1. Newton’s Method, Coordinate Descent and Gradient Descent

In this question, we will compare three different optimization methods: Newton’s method, coordinate descent and gradient descent. We will consider the simple set-up of unconstrained convex quadratic optimization; i.e we will consider the following problem:

\[
\min_{\vec{x} \in \mathbb{R}^d} \vec{x}^\top A \vec{x} - 2\vec{b}^\top \vec{x} + c
\]

where \( A > 0 \) and \( \vec{b} \in \mathbb{R}^d \).

(a) How many steps does Newton’s method take to converge to the optimal solution? Recall that the update rule for Newton’s method is given by the equation:

\[
\vec{x}_{t+1} = \vec{x}_t - (\nabla^2 f(\vec{x}_t))^{-1}\nabla f(\vec{x}_t).
\]

when optimizing a function \( f \).

(b) Now, consider the simple two variable quadratic optimization problem for \( \sigma > 0 \):

\[
\min_{\vec{x} \in \mathbb{R}^2} f(\vec{x}) = \sigma x_1^2 + x_2^2.
\]

How many steps does coordinate descent take to converge on this problem? Assume that we start by updating the variable \( x_1 \) in the first step, \( x_2 \) in step two and so on; therefore, we will update \( x_1 \) and \( x_2 \) in odd and even iterations respectively:

\[
(x_{t+1})_1 = \begin{cases} 
\arg\min_{x_1} f(x_1, (x_t)_2) & \text{for odd } t \\
(x_t)_1 & \text{otherwise}
\end{cases} \quad \text{and} \quad (x_{t+1})_2 = \begin{cases} 
\arg\min_{x_2} f((x_t)_1, x_2) & \text{for even } t \\
(x_t)_2 & \text{otherwise}
\end{cases}
\]

Here, \((x_t)_2\) represents \( x_2 \) at time \( t \) and so on.
(c) We will now analyze the performance of coordinate descent on another quadratic optimization problem:

$$\min_{\vec{x} \in \mathbb{R}^2} f(\vec{x}) = \sigma (x_1 + x_2)^2 + (x_1 - x_2)^2.$$ 

where we have, as before, $\sigma > 0$. Note that $(0, 0)$ is the optimal solution to this problem. Now, starting from the point $(1, 1)$, how many steps does coordinate descent take to converge to $(0, 0)$. What happens when $\sigma$ grows large? Hint: First find the update rule for $x_1$, i.e. keep $x_2$ fixed and figure out how $x_1$ changes when $t$ is odd. Then do the same for $x_2$ when $x_1$ is fixed.

(d) Finally, for the objective function from the previous part, how long does gradient descent take to converge to $(0, 0)$ starting from the point $(1, -1)$? Assume for this part that $\sigma > 1$ and reason about how many steps it takes for gradient descent to converge when $\sigma$ grows large. Hint: What is the step size for gradient descent? Also note that $f$ is given by:

$$f(\vec{x}) = \vec{x}^T A \vec{x}$$

where $A = 2 \left( \sigma \begin{bmatrix} \frac{\sqrt{2}}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{\sqrt{2}}{\sqrt{2}} \end{bmatrix} + \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} \right).$

2. Gradient Descent vs Newton Method

Run the jupyter-notebook ‘Gradient_vs_Newton.ipynb’ which demonstrates differences between gradient descent and Newton’s method.

3. (Optional) Ridge Regression Classifier Vs. SVM

In this problem, we explore Ridge Regression as a classifier, and compare it to SVM. Recall Ridge Regression solves the problem

$$\min_{\vec{w}} \|X \vec{w} - \vec{y}\|_2^2 + \lambda \|\vec{w}\|_2^2,$$

where $X \in \mathbb{R}^{m \times n}$, and $\vec{y} \in \mathbb{R}^n$

(a) Ridge Regression as is solves a regression problem. Given data $X \in \mathbb{R}^{m \times n}$ and labels $\vec{y} \in \{0, 1\}^m$, explain how we might be able to train a Ridge Regression model and use it to classify a test point.

(b) Complete the accompanying Jupyter Notebook to compare Ridge Regression and SVM.

4. Soft-margin SVM

Consider the soft-margin SVM problem,

$$p^*(C) = \min_{\vec{w} \in \mathbb{R}^m, b \in \mathbb{R}, \xi \in \mathbb{R}^n} \frac{1}{2} \|\vec{w}\|_2^2 + C \sum_{i=1}^n \xi_i$$

s.t. $1 - \xi_i - y_i(\vec{x}_i^T \vec{w} - b) \leq 0, \quad i = 1, 2, \ldots, n$

$$- \xi_i \leq 0, \quad i = 1, 2, \ldots, n,$$

where $\vec{x}_i \in \mathbb{R}^m$ refers to the $i^{th}$ training data point, $y_i \in \{-1, 1\}$ is its label, and $C \in \mathbb{R}_+$ (i.e. $C > 0$) is a hyperparameter.
Let \( \alpha_i \) denote the dual variable corresponding to the inequality \( 1 - \xi_i - y_i(x_i^\top \bar{w} - b) \leq 0 \) and let \( \beta_i \) denote the dual variable corresponding to the inequality \( -\xi_i \leq 0 \).

The Lagrangian is then given by

\[
L(\bar{w}, b, \xi, \alpha, \beta) = \frac{1}{2} \|\bar{w}\|_2^2 + C \sum_{i=1}^{n} \xi_i + \sum_{i=1}^{n} \alpha_i(1 - \xi_i - y_i(x_i^\top \bar{w} - b)) - \sum_{i=1}^{n} \beta_i \xi_i.
\]

Suppose \( \bar{w}^*, b^*, \xi^*, \alpha^*, \beta^* \) satisfy the KKT conditions.

Classify the following statements as true or false and justify your answers mathematically.

(a) Suppose the optimal solution \( \bar{w}^*, b^* \) changes when the training point \( x_i \) is removed. Then originally, we necessarily have \( y_i(x_i^\top \bar{w}^* - b^*) = 1 - \xi_i^* \).

(b) Suppose the optimal solution \( \bar{w}^*, b^* \) changes when the training point \( x_i \) is removed. Then originally, we necessarily have \( \alpha_i^* > 0 \).

(c) Suppose the data points are strictly linearly separable, i.e. there exist \( \bar{w} \) and \( \bar{b} \) such that for all \( i \),

\[
y_i(x_i^\top \bar{w} - \bar{b}) > 0.
\]

Then \( p^*(C) \to \infty \) as \( C \to \infty \).

5. Linear Quadratic Regulator

In this question, we will derive the Riccati equation for the LQR model studied in class. We first recall the statement of the LQR problem:

\[
\begin{align*}
\min_{\bar{x}_t, \bar{u}_t} & \sum_{t=0}^{N-1} \frac{1}{2} \left( \bar{x}_t^\top Q \bar{x}_t + \bar{u}_t^\top R \bar{u}_t \right) + \frac{1}{2} \bar{x}_N^\top Q \bar{x}_N \\
\text{s.t} & \quad \bar{x}_{t+1} = A \bar{x}_t + B \bar{u}_t \\
\text{and} & \quad \bar{x}_0 = \bar{x}_{\text{init}}
\end{align*}
\]

where \( \bar{x}_t \) is thought of as the state of the system and \( \bar{u}_t \) is the control input at time \( t \) and the matrices \( A \) and \( B \) define the dynamics of the system. While the problem can be solved as a quadratic program, we will now take a slightly different approach. We start by defining the functions, \( J_k \) for \( 0 \leq k \leq N \), as follows:

\[
J_k(\bar{x}) = \min_{\{\bar{u}_t\}_{t=k}^{N-1}} \sum_{t=k}^{N-1} \frac{1}{2} \left( \bar{x}_t^\top Q \bar{x}_t + \bar{u}_t^\top R \bar{u}_t \right) + \frac{1}{2} \bar{x}_N^\top Q \bar{x}_N \\
\text{s.t} & \quad \bar{x}_{t+1} = A \bar{x}_t + B \bar{u}_t \\
& \quad \bar{x}_k = \bar{x}.
\]

\( J_k \) can be thought of as the minimum cost that we would incur from time \( k \) assuming that we start at state \( \bar{x}_k = \bar{x} \). We can now decompose \( J_k \) for \( 0 \leq k \leq N - 1 \) further as follows:
\[ J_k(\vec{x}) = \min_{\vec{u}_k} \frac{1}{2} \left( \vec{x}_k^T Q \vec{x}_k + \vec{u}_k^T R \vec{u}_k \right) + \min_{\{\vec{u}_t\}_{t=k+1}^{N-1}} \sum_{t=k+1}^{N-1} \frac{1}{2} \left( \vec{x}_t^T Q \vec{x}_t + \vec{u}_t^T R \vec{u}_t \right) + \frac{1}{2} \vec{x}_N^T Q \vec{x}_N \\
\text{s.t } \vec{x}_{t+1} = A \vec{x}_t + B \vec{u}_t \\
\vec{x}_k = \vec{x}.
\]

Note that in particular, the first constraint implies that \( \vec{x}_{k+1} = A \vec{x}_k + B \vec{u}_k \). Therefore, the above characterization gives the following decomposition:

\[ J_k(\vec{x}) = \min_{\vec{u}} \frac{1}{2} \left( \vec{x}^T Q \vec{x} + \vec{u}^T R \vec{u} \right) + J_{k+1}(A \vec{x} + B \vec{u}). \tag{2} \]

We will see that the functions, \( J_k \), are all in fact quadratic functions in \( \vec{x} \) and this will give us convenient ways to derive the optimal control inputs at each time.

(a) First, we will show by reverse induction that each of the functions \( J_k \) for \( 0 \leq k \leq N \) are convex quadratics. In particular, prove that \( J_k(\vec{x}) = \vec{x}^T Q_k \vec{x} \) for some \( Q_k \succ 0 \) and determine the value of \( Q_k \) in terms of \( Q_{k+1} \).

Hint 1: \( J_N(\vec{x}) = \frac{1}{2} \vec{x}^T Q \vec{x} \). Therefore, \( Q_N = Q \). Also can use (2) above, and substitute \( J_{k+1}(A \vec{x} + B \vec{u}) = (A \vec{x} + B \vec{u})^T Q_{k+1}(A \vec{x} + B \vec{u}) \). Then solve the resulting QP to find the optimal \( \vec{u} \).

Hint 2: You should get the following recursion for \( Q_k \):

\[ Q_k = Q + A^T Q_{k+1} A - A^T Q_{k+1} B (R + B^T Q_{k+1} B)^{-1} B^T Q_{k+1} A. \]

(b) (Optional) Now, show that the expression for \( Q_l \) is equivalent for the expression obtained by using the Lagrangian. That is, show that \( Q_l \) from the previous part is the same as:

\[ Q_l = Q + A^T (Q_{l+1}^{-1} + BR^{-1} B^T)^{-1} A. \]

You may find useful the Sherman-Morrison-Woodbury matrix identity:

\[ (M + UWV)^{-1} = M^{-1} - M^{-1}U(W^{-1} + VM^{-1}U)VM^{-1}. \]

6. Homework process

Whom did you work with on this homework? List the names and SIDs of your group members.