A MATRIX GAME

BY LAVLIN, MADHAV AND VARSHITA

UNDERSTANDING THE GAME

- A Two player Zero -Sum Game
- PAY-OFF MATRIX : *A matrix A=(aij) where aij shows the payoff where player one makes move i and player two makes move j.*
- STOCHASTIC VECTOR : *Simply a vector with entries showing the probabilities of an option.*
- ***Given the pay-off matrix how would you pick a unique stochastic vector(fixed) that maximize your winnings ?***

AN IDEAL EXAMPLE

- **Rock-Paper-Scissors**
  - ***The player has 3 options namely ROCK,PAPER and SCISSORS***
  - **RULES: a)Rock beats Scissors**
    b)Scissors beats Paper
    c)Paper beats Rock

- stochastic vector

$$[\ x1 \quad x2 \quad x3\ ]$$

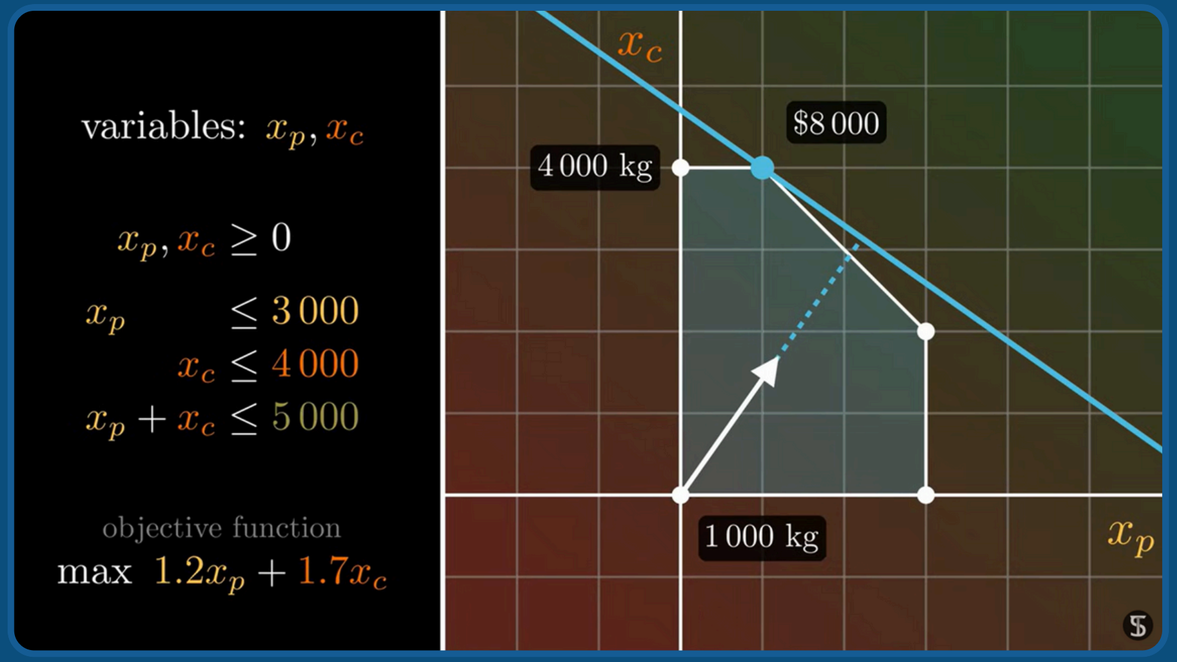
$$\begin{bmatrix} 0 & -1 & 1 \\ 1 & 0 & -1 \\ -1 & 1 & 0 \end{bmatrix}$$

- PAY-OFF Matrix

LINEAR PROGRAMMING

- Maximize **Objective Function**
- **Figure out Decision Variables**
- Given **Linear Inequalities**

- Graphical Method
- Simplex Method
- Always Optimal (Convex)



Expected Payoff $\Rightarrow \sum_{j=1}^{m} \sum_{i=1}^{n} y_i A_{ij} x_j$ $\left[\begin{array}{c} y A x^T \\ \text{or} \\ y^T A x \end{array} \right]$

Given $\begin{bmatrix} 0 & -1 & 2 \\ 3 & 0 & -4 \\ -5 & 6 & 0 \end{bmatrix}$

We assume $\begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix}$

Opponent then takes $\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}$

$$\begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix}$$

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} \begin{bmatrix} 0 & -1 & 2 \\ 3 & 0 & -4 \\ -5 & b & 0 \end{bmatrix}$$

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} \begin{bmatrix} 0x_1 - 1x_2 + 2x_3 \\ 3x_1 + 0x_2 - 4x_3 \\ -5x_1 + bx_2 + 0x_3 \end{bmatrix} \begin{matrix} a \\ b \\ c \end{matrix}$$

$$\begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix}$$

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} \begin{bmatrix} 0 & -1 & 2 \\ 3 & 0 & -4 \\ -5 & b & 0 \end{bmatrix}$$

$$\begin{matrix} 0 \\ 0 \\ 1 \end{matrix} \leftarrow \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} \begin{bmatrix} 0x_1 - 1x_2 + 2x_3 \\ 3x_1 + 0x_2 - 4x_3 \\ -5x_1 + bx_2 + 0x_3 \end{bmatrix} \begin{matrix} a \\ b \\ c \end{matrix}$$

$$\begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix}$$

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} \begin{bmatrix} 0 & -1 & 2 \\ 3 & 0 & -4 \\ -5 & b & 0 \end{bmatrix}$$

$$\begin{matrix} 0 \\ 0 \\ 1 \end{matrix} \leftarrow \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} \begin{bmatrix} 0x_1 - 1x_2 + 2x_3 \\ 3x_1 + 0x_2 - 4x_3 \\ -5x_1 + bx_2 + 0x_3 \end{bmatrix} \begin{matrix} a \\ b \\ c \end{matrix}$$

Payoff = $\min(a, b, c)$

We wish to maximise payoff. Basically,

maximise ω s.t.

- $\omega \leq 0x_1 - 1x_2 + 2x_3$
- $\omega \leq 3x_1 + 0x_2 - 4x_3$
- $\omega \leq -5x_1 + bx_2 + 0x^3$
- & $x_1 + x_2 + x_3 = 1$

CODE

```
from pulp import *
import pickle
```

modules

```
def read_matrix(filename):
    output = []
    with open(filename, 'rb') as file:
        while True:
            try:
                f = pickle.load(file)
                output.append(f)
            except:
                break
    return output
```

matrices

transpose

```
all_mat = read_matrix("mybinary.dat")
for A in all_mat:
    AT=[[A[0][i],A[1][i],A[2][i]] for i in range(len(A))]
```

model

```
x = []              #values of x
y = []              #values of y
n = len(A[0])
#for x
model_x = LpProblem("MatrixGame", sense=LpMaximize)
#for y
model_y = LpProblem("MatrixGame", sense=LpMaximize)
```

```
# Decision variables
for i in range(n):
    #for x
    x.append(i)
    x[i] = LpVariable("x_"+str(i), 0, None, LpContinuous)
    #for y
    y.append(i)
    y[i] = LpVariable("y_"+str(i), 0, None, LpContinuous)

w = LpVariable("Payoff", None, None, LpContinuous)

# Constraints
for i in range(len(A)):
    #for x
    model_x += LpAffineExpression([(x[j],A[i][j]) for j in range(n)]) >= w
    #for y
    model_y += LpAffineExpression([(y[j],AT[i][j]) for j in range(n)]) >= w

#for x
model_x += LpAffineExpression([(x[j],1) for j in range(n)]) == 1
#for y
model_y += LpAffineExpression([(y[j],1) for j in range(n)]) == 1

# Objective function
model_x += w
model_y += w
```

constraints

solve

```
# Solve the model, and silence the logging
model_x.solve(PULP_CBC_CMD(msg=False))
model_y.solve(PULP_CBC_CMD(msg=False))
```

output

```
#output of solution
print("MATRIX:")
f = [print(i) for i in A]
print("")
for v in model_x.variables():
    print(v.name, "=", v.varValue)
print("")
for v in model_y.variables():
    print(v.name, "=", v.varValue)
print("----------------------------------------")
print("")
```

OUTPUT

```
MATRIX:
[0, -1, 2]
[3, 0, -4]
[-5, 6, 0]

Payoff = 0.15686275
x_0 = 0.39215686
x_1 = 0.35294118
x_2 = 0.25490196

Payoff = 0.15686275
y_0 = 0.60784314
y_1 = 0.26470588
y_2 = 0.12745098
```

$$\begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix}$$

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} \begin{bmatrix} 0 & -1 & 2 \\ 3 & 0 & -4 \\ -5 & b & 0 \end{bmatrix}$$

$$\begin{matrix} 0 \\ 0 \\ 1 \end{matrix} \leftarrow \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} \begin{bmatrix} 0x_1 - 1x_2 + 2x_3 \\ 3x_1 + 0x_2 - 4x_3 \\ -5x_1 + bx_2 + 0x_3 \end{bmatrix} \begin{matrix} a \\ b \\ c \end{matrix}$$

Payoff = $\min(a, b, c)$

We wish to maximise payoff. Basically,

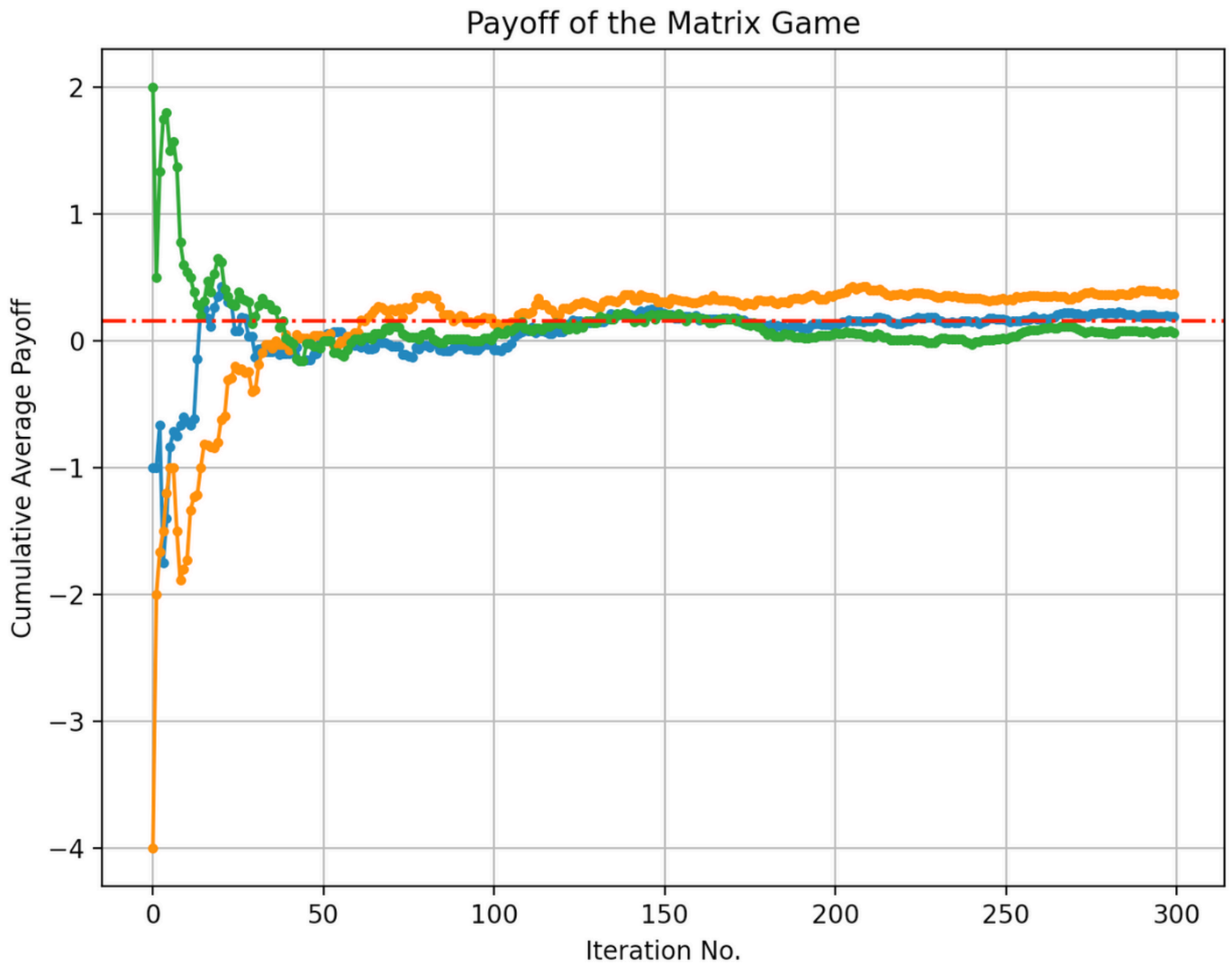
maximise ω s.t.

- $\omega \leq 0x_1 - 1x_2 + 2x_3$
- $\omega \leq 3x_1 + 0x_2 - 4x_3$
- $\omega \leq -5x_1 + bx_2 + 0x^3$
- & $x_1 + x_2 + x_3 = 1$

indept

```
payoff avg =  0.1568 , opponent stategy: [0.60784314 0.26470588 0.12745098]
payoff avg =  0.1568 , opponent stategy: [0.25 0.5  0.25]
payoff avg =  0.1568 , opponent stategy: [1 0 0]
```

GRAPH


Payoff of the Matrix Game
Cumulative Average Payoff
Iteration No.
2
1
0
−1
−2
−3
−4
0
50
100
150
200
250
300

THANK YOU VERY MUCH!