ORF307_HW5

March 20, 2023

ORF307 Homework 5

Due: Friday, March 31, 2023 9:00 pm ET

- Please export your code with output as pdf.
- If there is any additional answers, please combine them as **ONE** pdf file before submitting to the Gradescope.

Q1 Finding a direction in simplex

Let $P = \{x \in \mathbf{R}^3 \mid 2x_1 + 3x_2 + x_3 = 1, x \geq 0\}$ and consider the vector $x = (0, 0, 1)$. Let the cost vector be $c = (1, -1, 2)$. Find the set of basic directions at x that improve the cost. It is a minimization problem, so we want to decrease the cost.

Q2 Extreme points and basic feasible solutions

Let $P = \{x \in \mathbf{R}^5 \mid Ax = b, x \geq 0\}$ where

$$A = \begin{bmatrix} 0 & 1 & 1 & 1 & -2 \\ 0 & -1 & 1 & -1 & 0 \\ 2 & 0 & 1 & 0 & -1 \end{bmatrix}, \quad \text{and} \quad b = (1, 1, 1).$$

(a) Given the following 3 vectors

i. $\hat{x} = (0, 0, 1, 0, 0)$

ii. $\hat{x} = (0, 0, 1, 1, 1)$

iii. $\hat{x} = (0, 0, 0, -1, -1)$

list which ones are in P, which ones are basic solutions, and which ones are degenerate. Please explain.

(b) If they are basic feasible solutions are they extreme points? If yes, give a vector c for which $\hat{x}$ is the unique solution of

$$\begin{array}{ll} \text{minimize} & c^T x \\ \text{subject to} & Ax = b \\ & x \geq 0 \end{array}$$

[4]:
```
'''
This code is provided to help with questions 3 and 4
```

```python
'''

import numpy as np
import numpy.linalg as la

def simplex_iteration(x, B, problem):
    """Perform one simplex iteration given
    - basic feasible solution x
    - basis B

    It returns new x, new basis and termination flag (true/false)
    """
    A, b, c = problem['A'], problem['b'], problem['c']
    m, n = A.shape
    A_B = A[:, B]

    # Compute reduced cost vector
    p = la.solve(A_B.T, c[B])
    c_bar = c - A.T @ p

    # Check optimality
    if np.all(c_bar >= 0):
        print("Optimal solution found!")
        return x, B, True

    # Choose j such that c_bar < 0 (first one)
    j = np.where(c_bar < 0)[0][0]

    # Compute search direction d
    d = np.zeros(n)
    d[j] = 1
    d[B] = la.solve(A_B, -A[:, j])

    # Check for unboundedness
    if np.all(d >= 0):
        print("Unbounded problem!")
        return None, None, True

    # Compute step length theta
    d_i = np.where(d[B] < 0)[0]
    theta = np.min(- x[B[d_i]] / d[B[d_i]])
    i = B[d_i[np.argmin(- x[B[d_i]] / d[B[d_i]])]]

    # Compute next point
    x_next = x + theta * d

    # Compute next basis
```

```python
    B_next = B
    B_next[np.where(B == i)[0]] = j

    return x_next, B_next, False

def simplex_algorithm(x, B, problem, max_iter=1000):
    """Run simplex algorithm"""

    for k in range(max_iter):

        x, B, end = simplex_iteration(x, B, problem)

        if end:
            break
    return x, B
```

Q3 Simplex iterations

Solve the following optimization problem using the simplex method. At each iteration, record the indices in the basis, x, the reduced costs $\bar{c}$, the new feasible direction d, and step length $\theta^\star$. Any method to come up with a starting basic feasible solution is ok.

$$\begin{array}{ll}\text{minimize} & 8x_1 + 8x_2 + 5x_3 + 9x_4 \\ \text{subject to} & 2x_1 + x_2 + x_3 + 3x_4 = 5 \\ & x_1 + 3x_2 + x_3 + 2x_4 = 3 \\ & x \geq 0\end{array}$$