MATH 173B, HW1

Victor Pekkari — epekkari@ucsd.edu

January 23, 2025

Problem 1

A function f (below) is strongly convex iff:

$$f : \Omega \to \mathbb{R}$$
$$\nabla^2 f(x) \succeq \mu I \quad \text{where: } \exists \mu \in \mathbb{R} \quad \text{s.t. } \mu > 0$$

(a)

$$\nabla^2 f(x) = 30 \cdot x^4 \geq 0 \quad \forall x \in \mathbb{R}$$
$$\nabla^2 f(0) = 0$$

Answer: Not Strongly Convex

(b)

$$\nabla^2 f(x) = \frac{e^x + e^{2x}}{1 + e^x} + 2 > 0 \quad \forall x \in \mathbb{R}$$

Answer: Strongly convex

(c)

$$\nabla^2 f(x) = 2e^{x^2} + 4x^2 \cdot e^{x^2} \geq 0 \quad \forall x \in \mathbb{R}$$

Answer: Strongly Convex

(d)

$$\nabla^2 f(x) = \begin{bmatrix} 2 & 2 \\ 2 & 2 \end{bmatrix} \implies \lambda_1 = 4, \lambda_2 = 0$$

Answer: Not Strongly Convex

Problem 2

(a)

$$f(\vec{x}) = x_1^2 + x_2^2 + x_3^2, \quad \nabla^2 f(\vec{x}) = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad \lambda_{min}(\nabla^2 f(\vec{x})) = 0$$

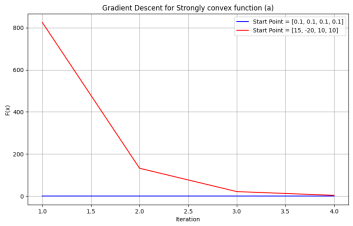
(b)

$$f(\vec{x}) = x_1^2 + x_2^2 + x_3^2 + x_4^2, \quad \nabla^2 f(\vec{x}) = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 2 \end{bmatrix}, \quad \lambda_i = 2 > 0 \quad \forall i \in [1, 2, 3, 4]$$

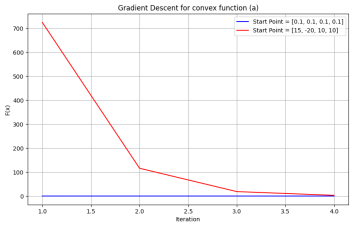
Problem 3

Conclusion: Strong convexity ensures stability which in turn gives us more freedom when choosing the learning rate (had the same lr for gd of the strongly cvx and just cvx this time though).

Strong convexity is furthermore the only thing that ensures exponential convergence, which then makes it reasonable to see that the strongly convex function converges faster.



Convex function (2.a)



Strongly Convex function (2.b)

Code:

```python
import numpy as np
import matplotlib.pyplot as plt
from typing import Callable

start1 = np.array([0.1, 0.1, 0.1, 0.1], dtype=float)
start2 = np.array([15, -20, 10, 10], dtype=float)
start_points = [start1, start2]

def cvx(x: np.ndarray) -> float:
    return x[0]**2 + x[1]**2 + x[2]**2

def strongly_cvx(x: np.ndarray) -> float:
    return x[0]**2 + x[1]**2 + x[2]**2 + x[3]**2

def gradient_cvx(x: np.ndarray) -> np.ndarray:
    return np.array([2*x[0], 2*x[1], 2*x[2], 0], dtype=float)

def gradient_strongly_cvx(x: np.ndarray) -> np.ndarray:
    return np.array([2*x[0], 2*x[1], 2*x[2], 2*x[3]], dtype=float)

def gradient_descent(function: Callable[[np.ndarray], float], gradient: Callable[[np.ndarra
    lr = 3e-1
    iteration_nbr = 5
    iterationList = []
    yList = []
    for start_point in start_points:
        x = start_point.astype(float)
        yList.append(function(x))
        iterationList.append(1)
        for i in range(2, iteration_nbr):
            iterationList.append(i)
            x -= lr * gradient(x)
            yList.append(function(x))
    length = len(yList) // 2
    iterationList = iterationList[:length]
    list1 = yList[:length]
    list2 = yList[length:]

    # Plotting list1 and list2
    plt.figure(figsize=(10, 6))
    plt.plot(iterationList, list1, label='Start Point = [0.1, 0.1, 0.1, 0.1]', color='blue'
    plt.plot(iterationList, list2, label='Start Point = [15, -20, 10, 10]', color='red')
    plt.xlabel('Iteration')
```

```
    plt.ylabel('F(x)')
    plt.title('Gradient Descent for Strongly convex function (a)')
    plt.legend()
    plt.grid(True)
    plt.savefig('strong_cvx2.png')
    plt.show()

# Example usage
#gradient_descent(cvx, gradient_cvx)
gradient_descent(strongly_cvx, gradient_strongly_cvx)


# For cvx function, you need to use 4-dimensional start points
# start1 = np.array([0.1, 0.1, 0.1, 0.1], dtype=float)
# start2 = np.array([15, -20, 10, 10], dtype=float)
# start_points = [start1, start2]
# gradient_descent(cvx, gradient_cvx)
```

Problem 4

Coin=c, Heads=h, Tails=t

(a)

Answer: $\Omega = \{[c_1 = h, c_2 = h, c_3 = h], [c_1 = h, c_2 = h, c_3 = t], [c_1 = h, c_2 = t, c_3 = h], [c_1 = h, c_2 = t, c_3 = t], [c_1 = t, c_2 = h, c_3 = h], [c_1 = t, c_2 = h, c_3 = t], [c_1 = t, c_2 = t, c_3 = h], [c_1 = t, c_2 = t, c_3 = t]\}$

(b)

Answer: The event space is the powerset of the sample space; the set of all possible subsets of the sample space

(c)

Answer: $P(event) = \frac{1}{8} \quad \forall event \in \Omega$

(d)

$$X : \Omega \to \mathbb{N}$$

Answer: It takes an element from the sample space ($\omega \in \Omega$) and outputs how many of the coins show heads.

(e)

$\mathbf{p_X(0) = \frac{1}{8}}$

one of our eight events in the sample space fulfills this, when none of the coins show heads.

$\mathbf{p_X(1) = \frac{3}{8}}$

Three of our eight events in the sample space fulfills this, when only coin1, coin2 or coin3 show heads.

$\mathbf{p_X(2) = \frac{3}{8}}$

Three of our eight events in the sample space fulfills this, when only coin1, coin2 or coin3 show tails.

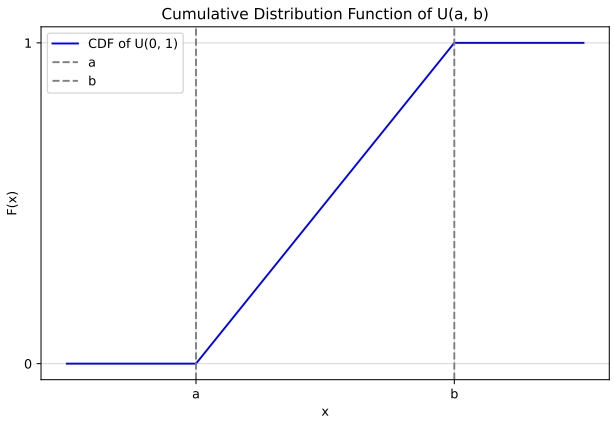
$\mathbf{p_X(3) = \frac{1}{8}}$

One of our eight events in the sample space fulfills this, when all our coins show heads.

Problem 5

Answer:

$$F_X(x) = \begin{cases} 0, & \text{if } x < a, \\ \frac{x-a}{b-a}, & \text{if } a \leq x \leq b, \\ 1, & \text{if } x > b. \end{cases}$$



(b)

Answer: $F_X(\frac{a+b}{2}) = \frac{1}{2}$

(c)

$$f_X(x) = \begin{cases} 1, & \text{for } 0 \leq x \leq 1, \\ 0, & \text{else} \end{cases}$$

$$P(x \in [\frac{1}{3}, \frac{1}{2}]) = \frac{1}{2} - \frac{1}{3} = \frac{1}{6}$$

$$P(x \in [\frac{4}{5}, \frac{5}{6}]) = \frac{5}{6} - \frac{4}{5} = \frac{1}{30}$$

$$P(x \in [\frac{1}{3}, \frac{1}{2}] \cup [\frac{4}{5}, \frac{5}{6}]) = P(x \in [\frac{1}{3}, \frac{1}{2}]) + P(x \in [\frac{4}{5}, \frac{5}{6}])$$

Answer: $P = \frac{1}{5}$.