MDS6106 - Optimization and Modeling: Exercise 5

December 15, 2024

Name: *Zijin CAI*
Student ID: *224040002*

Assignment A5.1: Smooth Image Inpainting

the total variation image inpainting problem:

$$\min_{\mathbf{x}\in\mathbb{R}^{mn}} \frac{1}{2}\|\mathbf{A}\mathbf{x}-\mathbf{b}\|^2 + \mu \sum_{i=1}^{m-1}\sum_{j=1}^{n-1} \|\mathbf{D}_{(i,j)}\mathbf{x}\|_2, \quad \mu > 0 \tag{1}$$

- $\mathbf{x} = \text{vec}(\mathbf{X})$ is the vectorized image $\mathbf{X} \in \mathbb{R}^{m\times n}$
- $\mathbf{A} \in \mathbb{R}^{s\times mn}$ is the selection matrix that selects the known pixels given the inpainting mask
- $\mathbf{b} \in \mathbb{R}^s$ contains the uncorrupted pixel values information
- $\mathbf{D}_{(i,j)}$ is the discrete image gradient operator that computes the gradient of the image at pixel (i, j)
- $\mu > 0$ is the given regularization parameter

(a)

the python code of image operations and example plots of damaged images:

```
original_image = cv2.imread('/path/to/image', cv2.IMREAD_GRAYSCALE)
mask = cv2.imread('path/to/mask', cv2.IMREAD_GRAYSCALE)
mask_color = np.array([53, 130, 134]) # any color

original_image = cv2.resize(original_image, (mask.shape[1], mask.shape[0]))
original_image_bgr = cv2.cvtColor(original_image, cv2.COLOR_GRAY2BGR)

damaged_image = original_image_bgr.copy()
damaged_image[mask == 0] = mask_color

# Continue to the plot part displaying the original and damaged images
```

(b)

the smoothed inpainting problem:

$$\min_x f(x) = \frac{1}{2}\|\mathbf{Ax} - \mathbf{b}\|^2 + \tilde{\varphi}(x), \quad \text{where } \tilde{\varphi}(x) = \frac{\mu}{2}\|Dx\|_2^2 \tag{2}$$

the gradient of the smoothed inpainting problem, and set it to zero:

$$\nabla f(x) = \mathbf{A}^T(\mathbf{Ax} - \mathbf{b}) + \mu\mathbf{D}^T\mathbf{Dx} = 0 \quad \Rightarrow \quad \mathbf{A}^T\mathbf{Ax} + \mu\mathbf{D}^T\mathbf{Dx} = \mathbf{A}^T\mathbf{b} \tag{3}$$

given that $\mathbf{A}^T\mathbf{A}$ and $\mathbf{D}^T\mathbf{D}$ are positive semi-definite, the Hessian of $f(x)$ is thus positive definite:

$$\nabla^2 f(x) = \mathbf{A}^T\mathbf{A} + \mu\mathbf{D}^T\mathbf{D} \tag{4}$$

Therefore, the smoothed inpainting problem is convex, and this shows that a point x is a solution to the smoothed inpainting problem if and only if it satisfies the equation $(\mathbf{A}^T\mathbf{A} + \mu\mathbf{D}^T\mathbf{D})\mathbf{x} = \mathbf{A}^T\mathbf{b}$.

(c)

the python code of implementing the `cg_method()`:

```
def cg_method(B, c, x0=None, tol=1e-4, max_iterations=1000, epsilon=1e-10):
    n = len(c)
    if x0 is None:
        x0 = np.zeros(n)

    x = x0.copy()
    r = B.dot(x) - c
    p = -r.copy()

    for k in range(max_iterations):
        Ap = B.dot(p)
        denominator = np.dot(p, Ap) + epsilon  # Add epsilon to avoid division by zero
        alpha = np.dot(r, r) / denominator
        x += alpha * p
        r_new = r + alpha * Ap

        if np.linalg.norm(r_new) <= tol:
            break

        beta = np.dot(r_new, r_new) / (np.dot(r, r) + epsilon)
        p = -r_new + beta * p
        r = r_new

    return x
```

(d)

the peak-signal-to-noise ratio (PSNR) and python code implementation:

$$\text{PSNR} = 10\log_{10}\left(mn/\|\mathbf{x} - \mathbf{u}\|^2\right) \quad \text{where } \mathbf{u} \in \mathbb{R}^{mn} \text{ is the vectorized original image} \tag{5}$$

```
def calculate_psnr(original, reconstructed):
    if np.mean((original - reconstructed) ** 2) == 0:
        return float('inf')

    img_size = original.shape[0] * original.shape[1]
    psnr = 10 * np.log10(img_size / np.linalg.norm(original - reconstructed) ** 2)

    return psnr
```

the comparison of different implementations ($\mathbf{x}_0 = 0$, and `tol`$= 1e-4$, $\mu = 0.01$):

image	**img_size**	**PSNR**(`cg_method`)	**cpu-time**(`cg_method`)	**PSNR**(`std_solver`)	**cpu-time**(`std_solver`)
`Ranni`	240×240	3.71267	0.00636	3.71267	2.07861
`niko`	640×640	5.29049	0.00463	5.29049	2.16062
`donk`	768×768	10.00728	0.00556	10.00728	2.06226

While it may be challenging to discern differences from the original images at a cursory glance, the detailed comparison are provided in the table . The conjugate gradient method significantly outperforms the standard solver in terms of speed, while maintaining comparable PSNR values applying the same tolerance. In fact, the `cg_method` is capable of reconstructing most images much faster than the standard solver without compromising on quality.

Assignment A5.2: KKT-Conditions and Optimality

(a)

Given the optimization problem:

$$\begin{aligned} \min_{x\in\mathbb{R}^3} \quad & f(x) := x_1^2 + x_2^2 + x_3^2 + x_1x_2 + x_2x_3 - 2x_1 - 5x_2 - 6x_3 \\ \text{s.t.} \quad & x_1 + x_2 + x_3 \le 1, \quad x_1 - x_2^2 = 0. \end{aligned}$$

The KKT conditions are:

1. **Stationarity:**

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial x_1} &= 2x_1 + x_2 - 2 + \lambda + \mu = 0, \\ \frac{\partial \mathcal{L}}{\partial x_2} &= 2x_2 + x_1 + x_3 - 5 - 2\mu x_2 = 0, \\ \frac{\partial \mathcal{L}}{\partial x_3} &= 2x_3 + x_2 - 6 = 0. \end{aligned}$$

2. **Primal feasibility:**

$$\begin{aligned} x_1 + x_2 + x_3 &\le 1, \\ x_1 - x_2^2 &= 0. \end{aligned}$$

3. **Dual feasibility:**

$$\lambda \ge 0.$$

4. **Complementary slackness:**

$$\lambda(x_1 + x_2 + x_3 - 1) = 0.$$

(b)

Given the point $\bar{x} = [0, 0, 1]^\top$, we verify it as a strict local minimum:

1. **KKT Conditions:** - **Stationarity:** Substitute $\bar{x}$ into the stationarity conditions:

$$\begin{aligned} 2(0) + 0 - 2 + \lambda_1 + \lambda_2 &= 0, \\ 2(0) + 0 + 1 - 5 + \lambda_1 - 2\lambda_2(0) &= 0, \\ 2(1) + 0 - 6 + \lambda_1 &= 0. \end{aligned}$$

Solving these, we find $\lambda_1 = 4$ and $\lambda_2 = -2$.

- **Primal feasibility:** Check if $\bar{x}$ satisfies the constraints:

$$0 + 0 + 1 \le 1, \quad 0 - 0^2 = 0.$$

Both constraints are satisfied.

- **Dual feasibility:** $\lambda_1 = 4 \ge 0$.

- **Complementary slackness:** For the inequality constraint, $\lambda_2(0+0+1-1)=0$

2. **Second-Order Optimality Conditions:** - The Hessian of the Lagrangian function must be positive semi-definite in the direction of feasible descent for the inequality constraints.

For the Lagrangian $\mathcal{L}(x,\lambda_1,\lambda_2)$, the Hessian of $f(x)$ is given by:

$$H_f = \begin{bmatrix} 2 & 1 & 0 \\ 1 & 2-2\lambda_2 & 1 \\ 0 & 1 & 2 \end{bmatrix}$$

At the point $\bar{x}=[0,0,1]^\top$, with $\lambda_2=2$, the Hessian matrix H becomes:

$$H = \begin{bmatrix} 2 & 1 & 0 \\ 1 & 6 & 1 \\ 0 & 1 & 2 \end{bmatrix}$$

To verify that $\bar{x}=[0,0,1]^\top$ is a strict local minimum, we need to check if the Hessian matrix H is positive definite. A matrix is positive definite if all its leading principal minors are positive.

1. The first leading principal minor is:
$$\det\begin{bmatrix}2\end{bmatrix} = 2 > 0$$

2. The second leading principal minor is:
$$\det\begin{bmatrix} 2 & 1 \\ 1 & 6 \end{bmatrix} = 2\cdot 6 - 1\cdot 1 = 11 > 0$$

3. The third leading principal minor is the determinant of the full Hessian matrix H:
$$\det\begin{bmatrix} 2 & 1 & 0 \\ 1 & 6 & 1 \\ 0 & 1 & 2 \end{bmatrix} = 2(6\cdot 2 - 1\cdot 1) - 1(1\cdot 2 - 0\cdot 1) + 0 = 2\cdot 11 - 2 = 20 > 0$$

Since all leading principal minors are positive, the Hessian matrix H is positive definite, which suggests that $\bar{x}=[0,0,1]^\top$ is a strict local minimum.

(c)

To determine if the optimization problem is convex, we need to check the convexity of the objective function and the constraints:

1. **Objective Function Convexity:** A function is convex if its Hessian matrix is positive semi-definite. The Hessian of the objective function $f(x)$ is:

$$H_f = \begin{bmatrix} 2 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 2 \end{bmatrix}$$

We have already calculated the leading principal minors and found them to be positive, which implies that H_f is positive definite. Therefore, $f(x)$ is convex.

2. **Constraints Convexity:** - The inequality constraint $g_1(x) = x_1 + x_2 + x_3 - 1 \leq 0$ is linear and hence convex. - The equality constraint $g_2(x) = x_1 - x_2^2 = 0$ includes a term x_2^2, which makes it non-convex.

Since the equality constraint is not affine, the optimization problem is not convex.