6}$ (i.e., the single `tol` argument controls all three criteria). Report the criterion triggered and the iteration count in each experiment.

Implementation tip: Observe that $G_\alpha(\mathbf{X}_t) = \|\mathbf{X}_t - \mathbf{X}_{t+1}\|_F/\alpha$, since $\mathbf{X}_{t+1} = P_{\mathcal{C}}(\mathbf{X}_t - \alpha\nabla f(\mathbf{X}_t))$. You can compute the gradient mapping from the update you already performed, avoiding a second projection (and hence a second SVD) per iteration.

(c) Convexity of $f$ and the gradient formula.
Argue $f$ is convex and verify $\nabla f(\mathbf{X}) = \mathcal{A}^*(\mathcal{A}(\mathbf{X}) - \mathbf{y})$ using $\langle \mathcal{A}(\mathbf{X}), \mathcal{A}(\mathbf{X})\rangle = \langle \mathbf{X}, \mathcal{A}^*(\mathcal{A}(\mathbf{X}))\rangle$.

(d) Projection onto the nuclear norm ball.
Since $\|\mathbf{X}\|_* = \|\boldsymbol{\sigma}(\mathbf{X})\|_1$, if $\mathbf{X} = \mathbf{U}\,\mathrm{diag}(\boldsymbol{\sigma})\mathbf{V}^T$ then

$$P_{\mathcal{B}_*}(\mathbf{X}) = \mathbf{U}\;\mathrm{diag}\big(P_{\mathcal{B}_1(\tau)}(\boldsymbol{\sigma})\big)\;\mathbf{V}^T.$$

(i) Prove this formula. Hint: Write the projection problem as $\min_{\mathbf{Y}} \|\mathbf{Y} - \mathbf{X}\|_F^2$ subject to $\|\mathbf{Y}\|_* \le \tau$. Expand $\|\mathbf{Y} - \mathbf{X}\|_F^2 = \|\mathbf{Y}\|_F^2 + \|\mathbf{X}\|_F^2 - 2\langle \mathbf{Y}, \mathbf{X}\rangle$. Apply the von Neumann trace inequality $\langle \mathbf{A}, \mathbf{B}\rangle \le \sum_i \sigma_i(\mathbf{A})\sigma_i(\mathbf{B})$ to show the minimizer can be chosen to

[1]These are standard composite stopping criteria for first-order methods; see, e.g., Nocedal and Wright, Numerical Optimization, 2nd ed.

share the left and right singular vectors of $\mathbf{X}$. Once you fix the singular vectors, the problem reduces to projecting $\boldsymbol{\sigma}$ onto $\mathcal{B}_1(\tau)$. Recall: equality holds in the von Neumann inequality when $\mathbf{A}$ and $\mathbf{B}$ admit SVDs with the same left and right singular vectors (in matching order).

(ii) Using `projectL1` from Problem 1, implement `projectNuclear` (Table 4).

Table 4: Arguments for `projectNuclear(X, tau)`.

| Argument | Type | Description |
|---|---|---|
| `X` | $m \times n$ `ndarray` | Matrix to project |
| `tau` | `float` $> 0$ | Nuclear norm ball radius |

Returns: $m \times n$ `ndarray`, the projection $P_{\mathcal{B}_*(\tau)}(\texttt{X})$. Use a full SVD (`numpy.linalg.svd`) since all singular values must be soft-thresholded.

(e) PGD on the nuclear norm ball.

Implement `pgdNuclear` (Table 5).

Table 5: Arguments for `pgdNuclear(y, Omega, m, n, tau, alpha, X_init, max_iter, tol)`.

| Argument | Type | Description |
|---|---|---|
| `y` | 1-d `ndarray`, length $\lvert\Omega\rvert$ | Observed measurements |
| `Omega` | $(\lvert\Omega\rvert, 2)$ `ndarray` | Observation index set |
| `m`, `n` | `int` | Matrix dimensions |
| `tau` | `float` $> 0$ | Nuclear norm ball radius |
| `alpha` | `float` | Step size (use $\alpha = 1$; see part (f)) |
| `X_init` | $m \times n$ `ndarray` | Initial iterate $\mathbf{X}_0$ |
| `max_iter` | `int` | Maximum iterations (use 1000) |
| `tol` | `float` | Tolerance $\epsilon$ for all three stopping criteria |

Returns: `(X, obj_hist, stop_info)` — same semantics as `pgdRank`.

Runtime note. Each `pgdNuclear` iteration requires a full SVD of a $965 \times 800$ matrix, so expect roughly 10–20 minutes total on a modern laptop. `pgdRank` uses only a truncated SVD and is much faster (typically under 5 minutes). Debug both methods on a small synthetic case first (e.g., $m = n = 20$).

(f) Stepsize selection.

The smoothness constant of $f$ is $\beta = \|\mathcal{A}^*\mathcal{A}\|_{\text{op}}$ (operator norm induced by the Frobenius norm). Prove that $\beta = 1$ when each entry $(i, j)$ appears in $\Omega$ at most once.[2] Hint: What is $\mathcal{A}^*(\mathcal{A}(\mathbf{X}))$ entry by entry?

Use $\alpha = 1/\beta = 1$ for both PGD methods.

(g) Experiments. Solve (RC) using `pgdRank` and (NNB) using `pgdNuclear`. In both cases, initialize $\mathbf{X}_0 = \mathcal{A}^*(\mathbf{y})$. Note: the initialization $\mathcal{A}^*(\mathbf{y})$ need not be feasible (it is generally neither low-rank nor in the nuclear norm ball). For PGD this is harmless: the first gradient step followed by projection produces a feasible $\mathbf{X}_1$.

(i) Visualize each recovered image with `mat2im`. Compare to your $\mathcal{A}^*(\mathbf{y})$ baseline from the setup.

(ii) Report: stopping criterion, iteration count, relative measurement error $\|\mathcal{A}(\mathbf{X}_T) - \mathbf{y}\|_2^2 / \|\mathbf{y}\|_2^2$.

[2] In our data, $\Omega$ is a subset of $[m] \times [n]$, so no entry is repeated. If entries could repeat (as can happen, though rarely, with i.i.d. sampling with replacement), then $\mathcal{A}^*\mathcal{A}$ scales each entry by its multiplicity $c_{ij}$ and $\beta = \max_{(i,j)} c_{ij}$.

(iii) Plot $f(\mathbf{X}_t)$ vs. $t$ on a log scale, both methods on the same axes.

Compare image quality. Explain, in light of (a), why (RC) may give a different result than (NNB)—and why either could outperform the other. Your answer should address convexity and the role of initialization.

3. [30 pts] Frank–Wolfe for matrix completion.

   Frank–Wolfe minimizes a smooth function over a compact convex set by alternating between a linear minimization oracle (LMO) and a convex combination step:

$$\mathbf{s}_t = \arg\min_{\mathbf{S}\in\mathcal{C}} \langle \nabla f(\mathbf{X}_t), \mathbf{S}\rangle, \tag{LMO}$$

$$\mathbf{X}_{t+1} = (1-\gamma_t)\mathbf{X}_t + \gamma_t \mathbf{s}_t. \tag{FW}$$

   Because (FW) is a convex combination of two feasible points, feasibility is maintained automatically—no projection is needed. Figure 4 illustrates one step.

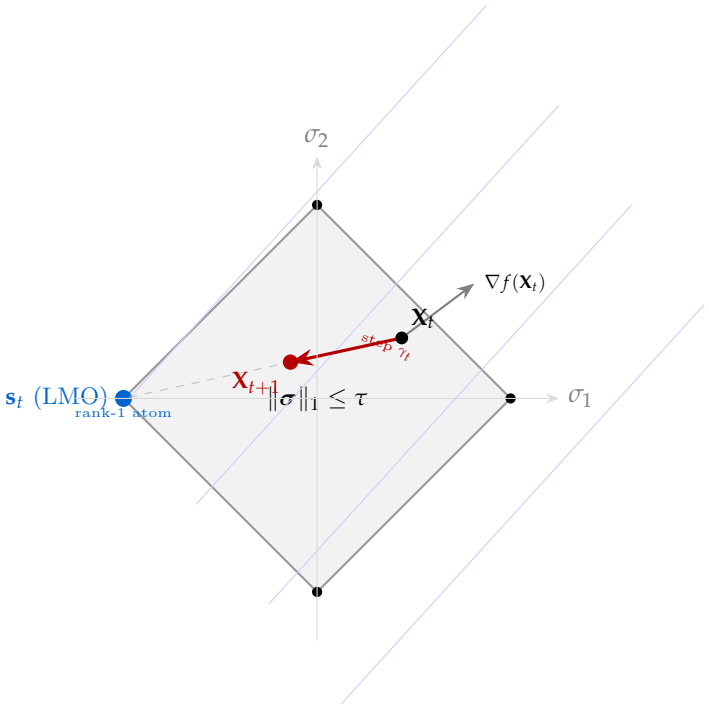



Figure 4: The Frank–Wolfe "Slide" (schematic, in singular value space): Instead of projecting, we find the extreme point $\mathbf{s}_t$ that is most "opposite" to the gradient direction (via the LMO). The next iterate $\mathbf{X}_{t+1}$ is simply a point on the segment connecting $\mathbf{X}_t$ and $\mathbf{s}_t$, ensuring it stays within the nuclear norm ball by construction.

   (a) Extreme points and the LMO.

   Definition 1. $\mathbf{x} \in \mathcal{C}$ is an extreme point of a convex set $\mathcal{C}$ if there are no $\mathbf{y} \neq \mathbf{z}$ in $\mathcal{C}$ and $\lambda \in (0,1)$ with $\mathbf{x} = \lambda\mathbf{y} + (1-\lambda)\mathbf{z}$. We write $\mathrm{ext}(\mathcal{C})$ for the set of extreme points.

   Intuitively, extreme points are "corners"—they cannot be expressed as midpoints of anything inside the set. For a polytope, extreme points are exactly the vertices. It is visually evident that the points in a polytope can be written as a convex combination of the vertices of the polytope.

   Cartheodory's theorem extends this intuition to general compact (closed and bounded) convex sets in $\mathbb{R}^d$: it tells us that every point in a compact convex set $C \subset \mathbb{R}^d$ can be written as a convex combination of at most $d+1$ extreme points of $C$.

   (i) Identify the extreme points of $\mathcal{B}_1(1) = \{\mathbf{x} \in \mathbb{R}^d : \|\mathbf{x}\|_1 \le 1\}$. Prove your answer.

(ii) Prove the following theorem.

Theorem 2 (Maximum of a convex function at an extreme point). Let $\mathcal{C} \subseteq \mathbb{R}^d$ be compact and convex, $h : \mathcal{C} \to \mathbb{R}$ convex. Then $\max_{\mathbf{x}\in\mathcal{C}} h(\mathbf{x})$ is attained at an extreme point.

Hint: Weierstrass guarantees the maximum is attained on a point $\mathbf{x}^\star$ in the set. Use Caratheodory's theorem on $\mathbf{x}^\star$.

(iii) We will use the following characterization of the extreme points of the nuclear norm ball:

Fact 3. The extreme points of $\mathcal{B}_*(\tau) = \{\mathbf{X} : \|\mathbf{X}\|_* \le \tau\}$ are exactly the rank-1 matrices $\tau\mathbf{u}\mathbf{v}^T$ with $\|\mathbf{u}\|_2 = \|\mathbf{v}\|_2 = 1$, together with the zero matrix.

(This follows from the fact that the nuclear norm ball is the closed convex hull of rank-1 matrices of nuclear norm $\tau$: $\mathcal{B}_*(\tau) = \overline{\text{conv}}\{\tau\mathbf{u}\mathbf{v}^T : \|\mathbf{u}\|_2 = \|\mathbf{v}\|_2 = 1\} \cup \{\mathbf{0}\}$. See, e.g., Jaggi, "Revisiting Frank–Wolfe," ICML 2013, Proposition 1.)

Applying Theorem 2 to $h(\mathbf{X}) = -\langle \mathbf{G}, \mathbf{X}\rangle$, derive the closed-form LMO solution

$$\mathbf{s}_t = -\tau\mathbf{u}_1\mathbf{v}_1^T,$$

where $\mathbf{u}_1, \mathbf{v}_1$ are the top left/right singular vectors of $\nabla f(\mathbf{X}_t)$. Hint: expand $\langle \mathbf{G}, \mathbf{u}\mathbf{v}^T\rangle$ via the SVD of $\mathbf{G}$.

(b) Implementation.

Stopping criteria for Frank–Wolfe. For Frank–Wolfe, use only the following two stopping criteria:

(i) Frank–Wolfe gap:

$$g_t = \langle \nabla f(\mathbf{X}_t),\, \mathbf{X}_t - \mathbf{s}_t\rangle \;\le\; \epsilon_g \cdot g_1,$$

where $g_1$ is the gap evaluated at $\mathbf{X}_1$ (the first feasible iterate, produced after the $\gamma_0 = 1$ step places us in the constraint set). Do not check this criterion until $t \ge 2$. The FW gap is the natural convergence certificate for Frank–Wolfe: it upper-bounds $f(\mathbf{X}_t) - f(\mathbf{X}^\star)$ (grad students will prove this in Problem 4).

(ii) Maximum iteration count.

Why not iterate-change or objective-change? The standard open-loop step size $\gamma_t = 2/(t+2)$ is diminishing, so both $\|\mathbf{X}_{t+1} - \mathbf{X}_t\|_F$ and $|f(\mathbf{X}_{t+1}) - f(\mathbf{X}_t)|$ shrink to zero regardless of whether the algorithm has converged—they measure the step size, not proximity to optimality. Using them as stopping criteria would cause premature termination. Use $\epsilon_g = 10^{-6}$.

Implement the functions described in Tables 6 and 7.

Table 6: Arguments for `lmoNuclear(G, tau)`.

| Argument | Type | Description |
|---|---|---|
| `G` | $m \times n$ `ndarray` | Gradient matrix $\nabla f(\mathbf{X}_t)$ |
| `tau` | `float` $> 0$ | Nuclear norm ball radius |

Returns: $m \times n$ `ndarray`, the LMO solution $-\tau\mathbf{u}_1\mathbf{v}_1^T$. Use `scipy.sparse.linalg.svds` with `k=1` and `which='LM'`.

Runtime note. Each Frank–Wolfe iteration requires only the top singular triplet (via `svds` with `k=1`), so it is much cheaper per iteration than `pgdNuclear`. Expect roughly 5–15 minutes for 2000 iterations on the full dataset.

Table 7: Arguments for `frankWolfe(y, Omega, m, n, tau, X_init, max_iter, tol)`.

| Argument | Type | Description |
|---|---|---|
| `y` | 1-d `ndarray`, length $\lvert\Omega\rvert$ | Observed measurements |
| `Omega` | $(\lvert\Omega\rvert, 2)$ `ndarray` | Observation index set |
| `m`, `n` | `int` | Matrix dimensions |
| `tau` | `float` $> 0$ | Nuclear norm ball radius |
| `X_init` | $m \times n$ `ndarray` | Initial iterate $\mathbf{X}_0$ |
| `max_iter` | `int` | Maximum iterations (use 2000) |
| `tol` | `float` | Tolerance $\epsilon_g$ for the FW gap criterion |

Returns: `(X, obj_hist, gap_hist, stop_info)` — $m \times n$ final iterate, 1-d arrays of $f(\mathbf{X}_t)$ and $g_t = \langle \nabla f(\mathbf{X}_t), \mathbf{X}_t - \mathbf{s}_t \rangle$ (both indexed from $t = 0$ to $t = T$, same length), and stopping criterion string.

Step size: Use the standard open-loop schedule $\gamma_t = 2/(t+2)$, where $t = 0, 1, 2, \ldots$ is the iteration index (so $\gamma_0 = 1$, $\gamma_1 = 2/3$, etc.).

(c) Experiments. Run `frankWolfe` with $\mathbf{X}_0 = \mathcal{A}^*(\mathbf{y})$. Note: Although $\mathbf{X}_0$ may lie outside $\mathcal{B}_*(\tau)$, the step size $\gamma_0 = 1$ gives $\mathbf{X}_1 = \mathbf{s}_0 \in \mathcal{B}_*(\tau)$, so all subsequent iterates are feasible.

(i) Visualize the recovered image with `mat2im`. Compare to your $\mathcal{A}^*(\mathbf{y})$ baseline.

(ii) Report: stopping criterion, iteration count, relative measurement error.

(iii) Two-panel figure: (left) $f(\mathbf{X}_t)$ and gap $g_t$ on a log scale vs. iteration; (right) $f(\mathbf{X}_t)$ for Frank–Wolfe and `pgdNuclear` on a log scale vs. iteration.

(d) Comparison.

(i) Which method converges faster empirically? Does the FW gap track the actual suboptimality $f(\mathbf{X}_t) - f(\mathbf{X}^\star)$ well? (Use the best objective value from `pgdNuclear` as a proxy for $f(\mathbf{X}^\star)$; note this proxy overestimates $f^\star$, so the bound $g_t \geq f(\mathbf{X}_t) - f^\star$ should certainly hold.)

(ii) What is the dominant per-iteration cost of each method? For very large $m, n$, which would you prefer and why?

(iii) Frank–Wolfe iterates are sums of at most $t$ rank-1 matrices. Why is this potentially very useful in practice?

(e) Rank control in Frank–Wolfe.

A notable drawback of Frank–Wolfe on the nuclear norm ball is that the rank of the iterates grows with each step: after $T$ iterations, $\mathbf{X}_T$ is a sum of at most $T$ rank-1 atoms, so $\mathsf{rank}(\mathbf{X}_T)$ can be as large as $\mathsf{min}(T, m, n)$, even when the ground truth $\mathbf{X}^\star$ has rank $r \ll T$.

(i) A natural heuristic is to round the Frank–Wolfe output to rank $r$ by computing the best rank-$r$ approximation (via truncated SVD) of the final iterate $\mathbf{X}_T$. Apply this to your Frank–Wolfe output: compute the rank-$r$ truncation $\hat{\mathbf{X}} = P_{\mathcal{C}_r}(\mathbf{X}_T)$, visualize the result with `mat2im` alongside the unrounded FW image and the PGD rank-constrained image for comparison, and report $\|\mathcal{A}(\hat{\mathbf{X}}) - \mathbf{y}\|_2^2 / \|\mathbf{y}\|_2^2$. How does this compare to the unrounded FW solution and to PGD on (RC)?

(ii) A more aggressive alternative is to truncate to rank $r$ at every iteration, i.e., replace the FW update with $\mathbf{X}_{t+1} = P_{\mathcal{C}_r}((1-\gamma_t)\mathbf{X}_t + \gamma_t \mathbf{s}_t)$. Would you expect this to preserve the convergence guarantees of Frank–Wolfe? Why or why not? (A brief but precise argument is sufficient.)

This problem is for CSCI 6960 students only.

4. [25 pts] Convergence of Frank–Wolfe.

   Assume $f : \mathbb{R}^{m\times n} \to \mathbb{R}$ is convex and $\beta$-smooth and $\mathcal{C}$ is compact convex with diameter $D = \max_{\mathbf{X},\mathbf{Y}\in\mathcal{C}} \|\mathbf{X}-\mathbf{Y}\|_F$.

   (a) From $\beta$-smoothness show $f(\mathbf{Y}) \le f(\mathbf{X}) + \langle \nabla f(\mathbf{X}), \mathbf{Y}-\mathbf{X}\rangle + \frac{\beta}{2}\|\mathbf{Y}-\mathbf{X}\|_F^2$. Substitute $\mathbf{X} = \mathbf{X}_t$, $\mathbf{Y} = (1-\gamma_t)\mathbf{X}_t + \gamma_t \mathbf{s}_t$ to get

   $$f(\mathbf{X}_{t+1}) \le f(\mathbf{X}_t) + \gamma_t \langle \nabla f(\mathbf{X}_t), \mathbf{s}_t - \mathbf{X}_t\rangle + \tfrac{\beta\gamma_t^2}{2} D^2.$$

   (b) Prove $g_t = \langle \nabla f(\mathbf{X}_t), \mathbf{X}_t - \mathbf{s}_t\rangle \ge f(\mathbf{X}_t) - f(\mathbf{X}^\star)$ for any minimizer $\mathbf{X}^\star \in \mathcal{C}$.

   (c) Using (a) and (b), prove that with $\gamma_t = 2/(t+2)$,

   $$f(\mathbf{X}_T) - f(\mathbf{X}^\star) \le \frac{2\beta D^2}{T+1}.$$

   Hint: Let $h_t = f(\mathbf{X}_t) - f(\mathbf{X}^\star)$; show (a) gives $h_{t+1} \le (1-\gamma_t)h_t + \frac{\beta\gamma_t^2}{2}D^2$, then prove by induction $h_t \le 2\beta D^2/(t+1)$; the bound $(t-1)\cdot 2/[t(t+1)] \le 2/(t+1)$ may be useful. Also, to prove the base case for $t = 0$, use the fact that $\mathbf{X}_1 = \mathbf{s}_0$ along with part (b).

   (d) What does this rate imply about the number of LMO calls needed to reach $\epsilon$-optimality? Both PGD and Frank–Wolfe achieve a sublinear $O(1/T)$ worst-case rate for general smooth convex problems. (Under additional assumptions such as strong convexity, PGD can achieve a linear rate $O(e^{-cT})$, while Frank–Wolfe generally cannot without algorithmic modifications.) What are the key practical differences between the methods? Address: per-iteration computational cost, convergence rate under additional assumptions, memory/storage of iterates, and structural properties of the iterates. When would you prefer Frank–Wolfe over PGD?