Practical works 3

Nhat-Nam Nguyen

1 Exercise 1

1.1 Problem Q_1

Definition of a Convex Function: A function $f : \mathbb{R}^d \to \mathbb{R}$ is convex if for all $x_1, x_2 \in \mathbb{R}^d$ and $\theta \in [0, 1]$,

$$f(\theta x_1 + (1-\theta)x_2) \leq \theta f(x_1) + (1-\theta) f(x_2).$$

Counterexample: Consider the cardinality function $\|\cdot\|_0$, which counts the number of nonzero components of a vector. Let $x_1 = (1, 0)$, $x_2 = (0, 1)$, and $\theta = 0.5$ in $\mathbb{R}^2$:

$$\theta x_1 + (1-\theta)x_2 = 0.5(1, 0) + 0.5(0, 1) = (0.5, 0.5).$$

The cardinality function evaluates to:

$$\|\theta x_1 + (1-\theta)x_2\|_0 = 2.$$

However, for $\|x_1\|_0 = 1$ and $\|x_2\|_0 = 1$, we have:

$$\theta\|x_1\|_0 + (1-\theta)\|x_2\|_0 = 0.5 \cdot 1 + 0.5 \cdot 1 = 1.$$

Since $2 > 1$, the cardinality function $\|\cdot\|_0$ is not convex.
Properties of a Norm: A function $\|\cdot\| : \mathbb{R}^d \to \mathbb{R}$ is a norm if it satisfies the following three properties:
1. **Non-negativity and Definiteness:**

$$\|x\| \geq 0, \quad \|x\| = 0 \iff x = 0.$$

The cardinality function satisfies this property because it is non-negative and equals 0 only for $x = 0$.
2. **Homogeneity:**

$$\|\alpha x\| = |\alpha|\|x\| \quad \text{for all } \alpha \in \mathbb{R}.$$

The cardinality function fails this property. For example, let $x = (1, 0)$ and $\alpha = 0.5$:

$$\|x\|_0 = 1, \quad \|\alpha x\|_0 = \|(0.5, 0)\|_0 = 1.$$

This violates the homogeneity condition because $\|\alpha x\|_0 \neq |\alpha|\|x\|_0$.
3. **Triangle Inequality:**

$$\|x + y\| \leq \|x\| + \|y\|.$$

The cardinality function fails this property. For example, let $x = (1, 0)$ and $y = (0, 1)$:

$$x + y = (1, 1), \quad \|x + y\|_0 = 2, \quad \|x\|_0 + \|y\|_0 = 1 + 1 = 2.$$

While this specific case satisfies equality, the triangle inequality does not hold in general for other vectors.
Since the cardinality function $\|\cdot\|_0$ is neither convex nor a norm, the minimization problem Q_1, which involves minimizing $\|x\|_0$, is not a convex minimization problem.

1.2 The convex envelop of the cardinality function

(a) Prove

For $y = (y_1, \ldots, y_d) \in \mathbb{R}^d$, $x = (x_1, \ldots, x_d) \in \mathbb{B}_\infty$. Since $x \in \mathbb{B}_\infty$, we have $|x_i| \leq 1$. Using this, we bound each term $y_i x_i$:

$$|y_i x_i| \leq |y_i| \quad \Rightarrow \quad y_i x_i - 1 \leq |y_i| - 1$$

$$\langle y, x \rangle - \|x\|_0 = \sum_{i=1}^{d} y_i x_i - \|x\|_0 = \sum_{x_i \neq 0} y_i x_i - \|x\|_0 = \sum_{x_i \neq 0} (y_i x_i - 1) \leq \sum_{x_i \neq 0} (|y_i| - 1).$$

Since $\sum_{x_i \neq 0}(|y_i| - 1) \leq \sum_{i=1}^{d}(|y_i| - 1)^+$, we conclude:

$$\langle y, x \rangle - \|x\|_0 \leq \sum_{i=1}^{d} (|y_i| - 1)^+.$$

(b) Deduce that equality

To compute the conjugate $g^*(y)$, we start from the definition of $g(x) = \|x\|_0 + \nu_{\mathbb{B}_\infty}(x)$, where:

$$g(x) = \begin{cases} \|x\|_0, & \text{if } x \in \mathbb{B}_\infty, \\ +\infty, & \text{if } x \notin \mathbb{B}_\infty. \end{cases}$$

The conjugate $g^*(y)$ is defined as:

$$g^*(y) = \sup_{x \in \mathbb{R}^d} \left(\langle y, x \rangle - g(x) \right).$$

Substituting $g(x)$, the domain reduces to $x \in \mathbb{B}_\infty$, so:

$$g^*(y) = \sup_{x \in \mathbb{B}_\infty} \left(\langle y, x \rangle - \|x\|_0 \right).$$

Because we have

$$\langle y, x \rangle - \|x\|_0 \leq \sum_{i=1}^{d} (|y_i| - 1)^+.$$

Therefore:

$$g^*(y) = \sum_{i=1}^{d} (|y_i| - 1)^+.$$

(c) Prove

Let $x = (x_1, \ldots, x_d) \in \overline{\mathbb{B}}_\infty$ and $y = (y_1, \ldots, y_d) \in \mathbb{R}^d$. We want to prove that:

$$\langle x, y \rangle - g^*(y) \leq \sum_{|y_i| < 1} |x_i| + \sum_{|y_i| \geq 1} (|y_i|(|x_i| - 1) + 1) \leq |x|_1.$$

From part (b), we have:

$$g^*(y) = \sum_{i=1}^{d} (|y_i| - 1)^+.$$

For $|y_i| < 1$, since $|x_i| \leq 1$, we have:

$$|y_i x_i| \leq |x_i|.$$

Substituting this into the left-hand side of the inequality:

$$\langle x, y\rangle - g^*(y) = \sum_{i=1}^{d} x_i y_i - \sum_{i=1}^{d} (|y_i| - 1)^+ \tag{1}$$

$$= \sum_{|y_i|<1} x_i y_i + \sum_{|y_i|\geq 1} x_i y_i - \sum_{|y_i|\geq 1} (|y_i| - 1) \tag{2}$$

$$= \sum_{|y_i|<1} x_i y_i + \sum_{|y_i|\geq 1} (x_i y_i - |y_i| + 1) \tag{3}$$

$$\leq \sum_{|y_i|<1} |x_i| + \sum_{|y_i|\geq 1} (|y_i||x_i| - |y_i| + 1) \tag{4}$$

$$= \sum_{|y_i|<1} |x_i| + \sum_{|y_i|\geq 1} (|y_i|(|x_i| - 1) + 1). \tag{5}$$

For $|y_i| \geq 1$, the term $|y_i|(|x_i| - 1) + 1$ is nonnegative, and $|x_i| \leq 1$ ensures:

$$|y_i|(|x_i| - 1) + 1 \leq |x_i|.$$

Combining both sums:

$$\begin{aligned}\sum_{|y_i|<1} |x_i| + \sum_{|y_i|\geq 1} (|y_i|(|x_i| - 1) + 1) &\leq \sum_{|y_i|<1} |x_i| + \sum_{|y_i|\geq 1} |x_i| \\ &= \sum_{i=1}^{d} |x_i| \\ &= \|x\|_1\end{aligned}$$

Thus, we have proved the required inequality:

$$\langle x, y\rangle - g^*(y) \leq \sum_{|y_i|<1} |x_i| + \sum_{|y_i|\geq 1} (|y_i|(|x_i| - 1) + 1) \leq \|x\|_1.$$

(d) Deduce that

From part (c), we have:

$$\langle x, y\rangle - g^*(y) \leq \sum_{|y_i|<1} |x_i| + \sum_{|y_i|\geq 1} (|y_i|(|x_i| - 1) + 1) \leq \|x\|_1,$$

for all $x \in \overline{\mathbb{B}}_\infty$ and $y \in \mathbb{R}^d$.
By definition, the biconjugate $(g^*)^*(x)$ is:

$$(g^*)^*(x) = \sup_{y\in\mathbb{R}^d} \left(\langle x, y\rangle - g^*(y)\right).$$

From part (c), we know that:

$$\sup_{y\in\mathbb{R}^d} \left(\langle x, y\rangle - g^*(y)\right) \leq \|x\|_1.$$

To prove equality, we construct a specific y that achieves the upper bound. Choose $y = \operatorname{sign}(x)$, where:

$$\operatorname{sign}(x_i) = \begin{cases} 1, & \text{if } x_i > 0, \\ -1, & \text{if } x_i < 0, \\ 0, & \text{if } x_i = 0. \end{cases}$$

With this choice of y, we have:

$$\begin{aligned}\langle x, y\rangle - g^*(y) &= \sum_{i=1}^{d} x_i \cdot \operatorname{sign}(x_i) - \sum_{i=1}^{d} (|\operatorname{sign}(x_i)| - 1)^+ \\ &= \sum_{i=1}^{d} |x_i| - 0 \\ &= \|x\|_1.\end{aligned}$$

Therefore, we have:

$$(g^*)^*(x) = \sup_{y \in \mathbb{R}^d} (\langle x, y \rangle - g^*(y)) = \|x\|_1,$$

for all $x \in \overline{\mathbb{B}}_\infty$.

(e) Prove

We aim to show:

$$\forall x \in \mathbb{R}^d \setminus \overline{\mathbb{B}}_\infty, \quad (g^*)^*(x) = +\infty.$$

We have

$$(g^*)^*(x) = \sup_{y \in \mathbb{R}^d} \left(\langle x, y \rangle - g^*(y)\right).$$

From part (b):

$$g^*(y) = \sum_{i=1}^{d} (|y_i| - 1)^+.$$

If $x \notin \overline{\mathbb{B}}_\infty$, there exists at least one i such that $|x_i| > 1$. For this i, choosing $|y_i| \to \infty$ makes:

$$\langle x, y \rangle = \sum_{i=1}^{d} x_i y_i \quad \text{arbitrarily large.}$$

Meanwhile, $g^*(y)$ grows slower since:

$$g^*(y) = \sum_{i=1}^{d} (|y_i| - 1)^+.$$

Thus, for any $M > 0$, we can find y such that:

$$\langle x, y \rangle - g^*(y) \geq M.$$

Therefore we have:

$$\sup_{y \in \mathbb{R}^d} (\langle x, y \rangle - g^*(y)) = +\infty, \quad \forall x \notin \overline{\mathbb{B}}_\infty.$$

$$\implies (g^*)^*(x) = +\infty.$$

(f) Deduce that

From parts (d) and (e), we consider two cases:
Case 1: If $x \in \overline{\mathbb{B}}_\infty$, from part (d), we have:

$$(g^*)^*(x) = \|x\|_1.$$

Case 2: If $x \notin \overline{\mathbb{B}}_\infty$, from part (e), we have:

$$(g^*)^*(x) = +\infty.$$

Combining both cases, the result is:

$$(g^*)^*(x) = \begin{cases} \|x\|_1, & \text{if } x \in \overline{\mathbb{B}}_\infty, \\ +\infty, & \text{if } x \notin \overline{\mathbb{B}}_\infty. \end{cases} \iff (g^*)^* = \|\cdot\|_1 + \iota_{\overline{\mathrm{B}}_\infty}$$

(g) Deduce that

From the **biconjugate theorem**:

$$f^{**} = \text{aff}(f),$$

where $\text{aff}(f)$ is the affine hull of f.
For $g(x) = \|\cdot\|_0 + \iota_{\mathbb{B}_\infty}(x)$, its conjugate and biconjugate are:

$$g^*(y) = \sum_{i=1}^{d}(|y_i| - 1)^+, \quad g^{**}(x) = \|x\|_1 + \iota_{\mathrm{B}_\infty}.$$

By the **biconjugate theorem**:

$$g^{**}(x) = \text{aff}(g(x)).$$

Hence, $\|\cdot\|_1$ is the affine hull of $\|\cdot\|_0$ over $\mathbb{B}_\infty$:

$$\|\cdot\|_1 + \iota_{\mathrm{B}_\infty} = \text{aff}\left(\|\cdot\|_0 + \iota_{\mathrm{B}_\infty}\right)$$

(h) Deduce that

To show that $\frac{1}{M}\|\cdot\|_1$ is the affine hull of $\|\cdot\|_0$ over the ball $\overline{\mathbb{B}}_M := M\overline{\mathbb{B}}_\infty$ for $M > 0$: We have scale the ball $\overline{\mathbb{B}}_\infty$ by M:

$$\overline{\mathbb{B}}_M = M\overline{\mathbb{B}}_\infty = \{x \in \mathbb{R}^d : \|x\|_\infty \leq M\}.$$

The scaled function:

$$g(x) = \|\cdot\|_0 + \iota_{\overline{\mathbb{B}}_M}(x),$$

where $\iota_{\overline{\mathbb{B}}_M}(x)$ is the indicator function of $\overline{\mathbb{B}}_M$. From the biconjugate theorem:

$$g^{**}(x) = \frac{1}{M}\|x\|_1, \quad \forall x \in \overline{\mathbb{B}}_M.$$

Thus, by scaling:

$$\frac{1}{M}\|\cdot\|_1 \text{ is the affine hull of } \|\cdot\|_0 \text{ over } \overline{\mathbb{B}}_M.$$

1.3 Several Approximations and Simplifications of Problem (Q1)

Consider the relationship between solutions of problems (Q3) and (Q4):

$$\text{(Q3)} \quad \min_x \|x\|_1 \quad \text{s.t.} \quad \|Ax - b\|_2 \leq \theta,$$

$$\text{(Q4)} \quad \min_x \mu\|x\|_1 + \frac{1}{2}\|Ax - b\|_2^2.$$

Proposition 1 *If x^* solves (Q4), then x^* also solves (Q3) with $\theta = \|Ax^* - b\|_2$.*

Proof. Let x^* solve (Q4) and define $\theta^* = \|Ax^* - b\|_2$. Since x^* minimizes the objective in (Q4), it satisfies:

$$\mu\|x^*\|_1 + \frac{1}{2}\|Ax^* - b\|_2^2 \leq \mu\|x\|_1 + \frac{1}{2}\|Ax - b\|_2^2, \quad \forall x \in \mathbb{R}^d.$$

In particular, for any x such that $\|Ax - b\|_2 \leq \theta^*$, we have:

$$\|x^*\|_1 \leq \|x\|_1.$$

Thus, x^* minimizes $\|x\|_1$ among all x feasible for (Q3) with $\theta = \theta^*$. Therefore, x^* solves (Q3). □
The reverse direction requires strong duality:

Proposition 2 *Suppose strong duality holds for (Q3). If x^* solves (Q3), then there exists $\mu \geq 0$ such that x^* solves (Q4).*

Proof. By strong duality, (Q3) is equivalent to its Lagrangian dual:

$$\max_{\mu \geq 0} \min_x \|x\|_1 + \mu\left(\frac{1}{2}\|Ax - b\|_2^2 - \frac{\theta^2}{2}\right).$$

If x^* solves (Q3), let $\mu^* \geq 0$ be a dual optimal solution. Rearranging terms shows that x^* solves (Q4) with $\mu = \mu^*$. □

1.4 Setup of the ADMM Algorithm

To apply the ADMM algorithm, Problem Q_4 is rewritten as:

$$(\mathrm{Q}_5) \qquad \min_{(x,y)\in\mathbb{R}^d\times\mathbb{R}^d} \mu\|y\|_1 + \frac{1}{2}\|Ax-b\|_2^2, \quad \text{subject to } x-y=0.$$

(a) Augmented Lagrangian

The augmented Lagrangian for Q_5 is given by:

$$L^\lambda(x,y,z) = \mu\|y\|_1 + \frac{1}{2}\|Ax-b\|_2^2 + z^\top(x-y) + \frac{\lambda}{2}\|x-y\|_2^2,$$

where: - $z \in \mathbb{R}^d$ is the dual variable (Lagrange multiplier), - $\lambda > 0$ is the penalty parameter.
ADMM solves Q_5 by alternating updates for x, y, and z in three steps:
1. x-update:

$$x^{k+1} = \arg\min_x \left(\frac{1}{2}\|Ax-b\|_2^2 + \frac{\lambda}{2}\|x-y^k\|_2^2 + z^{k\top}(x-y^k)\right).$$

2. y-update:

$$y^{k+1} = \arg\min_y \left(\mu\|y\|_1 + z^{k\top}(x^{k+1}-y) + \frac{\lambda}{2}\|x^{k+1}-y\|_2^2\right).$$

3. Dual variable z-update:

$$z^{k+1} = z^k + \lambda(x^{k+1}-y^{k+1}).$$

(b) Prove the x-update formula

The x-update in ADMM solves the subproblem:

$$x^{k+1} = \arg\min_x \left(\frac{1}{2}\|Ax-b\|_2^2 + z^\top(x-y) + \frac{\lambda}{2}\|x-y\|_2^2\right) = \arg\min_x H(x)$$

easy to see the H(x) is a convex function. So we have:

$$\frac{\partial}{\partial x}\left(\frac{1}{2}x^\top A^\top Ax + \frac{\lambda}{2}x^\top x - x^\top(A^\top b + \lambda y^k - z^k))\right) = 0.$$

This leads to:

$$(A^\top A + \lambda I)x^{k+1} = A^\top b + \lambda(y^k - \frac{z^k}{\lambda}).$$

Solve for x^{k+1}:

$$x^{k+1} = (\lambda Id + A^\top A)^{-1}(A^\top b + \lambda y^k - z^k).$$

(c) Prove the y-update formula

The y-update in ADMM solves:

$$\begin{aligned}
y^{k+1} &= \arg\min_y \left(\mu\|y\|_1 + z^{k\top}(x^{k+1}-y) + \frac{\lambda}{2}\|x^{k+1}-y\|_2^2\right) \\
&\iff y^{k+1} = \arg\min_y \left(\mu\|y\|_1 + z^{k\top}(x^{k+1}-y) + \frac{\lambda}{2}\|x^{k+1}-y\|_2^2\right) \\
&\iff y^{k+1} = \arg\min_y \left(\mu\|y\|_1 + \frac{\lambda}{2}\|x^{k+1}-y\|_2^2 + z^{k\top}(x^{k+1}-y) + \frac{1}{2\lambda}(z^k)^2\right) \\
&\iff y^{k+1} = \arg\min_y \left(\mu\|y\|_1 + \frac{\lambda}{2}\|y-(x^{k+1}+\frac{z^k}{\lambda})\|_2^2\right) \\
&\iff y^{k+1} = \mathrm{prox}_{\frac{\mu}{\lambda}\|\cdot\|_1}\left(x^{k+1}+\frac{z^k}{\lambda}\right).
\end{aligned}$$

1.5 Matlab code

The following MATLAB implementation solves Problem (Q_5) using the ADMM algorithm. We first demonstrate with a simple test case using a 3×3 matrix.

```
% Test Data
A = [1, 2, 3;
     4, 5, 6;
     7, 8, 10];     % A 3x3 matrix
b = [1; 2; 3];      % Target vector (dimension 3x1)
mu = 0.5;           % Regularization parameter
lambda = 1.0;       % Augmented Lagrangian parameter
tol = 1e-6;         % Convergence tolerance
max_iter = 1000;    % Maximum number of iterations

% Initialization
[d, ~] = size(A);         % Dimension of x
x = zeros(d, 1);          % Initialize x
y = zeros(d, 1);          % Initialize y
z = zeros(d, 1);          % Initialize dual variable z
obj_values = [];          % Store objective function values
x_prev = x;               % To track previous x
y_prev = y;               % To track previous y

% ADMM iterations
for k = 1:max_iter
    % Step 1: x-update
    x = (lambda * eye(d) + A' * A) \ (A' * b + lambda * y - z);

    % Step 2: y-update (proximal operator of l1-norm)
    v = x + (z / lambda);
    y = sign(v) .* max(abs(v) - mu / lambda, 0);

    % Step 3: z-update (dual variable update)
    z = z + lambda * (x - y);

    % Track objective value
    obj = mu * norm(y, 1) + 0.5 * norm(A * x - b, 2)^2;
    obj_values = [obj_values; obj];

    % Check for convergence
    if norm(x - y, 2) < tol && norm(y - y_prev, 2) < tol
        fprintf('Converged at iteration %d\n', k);
        break;
    end

    % Update previous x and y
    x_prev = x;
    y_prev = y;
end

% Display results
fprintf('Optimal x:\n');
disp(x);
fprintf('Optimal y:\n');
disp(y);
fprintf('Optimal dual variable z:\n');
disp(z);
```

Func 1: ADMM Implementation for ℓ_1-Minimization

This implementation efficiently solves the ℓ_1-minimization problem while maintaining numerical stability and providing useful diagnostics for algorithm performance.

2 Going back to the low-rank nonnegative matrix completion problem

2.1 SVD Theorem and Example

The Singular Value Decomposition (SVD) theorem states that any matrix $X \in \mathbb{R}^{m \times n}$ can be written as:

$$X = U^X \Sigma^X (V^X)^\top,$$

where:

- $U^X \in \mathbb{R}^{m \times m}$ is an orthogonal matrix (columns are left singular vectors),
- $V^X \in \mathbb{R}^{n \times n}$ is an orthogonal matrix (columns are right singular vectors),
- $\Sigma^X \in \mathbb{R}^{m \times n}$ is a diagonal matrix containing the singular values of X.

Example of SVD.

$$X = \begin{bmatrix} 3 & 1 \\ 1 & 3 \end{bmatrix}.$$

The SVD of X is computed as follows:

$$U^X = \begin{bmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \end{bmatrix},$$
$$\Sigma^X = \begin{bmatrix} 4 & 0 \\ 0 & 2 \end{bmatrix},$$
$$(V^X)^\top = \begin{bmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \end{bmatrix}.$$

Thus, the decomposition is:

$$X = U^X \Sigma^X (V^X)^\top = \begin{bmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \end{bmatrix} \begin{bmatrix} 4 & 0 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \end{bmatrix}^\top.$$

2.2 Link between the Frobenius norm and the Euclidean norm via SVD

To prove:

$$\forall X \in \mathbb{R}^{m \times n}, \quad \|X\|_F = \sqrt{\text{Trace}(X^\top X)} = \|\sigma^X\|_2.$$

Proof. Using the SVD decomposition $X = U^X \Sigma^X (V^X)^\top$, we have:

$$X^\top X = (V^X)(\Sigma^X)^\top (U^X)^\top U^X \Sigma^X (V^X)^\top = V^X (\Sigma^X)^2 (V^X)^\top$$

The eigenvalues of $X^\top X$ are the squared singular values $(\sigma_i^X)^2$. Therefore:

$$\begin{aligned} \text{Trace}(X^T X) &= \text{Trace}(V \Sigma^T \Sigma V^T) \\ &= \text{Trace}(\Sigma^T \Sigma) \quad (\text{since Trace}(AB) = \text{Trace}(BA)) \\ &= \sum_i \sigma_i^2 \\ &= \|\sigma^X\|_2^2 \end{aligned}$$

Taking the square root:

$$\|X\|_F = \sqrt{\sum_i (\sigma_i^X)^2} = \|\sigma^X\|_2.$$

2.3 Link between the rank operator and the cardinality function via SVD

To prove:

$$\forall X \in \mathbb{R}^{m \times n}, \quad \text{rank}(X) = \text{rank}(X^\top X) = \|\sigma^X\|_0.$$

Proof. Using the SVD decomposition $X = U^X \Sigma^X (V^X)^\top$, we know:

$$X^\top X = (V^X)(\Sigma^X)^2 (V^X)^\top.$$

and

$$\begin{aligned}
\operatorname{rank}(X) &= \operatorname{rank}(U^X \Sigma^X (V^X)^T) \\
&= \operatorname{rank}(\Sigma^X) \quad (\text{since } U^X \text{ and } V^X \text{ are full rank}) \\
&= \text{number of nonzero diagonal entries in } \Sigma^X \\
&= \|\sigma^X\|_0
\end{aligned}$$

The rank of $X^\top X$ is the number of nonzero eigenvalues, which corresponds to the number of nonzero singular values σ^X. Thus:

$$\begin{aligned}
\operatorname{rank}((X)^T X) &= \operatorname{rank}(V^X \operatorname{diag}((\sigma^X)^2)(V^X)^T) \\
&= \operatorname{rank}(\operatorname{diag}((\sigma^X)^2)) \quad (\text{since } V^X \text{ is full rank}) \\
&= \text{number of nonzero diagonal entries in } \operatorname{diag}((\sigma^X)^2) \\
&= \|\sigma^X\|_0
\end{aligned}$$

Since $\operatorname{rank}(X) = \operatorname{rank}(X^\top X)$, it follows:

$$\operatorname{rank}(X) = \|\sigma^X\|_0.$$

2.4 A list of admitted results

2.5 Convex approximation of Problem (P1)

2.6 Setup of the ADMM algorithm

(a) Associated ADMM Algorithm for Problem P_3

To solve Problem P_3, we rewrite it in the equivalent form:

$$\min_{X,Y} \mu\|Y\|_N + \frac{1}{2}\|P_\Omega(X) - P_\Omega(B)\|_F^2, \quad \text{subject to } X \geq 0,\ X - Y = 0.$$

The augmented Lagrangian associated with this problem is:

$$\mathcal{L}^\lambda(X, Y, Z) = \mu\|Y\|_N + \frac{1}{2}\|P_\Omega(X) - P_\Omega(B)\|_F^2 + \langle Z, X - Y\rangle + \frac{\lambda}{2}\|X - Y\|_F^2,$$

where $\lambda > 0$ is the penalty parameter.
The ADMM algorithm decomposes this problem into the following three steps:

Step 1: X-update

Solve:

$$X^{k+1} = \arg\min_{X \geq 0} \left(\frac{1}{2}\|P_\Omega(X) - P_\Omega(B)\|_F^2 + \langle Z^k, X - Y^k\rangle + \frac{\lambda}{2}\|X - Y^k\|_F^2 \right).$$

Step 2: Y-update

Solve:

$$Y^{k+1} = \arg\min_{Y} \left(\mu\|Y\|_N + \langle Z^k, X^{k+1} - Y\rangle + \frac{\lambda}{2}\|X^{k+1} - Y\|_F^2 \right).$$

This step involves the proximal operator of the nuclear norm:

$$Y^{k+1} = \operatorname{prox}_{\frac{\mu}{\lambda}\|\cdot\|_N}(X^{k+1} + \frac{Z^k}{\lambda}),$$

Step 3: Z-update

Update the dual variable Z:

$$Z^{k+1} = Z^k + \lambda(X^{k+1} - Y^{k+1}).$$

(b) First step of the ADMM algorithm

Proof. The first step of ADMM requires solving the minimization problem:

$$X_{k+1} = \arg\min_{X \geq 0} \left(\frac{1}{2} \|P_\Omega(X) - P_\Omega(B)\|_F^2 + \langle Z^k, X - Y^k \rangle + \frac{\lambda}{2} \|X - Y^k\|_F^2 \right).$$

Using the decomposition $X = P_\Omega(X) + P_{\Omega^c}(X)$ and the orthogonality of projections $\langle P_\Omega(X), P_{\Omega^c}(X) \rangle_F = 0$, we can rewrite the objective function as:

$$\begin{aligned} &\frac{1}{2} \|P_\Omega(X) - P_\Omega(B)\|_F^2 + \langle Z^k, P_\Omega(X) + P_{\Omega^c}(X) - Y^k \rangle \\ &+ \frac{\lambda}{2} \|P_\Omega(X) + P_{\Omega^c}(X) - Y^k\|_F^2 \end{aligned}$$

Taking derivatives with respect to $P_\Omega(X)$ and setting to zero:

$$P_\Omega(X) - P_\Omega(B) + P_\Omega(Z^k) + \lambda(P_\Omega(X) - P_\Omega(Y^k)) = 0$$

This gives:

$$(1 + \lambda) P_\Omega(X) = P_\Omega(B + \lambda Y^k - Z^k)$$

Therefore:

$$P_\Omega(X) = \frac{1}{1 + \lambda} P_\Omega(B + \lambda Y^k - Z^k)$$

Taking derivatives with respect to $P_{\Omega^c}(X)$ and setting to zero:

$$P_{\Omega^c}(Z^k) + \lambda(P_{\Omega^c}(X) - P_{\Omega^c}(Y^k)) = 0$$

This gives:

$$P_{\Omega^c}(X) = P_{\Omega^c} \left(Y^k - \frac{1}{\lambda} Z^k \right)$$

Combining these results and applying the nonnegativity constraint via the projection operator P_+:

$$X_{k+1} = P_+ \left(\frac{1}{1 + \lambda} P_\Omega(B + \lambda Y^k - Z^k) \right) + P_+ \left(P_{\Omega^c} \left(Y^k - \frac{1}{\lambda} Z^k \right) \right)$$

where P_+ ensures the constraint $X \geq 0$ is satisfied by projecting onto the nonnegative orthant. This proves the required formula. □

(c) Prove second step of the ADMM

The Y-update in ADMM solves:

$$\begin{aligned} Y^{k+1} &= \arg\min_Y \left(\mu \|Y\|_N + \langle Z^k, X^{k+1} - Y \rangle + \frac{\lambda}{2} \|X^{k+1} - Y\|_F^2 \right) \\ &\iff Y^{k+1} = \arg\min_Y \left(\mu \|Y\|_N + \langle Z^k, X^{k+1} - Y \rangle + \frac{\lambda}{2} \|X^{k+1} - Y\|_F^2 \right) \\ &\iff Y^{k+1} = \arg\min_Y \left(\mu \|Y\|_N + \frac{\lambda}{2} \|X^{k+1} - Y\|_F^2 + \langle Z^k, X^{k+1} - Y \rangle + \frac{1}{2\lambda} \|Z^k\|_F^2 \right) \\ &\iff Y^{k+1} = \arg\min_Y \left(\mu \|Y\|_N + \frac{\lambda}{2} \|Y - (X^{k+1} + \frac{1}{\lambda} Z^k)\|_F^2 \right) \\ &\iff Y^{k+1} = \text{prox}_{\frac{\mu}{\lambda} \|\cdot\|_N} \left(X^{k+1} + \frac{1}{\lambda} Z^k \right) \end{aligned}$$

Let $W = X^{k+1} + \frac{1}{\lambda} Z^k$ have SVD decomposition $W = U^k \text{diag}(\sigma^k)(V^k)^\top$. From the admitted results, we know that:

$$\text{prox}_{\frac{\mu}{\lambda} \|\cdot\|_N}(W) = U^k \text{diag} \left(\text{prox}_{\frac{\mu}{\lambda} \|\cdot\|_1}(\sigma^k) \right) (V^k)^\top$$

Therefore:

$$Y^{k+1} = U^k \text{diag} \left(\text{prox}_{\frac{\mu}{\lambda} \|\cdot\|_1}(\sigma^k) \right) (V^k)^\top$$

where $U^k \text{diag}(\sigma^k)(V^k)^\top$ is the SVD of $(X^{k+1} + \frac{1}{\lambda} Z^k)$.

2.7 Matlab code

```
function [X, Y, Z] = matrix_completion_admm(B, Omega, mu, lambda, max_iter, tol)
% Solves the low-rank matrix completion problem using ADMM
% Following the theoretical setup:
% min_{X,Y} mu||Y||_N + (1/2)||P_Omega(X) - P_Omega(B)||_F^2
% subject to X >= 0, X - Y = 0
%
% Input:
%   B: Original matrix with known entries
%   Omega: Binary matrix indicating known entries (1 for known, 0 for unknown)
%   mu: Regularization parameter
%   lambda: ADMM penalty parameter
%   max_iter: Maximum iterations
%   tol: Convergence tolerance

% Get dimensions
[m, n] = size(B);

% Initialize variables
X = zeros(m, n);
Y = zeros(m, n);
Z = zeros(m, n);

% Define projection operators
P_Omega = @(X) X .* Omega;      % Projection onto known entries
P_Omegac = @(X) X .* (1-Omega); % Projection onto unknown entries
P_plus = @(X) max(X, 0);        % Projection onto nonnegative orthant

P_Omega_B = P_Omega(B);
Y_prev = Y;

% Main ADMM loop
for iter = 1:max_iter
    % X-update:
    % X_{k+1} = P_+((1/(1+  ))P_  (B +  Y_k  - Z_k)) + P_+(P_  c(Y_k - (1/  )Z_k))

    % First term of xk: (1/(1+  ))P_  (B +  Y_k  - Z_k)
    term1 = (1/(1 + lambda)) * P_Omega(B + lambda*Y - Z);

    % Second term of xk: P_  c(Y_k - (1/  )Z_k)
    term2 = P_Omegac(Y - (1/lambda)*Z);

    % Combine terms and project onto nonnegative orthant
    X = P_plus(term1) + P_plus(term2);

    % Y-update:
    % Y_{k+1} = prox_{(  /  )||  ||_N}(X_{k+1} + (1/  )Z_k)
    [U, S, V] = svd(X + (1/lambda)*Z, 'econ');
    % Apply soft thresholding to singular values
    s = diag(S);
    s_thresh = max(s - mu/lambda, 0);
    Y = U * diag(s_thresh) * V';

    % Z-update (dual variable update):

    Z_prev = Z;
    Z = Z + lambda*(X - Y);

    % Check convergence
    primal_res = norm(X - Y, 'fro');
    dual_res = lambda * norm(Y - Y_prev, 'fro');

    if primal_res < tol && dual_res < tol
        fprintf('Converged at iteration %d\n', iter);
        fprintf('Primal residual: %e\n', primal_res);
        fprintf('Dual residual: %e\n', dual_res);
        break;
    end

```

```
    % Store Y for next iteration's dual residual
    Y_prev = Y;
end
end
```

Func 2: ADMM Algorithm for Matrix Completion

To test the effectiveness of our ADMM implementation for matrix completion, we create a test function that generates synthetic data and evaluates the algorithm's performance.

Test Function Implementation

```
function test_matrix_completion()
    m = 50;  % rows
    n = 40;  % columns
    r = 3;   % rank

    % Create low-rank matrix
    U = randn(m, r);
    V = randn(n, r);
    B_true = max(U * V', 0);  % True nonnegative matrix

    % Sample entries randomly
    sample_rate = 0.5;
    Omega = rand(m, n) < sample_rate;
    B = B_true .* Omega;  % Observed matrix

    % Algorithm parameters
    mu = 0.1;
    lambda = 2.0;
    max_iter = 10000;
    tol = 1e-6;

    % Run ADMM
    [X, Y, Z] = matrix_completion_admm(B, Omega, mu, lambda, max_iter, tol);

    % Evaluate and visualize results
    rel_error = norm(X - B_true, 'fro') / norm(B_true, 'fro');
    fprintf('Relative error: %e\n', rel_error);
    fprintf('Rank of solution: %d\n', rank(X, 1e-6));

    % Visualization code...
end
```

Func 3: Test Function for Matrix Completion

The test process begins by generating a synthetic low-rank matrix. This involves several steps:

1. **Dimension Selection:**
   - Matrix dimensions: $m = 50$ rows and $n = 40$ columns
   - True rank: $r = 3$ (significantly smaller than matrix dimensions)
2. **True Matrix Construction:**
   - Generate random factors $U \in \mathbb{R}^{m \times r}$ and $V \in \mathbb{R}^{n \times r}$
   - Construct $B_{\text{true}} = \max(UV^T, 0)$ to ensure nonnegativity
   - This construction guarantees a matrix with rank at most r
3. **Creating Observed Matrix:**
   - Generate observation mask Ω with sampling rate 0.5 (50% observed entries)
   - Create partially observed matrix $B = B_{\text{true}} \odot \Omega$

2.7.1 Algorithm Parameters

The ADMM algorithm requires careful parameter selection:

- $\mu = 0.1$: Controls the strength of nuclear norm regularization
- $\lambda = 2.0$: ADMM penalty parameter
- tol $= 10^{-6}$: Convergence tolerance
- max_iter $= 10000$: Maximum iterations allowed

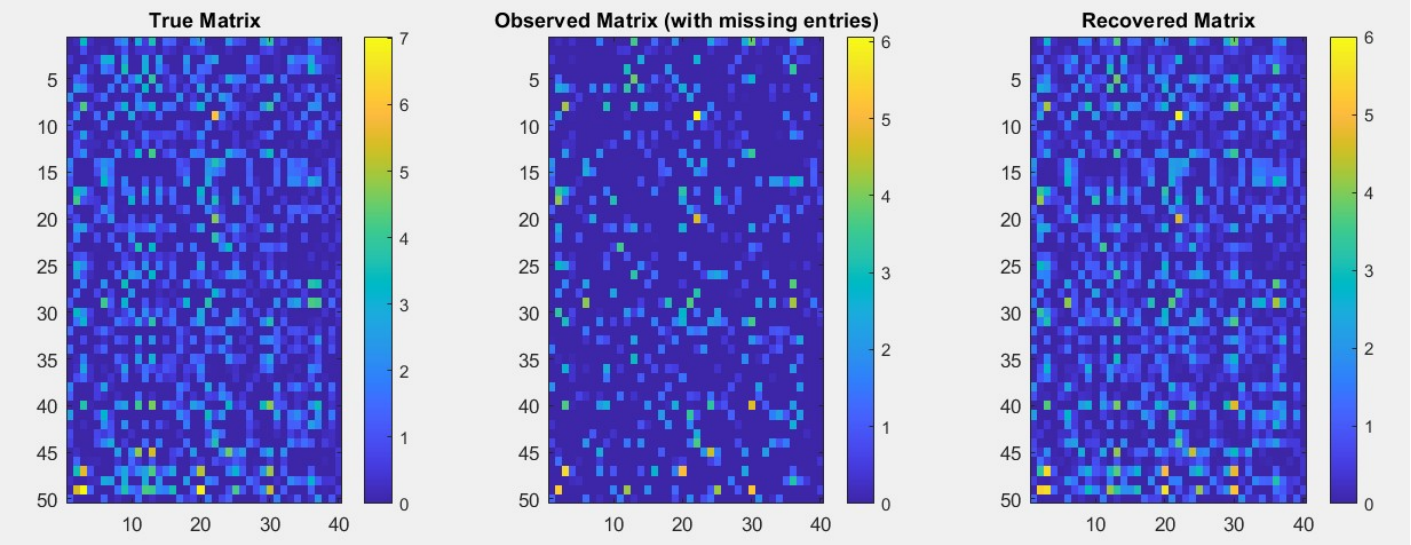

Figure 1: Comparison of True, Observed, and Recovered Matrices. Left: Original matrix with true values. Middle: Observed matrix with 50% missing entries (shown in dark blue). Right: Recovered matrix using ADMM algorithm.

The visual results demonstrate the effectiveness of our matrix completion algorithm. The recovered matrix (right) shows remarkable similarity to the true matrix (left), despite having access to only 50% of the entries in the observed matrix (middle). Key patterns and intensity distributions are well-preserved, particularly the distinctive yellow spots (high values) and the overall blue-gradient structure.
The algorithm converged after 5093 iterations with the following metrics:

- Primal residual: 6.299637×10^{-9}
- Dual residual: 9.990295×10^{-7}
- Relative error: 27.97%

The small residuals indicate strong convergence, suggesting that the constraints are well-satisfied. The relative error of approximately 28% is reasonable considering that half of the original data was missing.
However, the rank of the solution (25) is significantly higher than our true rank ($r = 3$), which warrants discussion. This higher rank can be attributed to several factors:

1. The relatively small value of μ (0.1) places less emphasis on rank minimization, allowing the solution to better fit the observed data at the cost of higher rank
2. The nonnegativity constraint ($\max(UV^T, 0)$) in the true matrix construction can increase the effective rank
3. Numerical artifacts from the optimization process contribute to small but non-zero singular values

The visual quality of the reconstruction and the small convergence residuals suggest that the algorithm has successfully captured the essential structure of the original matrix, despite the higher numerical rank.

3 Matlab Code Implementation

The implementation of the matrix completion algorithm is organized into four MATLAB files.

1. `ADMM.m`: This file contains the main iteration loop and update steps for solving Problem (Q_5).

2. `matrix_completion_admm.m`: The primary implementation of the matrix completion algorithm using ADMM. This file contains the specialized version of ADMM for handling matrix completion problems, including the nuclear norm minimization and projection operations.

3. `test_matrix_completion.m`: A test script that generates synthetic data and demonstrates the basic usage of the matrix completion algorithm with a simple example.

4. `analyze_matrix_completion.m`: A comprehensive analysis script that tests different parameter combinations and generates performance metrics.