COMP0083: Optimization

2024

Contents

Part I Questions with multiple answers

Question 1

Answer: (a)

Here we plot all the options.

(b) is not a convex function as we can show that

$$f((1-\lambda)x + \lambda y) > (1-\lambda)f(x) + \lambda f(y), \tag{1}$$

if we choose x and y at different side of intersection points.

(c) is not convex as we can find a local maximum from the plot. Mathematically, we can compute the first and the second derivative as

$$\begin{aligned} h'(x) &= \frac{d}{dx}\left(x^4 - 5 - e^x\right) = 4x^3 - e^x, \\ h''(x) &= \frac{d}{dx}\left(4x^3 - e^x\right) = 12x^2 - e^x. \end{aligned} \tag{2}$$

We can find $h''(0) < 0$ when x is small. This further show that $h(x)$ is not a convex function.

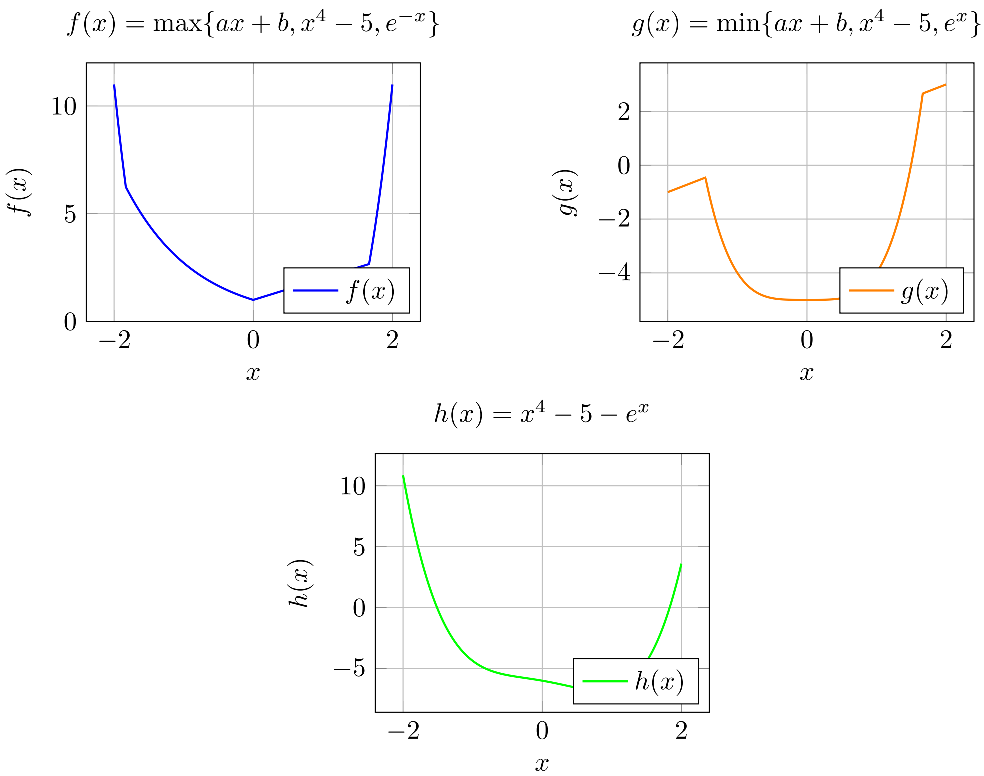


Figure 1: Plots of $f(x)$, $g(x)$, and $h(x)$

Question 2

Answer: (a)

We can take the subdifferential $\partial f(x)$ as

$$f'(x) = \begin{cases} -1 & \text{if } x \in [-1, 0) \\ [-1, 2x] & \text{if } x = 0 \\ 2x & \text{if } x > 0 \end{cases} \tag{3}$$

(a) and (b) are different at $x = 0$. From the equation above, we find the value of subdifferential at $x = 0$ is $[-1, 0]$. In that case, the correct plot is (a).

Question 3

Answer: (a)

We can express $f(x)$ as

$$\begin{aligned} &f(x) = \langle Ax, x\rangle + \langle x, b\rangle + c, \\ \Rightarrow &f(x) = x^* Ax + b^* x + c. \end{aligned} \tag{4}$$

Then, we take the gradient of $f(x)$:

$$\begin{aligned} &\nabla f(x) = \nabla(x^* Ax) + \nabla(b^* x), \\ \Rightarrow &\nabla f(x) = A^* x + Ax + b, \end{aligned} \tag{5}$$

where A not need to be a symmetric matrix.

Question 4

Answer: (a)

The Fenchel conjugate of $f(x)$ is:

$$f^*(u) = \sup_{x\in\mathbb{R}} \{\langle u, x\rangle - f(x)\}. \tag{6}$$

In this question, we substitute $f(x) = g(3x)$:

$$\begin{aligned} &f^*(u) = \sup_{x\in\mathbb{R}} \{\langle u, x\rangle - g(3x)\}, \\ \Rightarrow &f^*(u) = \sup_{x\in\mathbb{R}} \{\langle u, \frac{z}{3}\rangle - g(z)\}, \\ \Rightarrow &f^*(u) = \sup_{x\in\mathbb{R}} \{\langle \frac{u}{3}, z\rangle - g(z)\}, \\ \Rightarrow &f^*(u) = g^*(\frac{u}{3}). \end{aligned} \tag{7}$$

Part II Theory on convex analysis and optimization

Problem 1

1.1. The Fenchel conjugate of $f(x)$ is:

$$f^*(u) = \sup_{x\in\mathbb{R}} \{\langle u, x\rangle - f(x)\}. \tag{8}$$

Due to the function structure, we can only consider $x > 0$ condition. Substitute $f(x) = -\log x$:

$$f^*(u) = \sup_{x>0} (ux + \log x). \tag{9}$$

To find the value of u, we can take the derivative of conjugate function:

$$\frac{\partial f^*(u)}{\partial x} = \sup_{x>0} (u + \frac{1}{x}) = 0 \Rightarrow x = -\frac{1}{u}. \tag{10}$$

If $u \leq 0$, then $x > 0$, we will have a negative term ux which pulls the function toward $-\infty$. In that case, we have $f^*(u) = -\infty$. If $u > 0$, then $x < 0$. We have $f^*(u) = 1 - \log u$.

Finally, we get the conjugate of $f(x)$:

$$f^*(u) = \begin{cases} 1 - \log u & \text{if } u > 0 \\ -\infty & \text{if } u \leq 0 \end{cases} \tag{11}$$

1.2. The Fenchel conjugate is:

$$f^*(u) = \sup_{x\in\mathbb{R}} \left(ux - 2x^2\right). \tag{12}$$

Take the derivative:

$$\frac{\partial f^*(u)}{\partial x} = \sup_{x>0} (u - 4x) = 0 \Rightarrow x = \frac{u}{4}. \tag{13}$$

Substitute u back, we have:

$$f^*(u) = \frac{u^2}{4} - 2\left(\frac{u}{4}\right)^2 = \frac{u^2}{8}. \tag{14}$$

1.3. The Fenchel conjugate is:

$$f^*(u) = \sup_{x\in[-1,1]} (ux). \tag{15}$$

If $u > 0$, the supremum is achieved at $x = 1$:

$$f^*(u) = u. \tag{16}$$

If $u < 0$, the supremum is achieved at $x = -1$:

$$f^*(u) = -u. \tag{17}$$

If $u = 0$, the supremum is:

$$f^*(u) = 0. \tag{18}$$

Finally, we get the conjugate of $f(x)$:

$$f^*(u) = |u|. \tag{19}$$

Problem 2

2.1. Let us start with $n = 2$. From the definition of convexity, we have

$$f(\lambda x_1 + (1-\lambda)x_2) \le \lambda f(x_1) + (1-\lambda) f(x_2). \tag{20}$$

This satisfies Jensen's inequality for $n = 2$.

Now, we assume Jensen's inequality for $n = k$:

$$f\left(\sum_{i=1}^{k} \lambda_i x_i\right) \le \sum_{i=1}^{k} \lambda_i f(x_i). \tag{21}$$

Consider $n = k + 1$:

$$\sum_{i=1}^{k+1} \lambda_i x_i = \lambda_{k+1} x_{k+1} + (1-\lambda_{k+1}) \left(\frac{\sum_{i=1}^{k} \lambda_i x_i}{1-\lambda_{k+1}}\right), \tag{22}$$

where we can apply the convexity:

$$f\left(\sum_{i=1}^{k+1} \lambda_i x_i\right) \le \lambda_{k+1} f(x_{k+1}) + (1-\lambda_{k+1}) f\left(\frac{\sum_{i=1}^{k} \lambda_i x_i}{1-\lambda_{k+1}}\right). \tag{23}$$

Based on the assumption, we have:

$$f\left(\frac{\sum_{i=1}^{k} \lambda_i x_i}{1-\lambda_{k+1}}\right) \le \sum_{i=1}^{k} \frac{\lambda_i}{1-\lambda_{k+1}} f(x_i) \tag{24}$$

Summarize:

$$f\left(\sum_{i=1}^{k+1} \lambda_i x_i\right) \le \lambda_{k+1} f(x_{k+1}) + \sum_{i=1}^{k} \lambda_i f(x_i) = \sum_{i=1}^{k+1} \lambda_i f(x_i) \tag{25}$$

By induction, Jensen's inequality holds for all n. ■

2.2. Let $f(x) = -\log(x)$. We need to show that:

$$f(\lambda x_1 + (1-\lambda)x_2) \le \lambda f(x_1) + (1-\lambda) f(x_2). \tag{26}$$

Substitute the function, we have

$$-\log(\lambda x_1 + (1-\lambda)x_2) \le -(\lambda \log(x_1) + (1-\lambda)\log(x_2)). \tag{27}$$

If we remove the negative sign, we get the logarithmic inequality which further prove the above inequality. In that case, $f(x) = -\log(x)$ is convex. ■

2.3. We can apply Jensen's inequality for $f(x) = -\log(x)$:

$$\begin{aligned} -\log\left(\sum_{i=1}^{n} \lambda_i x_i\right) &\leq \sum_{i=1}^{n} \lambda_i \left(-\log\left(x_i\right)\right), \\ \Rightarrow \log\left(\sum_{i=1}^{n} \lambda_i x_i\right) &\geq \sum_{i=1}^{n} \lambda_i \log\left(x_i\right). \end{aligned} \tag{28}$$

Let $\lambda_i = \frac{1}{n}$, so $\sum_{i=1}^{n} \lambda_i = 1$. Then:

$$\begin{aligned} \log\left(\frac{1}{n}\sum_{i=1}^{n} x_i\right) &\geq \frac{1}{n}\sum_{i=1}^{n} \log\left(x_i\right), \\ \xRightarrow{\exp} \frac{1}{n}\sum_{i=1}^{n} x_i &\geq \exp\left(\frac{1}{n}\sum_{i=1}^{n} \log\left(x_i\right)\right). \\ \Rightarrow \frac{1}{n}\sum_{i=1}^{n} x_i &\geq \sqrt[n]{x_1 \cdots x_n}. \end{aligned} \tag{29}$$

■

Problem 3

By the definition of the convex hull, any $x \in C$ can be expressed as

$$x = \sum_{i=1}^{m} \lambda_i a_i, \qquad \text{with } \lambda_i \geq 0, \sum_{i=1}^{m} \lambda_i = 1. \tag{30}$$

Combining that f is a convex function, we have:

$$f(x) = f\left(\sum_{i=1}^{m} \lambda_i a_i\right) \leq \sum_{i=1}^{m} \lambda_i f\left(a_i\right) \leq \max_{i \cdots m} f(a_i). \tag{31}$$

Therefore, the maximum of $f(x)$ on C is attained at one of the vertices $a_1, \ldots, a_m$. ■

Problem 4

We can first expend this function as:

$$f(x, y) = \|2x - y\|_2^2 = (2x - y)^\top (2x - y) = 4x^\top x - 4x^\top y + y^\top y. \tag{32}$$

Based on the property of the convex function, we need to show every term in $f(x, y)$ is convex. The first term $4x^\top x$ is convex as it is quadratic in x. With same reason, $y^\top y$ is convex as well. For $4x^\top y$, one can show it based on definition. Let us assume $g(x, y) = -4x^\top y$. Then, we need to prove the inequality for all $x_1, x_2 \in \mathbb{R}^n, y_1, y_2 \in \mathbb{R}^n$, and $\lambda \in [0, 1]$:

$$g\left(\lambda\left(x_1, y_1\right) + (1 - \lambda)\left(x_2, y_2\right)\right) \leq \lambda g\left(x_1, y_1\right) + (1 - \lambda) g\left(x_2, y_2\right) \tag{33}$$

The left-hand side can be written as:

$$\begin{aligned} LHS &= -4\left(\lambda x_1 + (1 - \lambda) x_2\right)^\top \left(\lambda y_1 + (1 - \lambda) y_2\right), \\ \Rightarrow LHS &= -4\left[\lambda^2\left(x_1^\top y_1\right) + \lambda(1 - \lambda)\left(x_1^\top y_2\right) + \lambda(1 - \lambda)\left(x_2^\top y_1\right) + (1 - \lambda)^2\left(x_2^\top y_2\right)\right]. \end{aligned} \tag{34}$$

The right-hand side is:

$$RHS = -4\left(\lambda x_1^\top y_1 + (1 - \lambda) x_2^\top y_2\right) \tag{35}$$

Then, we can show that:

$$LHS - RHS = 4\lambda(1 - \lambda)\left[x_1^\top y_1 + x_2^\top y_2 - x_1^\top y_2 - x_2^\top y_1\right] = 0 \tag{36}$$

Thus, $g(x, y)$ is convex as well. Also, $g(x, y)$ is linear in both x and y which shows convexity. In summary, all the terms in $f(x, y)$ are convex which leads to the convexity of $f(x, y)$. ■

Problem 5

5.1. The Fenchel-Rockafellar duality theorem states that:

$$\min_x f(x) + g(Ax) \text{ is dual to } \max_\lambda -f^*\left(-A^T\lambda\right) - g^*(\lambda). \tag{37}$$

In our problem, we define $f(x)$ as $\frac{1}{2}\|x\|^2$ and its Fechel conjugate is:

$$f^*(z) = \sup_x \left(\langle z, x\rangle - \frac{1}{2}\|x\|^2\right). \tag{38}$$

To find maximum of the objective, we can take the gradient:

$$\nabla_x \left(\langle z, x\rangle - \frac{1}{2}\|x\|^2\right) = 0 \Rightarrow z = x. \tag{39}$$

Substituting back:

$$f^*(z) = \frac{1}{2}\|z\|^2. \tag{40}$$

Following the hint, we define $g(Ax)$ based on the constraint:

$$g(Ax) = \begin{cases} 0 & \text{if } Ax - b \leq \varepsilon \\ +\infty & \text{otherwise} \end{cases} \tag{41}$$

Its Fechel conjugate is:

$$g^*(\lambda) = \sup_{y\in\mathbb{R}^n} \left(\langle \lambda, y\rangle - g(y)\right), \tag{42}$$

where we set $y = Ax$. As $g(y) = 0$ for $y - b \leq \varepsilon$, the supremum is also in this domain:

$$g^*(\lambda) = \sup_{y-b\leq\varepsilon} \left(\langle \lambda, y\rangle\right). \tag{43}$$

Here we can express our y as:

$$y = b + u, \qquad \text{where } u \in [-\varepsilon, \varepsilon]^n. \tag{44}$$

In that case, we have:

$$\begin{aligned} g^*(\lambda) &= \sup_{u\in[-\varepsilon,\varepsilon]^n} \langle \lambda, b + u\rangle = \langle \lambda, b\rangle + \sup_{u\in[-\varepsilon,\varepsilon]^n} \langle \lambda, u\rangle, \\ \Rightarrow g^*(\lambda) &= \langle \lambda, b\rangle + \varepsilon\|\lambda\|. \end{aligned} \tag{45}$$

Now, we have two conjugates. Substitute these back:

$$\text{Dual} = \max_\lambda -\frac{1}{2}\left\|A^T\lambda\right\|^2 - \langle \lambda, b\rangle - \varepsilon\|\lambda\|, \tag{46}$$

which can be expressed in a different way:

$$\min_\lambda \frac{1}{2}\left\|A^T\lambda\right\|^2 + \langle \lambda, b\rangle + \varepsilon\|\lambda\|. \tag{47}$$

■

5.2. To show whether the strong duality holds or not, we can follow the **Theorem 8.1.1** in the notes. This gives an equivalent statement:

$$f(\hat{x}) + f^*\left(-A^*\hat{u}\right) = \langle -A^*\hat{u}, \hat{x}\rangle \text{ and } g(A\hat{x}) + g^*(\hat{u}) = \langle \hat{u}, A\hat{x}\rangle. \tag{48}$$

In that case, we can check it by substitution. For the first part, we have:

$$f(\hat{x}) + f^*\left(-A^*\hat{u}\right) = \frac{1}{2}\|\hat{x}\|^2 + \sup_x \left(\langle -A^*\hat{u}, x\rangle - \frac{1}{2}\|\hat{x}\|^2\right). \tag{49}$$

As we mention in 5.1, the supremum is achieved when $x = -A^*\hat{u}$. Thus, we can have

$$f(\hat{x}) + f^*\left(-A^*\hat{u}\right) = \frac{1}{2}\|\hat{x}\|^2 + \langle -A^*\hat{u}, x\rangle - \frac{1}{2}\|\hat{x}\|^2 = \langle -A^*\hat{u}, x\rangle. \tag{50}$$

For the second part, we have:

$$g(A\hat{x}) + g^*(\hat{u}) = \sup_{A\hat{x}\in\mathbb{R}^n} (\langle \hat{u}, A\hat{x}\rangle - g(A\hat{x})) = \langle \hat{u}, A\hat{x}\rangle, \tag{51}$$

where $g(A\hat{x})$ is zero within the constraint. Thus, we show that our problem satisfies the conditions for strong duality.

5.3. The KKT conditions are shown in Theorem 8.1.1. Combine with the problem:

$$\begin{aligned} \hat{x} &\in \partial f^*\left(-A^*\hat{u}\right), \\ A\hat{x} &\in \partial g^*(\hat{u}), \\ -A^*\hat{u} &\in \partial f(\hat{x}), \\ \hat{u} &\in \partial g(A\hat{x}). \end{aligned} \tag{52}$$

5.4. FISTA aim to solve the problem:

$$\min_{x\in\mathbb{R}^d} F(x) = f(x) + g(x), \tag{53}$$

where $f(x)$ is smooth and convex, with a Lipschitz continuous gradient. At the same time, $g(x)$ is convex but not necessarily smooth. We have $(t_k)_{k\in N}, t_0 = 1, t_k \geq 1$ and $t_k^2 - t_k \leq t_{k-1} \forall k$. Let $y_0 \in X, \gamma \leq 1/L$. Define the iterative steps in FISTA:

$$\begin{aligned} x_{k+1} &= \operatorname{prox}_{\gamma,g}\left(y_k - \gamma\nabla f^*\left(y_k\right)\right), \\ y_{k+1} &= x_{k+1} + \frac{t_k - 1}{t_{k+1}}\left(x_{k+1} - x_k\right). \end{aligned} \tag{54}$$

In the notes **Theorem 5.3.5**, we have:

$$\left\|x_{k+1} - x\right\|^2 \leq \left\|x_k - x\right\|^2 + 2\gamma\left(F(x) - F\left(x_{k+1}\right)\right) + (\gamma L - 1)\left\|x_{k+1} - x_k\right\|^2, \forall x \in X. \tag{55}$$

Follow the iterative steps defined above, we have a similar inequality:

$$\left\|x_{k+1} - x\right\|^2 \leq \left\|y_k - x\right\|^2 + 2\gamma\left(F(x) - F\left(x_{k+1}\right)\right) + (\gamma L - 1)\left\|x_{k+1} - x_k\right\|^2, \forall x \in X. \tag{56}$$

Since $\gamma L - 1 \leq 0$, we have:

$$F\left(x_{k+1}\right) + \frac{\left\|x_{k+1} - x\right\|^2}{2\gamma} \leq F(x) + \frac{\left\|y_k - x\right\|^2}{2\gamma}, \quad \forall x \in X. \tag{57}$$

To further solve above inequality, we can start to find $\left\|x_{k+1} - x\right\|^2$ and $\left\|y_k - x\right\|^2$. It is a good idea to focus on the iterative steps. y_{k+1} can be written as:

$$y_{k+1} = \left(1 - \frac{1}{t_{k+1}}\right) x_{k+1} + \frac{1}{t_{k+1}}[x_k + t_k\left(x_{k+1} - x_k\right)], \tag{58}$$

where we denote $v_{k+1} = x_k + t_k\left(x_{k+1} - x_k\right)$. This further give us the expression of y_k:

$$y_k = \left(1 - \frac{1}{t_k}\right) x_k + \frac{v_k}{t_k}. \tag{59}$$

On the other hand, we have x_{k+1} as:

$$x_{k+1} = \left(1 - \frac{1}{t_k}\right) x_k + \frac{v_{k+1}}{t_k}. \tag{60}$$

Setting x as convex combination of x_k and x_*, we then have:

$$\begin{aligned} x_{k+1} - x &= \frac{v_{k+1} - x_*}{t_k}, \\ y_k - x &= \frac{v_k - x_*}{t_k}. \end{aligned} \tag{61}$$

Substitute these back to the inequality:

$$
\begin{aligned}
&F\left(x_{k+1}\right)+\frac{\left\|v_{k+1}-x_*\right\|^2}{2\gamma t_k^2} \leq\left(1-\frac{1}{t_k}\right) F(x_k)+\frac{1}{t_k} F(x_*)+\frac{\left\|v_k-x_*\right\|^2}{2\gamma t_k^2}, \\
\Rightarrow & F\left(x_{k+1}\right)-F\left(x_*\right)+\frac{\left\|v_{k+1}-x_*\right\|^2}{2\gamma t_k^2} \leq\left(1-\frac{1}{t_k}\right)\left(F(x_k)-F(x_*)\right)+\frac{\left\|v_k-x_*\right\|^2}{2\gamma t_k^2}, \\
\Rightarrow & t_k^2\left(F\left(x_{k+1}\right)-F\left(x_*\right)\right)+\frac{\left\|v_{k+1}-x_*\right\|^2}{2\gamma} \leq\left(t_k^2-t_k\right)\left(F(x_k)-F(x_*)\right)+\frac{\left\|v_k-x_*\right\|^2}{2\gamma}, \\
\Rightarrow & F\left(x_{k+1}\right)-F\left(x_*\right)+\frac{\left\|v_{k+1}-x_*\right\|^2}{2\gamma t_k^2} \leq\left(1-\frac{1}{t_k}\right)\left(F(x_k)-F(x_*)\right)+\frac{\left\|v_k-x_*\right\|^2}{2\gamma t_k^2}, \\
\Rightarrow & t_k^2\left(F\left(x_{k+1}\right)-F\left(x_*\right)\right)+\frac{\left\|v_{k+1}-x_*\right\|^2}{2\gamma} \leq t_{k-1}^2\left(F(x_k)-F(x_*)\right)+\frac{\left\|v_k-x_*\right\|^2}{2\gamma},
\end{aligned} \tag{62}
$$

where we find the recursive inequality. In that case, we have

$$
\begin{aligned}
& t_{k-1}^2\left(F\left(x_k\right)-F\left(x_*\right)\right)+\frac{\left\|v_k-x_*\right\|^2}{2\gamma} \leq t_0^2\left(F\left(x_1\right)-F\left(x_*\right)\right)+\frac{\left\|v_1-x_*\right\|^2}{2\gamma}, \\
\Rightarrow & t_{k-1}^2\left(F\left(x_k\right)-F\left(x_*\right)\right)+\frac{\left\|v_k-x_*\right\|^2}{2\gamma} \leq \frac{\left\|v_0-x_*\right\|^2}{2\gamma}, \\
\Rightarrow & t_{k-1}^2\left(F\left(x_k\right)-F\left(x_*\right)\right) \leq \frac{\left\|y_0-x_*\right\|^2}{2\gamma}, \\
\Rightarrow & \left(F\left(x_k\right)-F\left(x_*\right)\right) \leq \frac{\left\|y_0-x_*\right\|^2}{2\gamma t_{k-1}^2},
\end{aligned} \tag{63}
$$

where gives us a convergence rate of $\mathcal{O}(\frac{1}{k^2})$. Now, we can apply this result to our dual problem by setting $F=f^*+g^*$. Since we do not change any conditions, we get the same convergence rate $\mathcal{O}(\frac{1}{k^2})$.

Part III Solving the Lasso problem

1. Implement PSGA

Here is code for PSGA:

Algorithm 1 Proximal Stochastic Gradient Algorithm (PSGA)

Input: $A \in \mathbb{R}^{n \times d}$, $y \in \mathbb{R}^n$, regularization parameter λ, max iterations max_iters
Output: Solution x, objective values, and ergodic objective values
Initialize $x \leftarrow$ zeros of size d
Compute Frobenius norm of $\mathrm{A}^2 \leftarrow \|A\|_F^2$
Initialize gamma_sum $\leftarrow 0$, weighted_sum $\leftarrow \mathbf{0}$
for $k = 1$ ***to*** *max_iters* **do**
  Compute step size: $\gamma_k \leftarrow \frac{n}{\|A\|_F^2 \sqrt{k+1}}$
  Randomly sample $i_k \leftarrow$ random integer in $[1, n]$
  Compute gradient: grad $\leftarrow (A[i_k, :] \cdot x - y[i_k]) \cdot A[i_k, :]$
  Update x: $x \leftarrow \mathrm{prox}_{\gamma_k \lambda}(x - \gamma_k \cdot \mathrm{grad})$
  Update ergodic mean: weighted_sum $\leftarrow$ weighted_sum $+ \gamma_k \cdot x$
return x, objective values, ergodic objective values

The specific code is provided in A.

2. Implement RCPGA

Here is code for RCPGA:

Algorithm 2 Randomized Coordinate Proximal Gradient Algorithm (RCPGA)

Input: $A \in \mathbb{R}^{n \times d}$, $y \in \mathbb{R}^n$, λ, max iterations max_iters
Output: Solution x, objective values objective_values
Initialization:
 Initialize $x \leftarrow$ zeros of size d
 Compute $L_j \leftarrow \frac{1}{n} \sum_{i=1}^{n} A[i, j]^2$ for all $j = 1, \ldots, d$
for $k = 1$ ***to*** *max_iters* **do**
  Randomly select coordinate: $j_k \leftarrow$ random integer in $[1, d]$
  Compute gradient for j_k:

$$\mathrm{grad}_{j_k} \leftarrow \frac{1}{n} \sum_{i=1}^{n} \big((A[i, :] \cdot x - y[i]) \cdot A[i, j_k] \big)$$

  Perform proximal update for j_k:

$$x_{j_k} \leftarrow \mathrm{sign}\left(x_{j_k} - \mathrm{grad}_{j_k}/L_{j_k}\right) \cdot \max\left(\left|x_{j_k} - \mathrm{grad}_{j_k}/L_{j_k}\right| - \lambda/L_{j_k}, 0\right)$$

  Compute objective value:

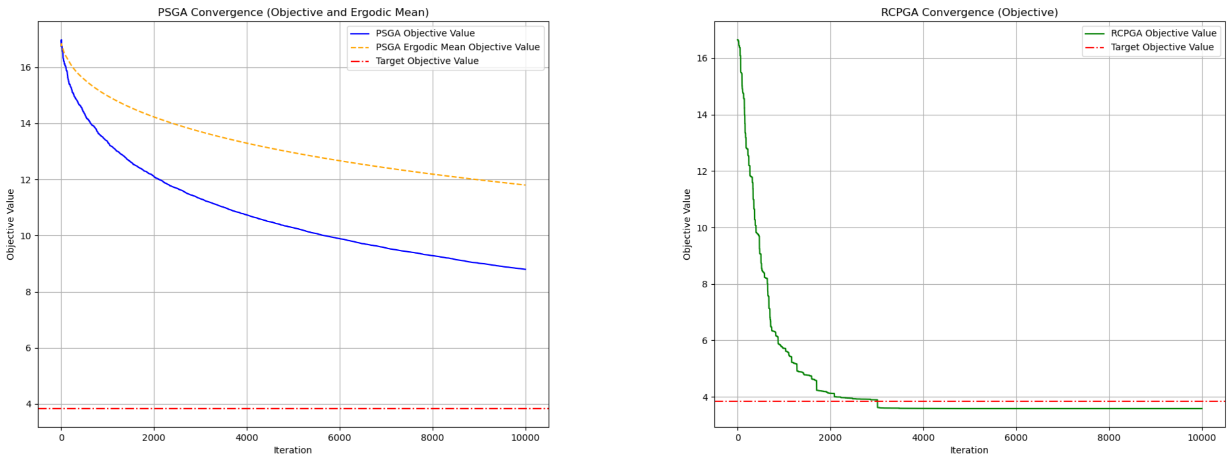
$$\mathrm{obj\_val} \leftarrow \frac{1}{2n}\|Ax - y\|^2 + \lambda\|x\|_1$$

  Append obj_val to objective_values
return x, objective_values

The specific code is provided in A.

3. Choose parameter and plot function

Here are the plots for both PSGA and RCPGA with same regularisation parameter $\lambda = 0.1$. In 10k iterations, RCPGA converges much faster than PSGA. This is likely due to the coordinate-wise updates in RCPGA, which focus on optimizing individual variables, making it more efficient for problems with sparse structure or separable objectives. The ergodic mean in PSGA serves as a smoothing mechanism but converges slower than direct iterates.



(a) PSGA Plot

(b) RCPGA Plot

Figure 2: Run PSGA and RCPGA in 10k iterations

Here is another plot for PSGA. In this plot, we take 200k iterations.

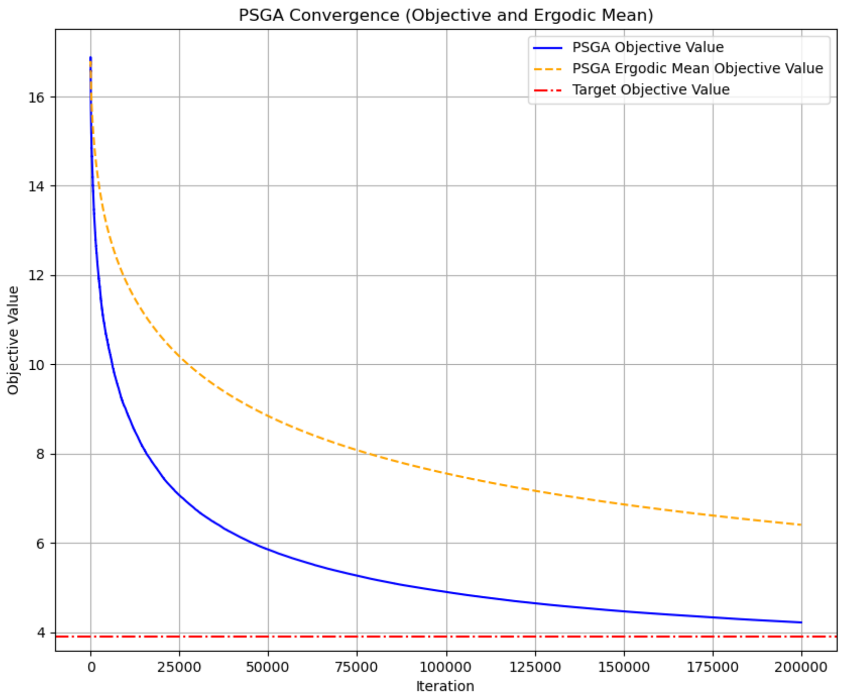


Figure 3: Run PSGA in 200k iterations

A Code for Part III

Please check the zip file for detailed code.

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

def generate_problem(n, d, s, std=0.06):
    # Generate xs
    # vectors with entries in [0.5, 1] and [-1, -0.5]
    # respectively
    assert s % 2 == 0, "s needs to be divisible by 2"
    xsp = 0.5 * (np.random.rand(s // 2) + 1)
    xsn = - 0.5 * (np.random.rand(s // 2) + 1)
    xsparse = np.hstack([xsp, xsn, np.zeros(d - s)])
    random.shuffle(xsparse)
    # Generate A
    A = np.random.randn(n, d)
    # Generate eps
    y = A @ xsparse + std * np.random.randn(n)
    return xsparse, A, y
def PSGA(A, y, lamb_da, num_iterations=100000):
    n, d = A.shape
    x = np.zeros(d)  # Here we initialize x
    L = np.linalg.norm(A, ord='fro') ** 2  # we take the Frobenius norm of A
    gamma_sum = 0
    weighted_sum = np.zeros(d)  # compute the ergodic mean

    objective_values = []  # track objective function values
    ergodic_objective_values = []  # track ergodic mean objective values

    for k in range(1, num_iterations + 1):
        # step size
        gamma_k = n / (L * np.sqrt(k + 1))
        gamma_sum += gamma_k

        # randomly sample a row
        i_k = np.random.randint(0, n)
        a_i = A[i_k,:]
        y_i = y[i_k]

        # compute gradient
        grad = ((np.dot(a_i, x) - y_i) * a_i).T

        # proximal update
        x = np.sign(x - gamma_k * grad) * np.maximum(np.abs(x - gamma_k * grad) - gamma_k * lamb_da
    , 0)

        # update the weighted sum for the ergodic mean
        weighted_sum += gamma_k * x

        # compute ergodic mean
        ergodic_x = weighted_sum / gamma_sum

        # store objective values
        obj_val = 0.5 / n * np.linalg.norm(A @ x - y) ** 2 + lamb_da * np.linalg.norm(x, 1)
        ergodic_obj_val = 0.5 / n * np.linalg.norm(A @ ergodic_x - y) ** 2 + lamb_da * np.linalg.
    norm(ergodic_x, 1)
        objective_values.append(obj_val)
        ergodic_objective_values.append(ergodic_obj_val)

    return x, objective_values, ergodic_objective_values


def RCPGA(A, y, lamb_da, num_iterations=100000):
    n, d = A.shape
    x = np.zeros(d)  # Initialize x
    Ls = np.sum(A ** 2, axis=0) / n  # Coordinate-wise Lipschitz constants

    objective_values = []  # To track objective function values

    for k in range(num_iterations):

        # Randomly select a coordinate
        j_k = np.random.randint(0, d)
```

```

        # Compute gradient for the selected coordinate
        grad_j = (1 / n) * np.sum((A @ x - y) * A[:, j_k])

        # Proximal update for the selected coordinate
        x[j_k] = np.sign(x[j_k] - grad_j / Ls[j_k]) * \
                 np.maximum(np.abs(x[j_k] - grad_j / Ls[j_k]) - lamb_da / Ls[j_k], 0)

        # Store objective values
        obj_val = 0.5 / n * np.linalg.norm(A @ x - y) ** 2 + lamb_da * np.linalg.norm(x, 1)
        objective_values.append(obj_val)

    return x, objective_values

# Parameters for generating the problem
n, d, s = 1000, 500, 50
lamb_da = 0.1
num_iterations = 10000
num_more_iterations = 190000

# Generate problem data
x_star, A, y = generate_problem(n, d, s)

# Target objective value
target_obj = 0.5/n * np.linalg.norm(x_star, 1) + lamb_da * np.linalg.norm(x_star, 1)

# Run PSGA
x_psga, obj_values_psga, ergodic_obj_values_psga = PSGA(A, y, lamb_da, num_iterations)

# Run RCPGA
x_rcpga, obj_values_rcpga = RCPGA(A, y, lamb_da, num_iterations)

# Plot PSGA results and target objective value
plt.figure(figsize=(10, 8))
plt.plot(obj_values_psga, label="PSGA Objective Value", color='blue')
plt.plot(ergodic_obj_values_psga, label="PSGA Ergodic Mean Objective Value", linestyle='--', color='orange')
plt.axhline(y=target_obj, color='red', linestyle='-.', label="Target Objective Value")
plt.xlabel("Iteration")
plt.ylabel("Objective Value")
plt.title("PSGA Convergence (Objective and Ergodic Mean)")
plt.legend()
plt.grid()
plt.savefig("psga.png")
plt.show()

# Plot RCPGA results
plt.figure(figsize=(10, 8))
plt.plot(obj_values_rcpga, label="RCPGA Objective Value", color='green')
plt.axhline(y=target_obj, color='red', linestyle='-.', label="Target Objective Value")
plt.xlabel("Iteration")
plt.ylabel("Objective Value")
plt.title("RCPGA Convergence (Objective)")
plt.legend()
plt.grid()
plt.savefig("rcpga.png")
plt.show()


# Run PSGA with more iterations
x_psga, obj_values_psga, ergodic_obj_values_psga = PSGA(A, y, lamb_da, num_iterations + num_more_iterations)
# Plot PSGA in more iterations
plt.figure(figsize=(10, 8))
plt.plot(obj_values_psga, label="PSGA Objective Value", color='blue')
plt.plot(ergodic_obj_values_psga, label="PSGA Ergodic Mean Objective Value", linestyle='--', color='orange')
plt.axhline(y=target_obj, color='red', linestyle='-.', label="Target Objective Value")
plt.xlabel("Iteration")
plt.ylabel("Objective Value")
plt.title("PSGA Convergence (Objective and Ergodic Mean)")
plt.legend()
plt.grid()
plt.savefig("psga_100k.png")
```

```
plt.show()
```

Part III Code