Université Jean Monnet Saint-Etienne
Faculté des sciences et techniques

Optimization and Operational Research

Aninda Maulik, Poulomi Nandy

22 March 2020

1 Example:Using CVXPY

CVXPY is a Python-embedded modeling language for convex optimization problems. It automatically transforms the problem into standard form, calls a solver, and unpacks the results. An example of a simple optimization problem solved using CVXPY in Python is given below:

```
import cvxpy as cp
# Create two scalar optimization variables.
x = cp.Variable()
y = cp.Variable()
# Create two constraints.
constraints = [x + y == 1,
x - y >= 1]
# Form objective.
obj = cp.Minimize((x - y)**2)
# Form and solve problem.
prob = cp.Problem(obj, constraints)
prob.solve() # Returns the optimal value.
print("status:", prob.status)
print("optimal value", prob.value)
print("optimal var", x.value, y.value)
```

2 Problem:Unconstrained Optimization

We start with 3 unconstrained optimization problems that are easy to formulate to learn the basic features of CVXPY.

2.1 Problem:1

```
import cvxpy as cp

# Create two scalar optimization variables.
x1 = cp.Variable()
x2 = cp.Variable()

# Form objective.
obj = cp.Minimize((x1-4)**2+7*(x2-4)**2+4*x2)
# Form and solve problem.
prob = cp.Problem(obj)
prob.solve() # Returns the optimal value.
print("status:", prob.status)
print("optimal value", prob.value)
print("optimal var", x1.value, x2.value)
#Solution
#status: optimal
#optimal value 15.428571428571429
#optimal var 4.0 3.7142857142857144
```

Solution explanation

$$f(x_1, x_2) = (x_1–4)^2 + 7(x_2–4) + 4x_2 \tag{1}$$

$$f'(x_1, x_2) = 2(x_1 - 4), 14(x_2 - 4) + 4 \tag{2}$$

To find local minimum values, we assign these partial derivatives to zero,
$2\ (\ x_1 - 4) = 0$,
$14(x_2 - 4) + 4 = 0$
$x_1 = 4$,
$x_2 = 26/7 = 3.71428571$
These Are Local Minimal Values Of $f(x_1, x_2)$ *function.*

Hessian Matrix of this f $(\ x_1, x_2)$*function*,
$H = [2, 0], [0, 14]$
Determinant of H *is*, $det(H) = 28 - 0 \geq 0$
$Traceof His, tr(H) = 2 + 14 \geq 0$

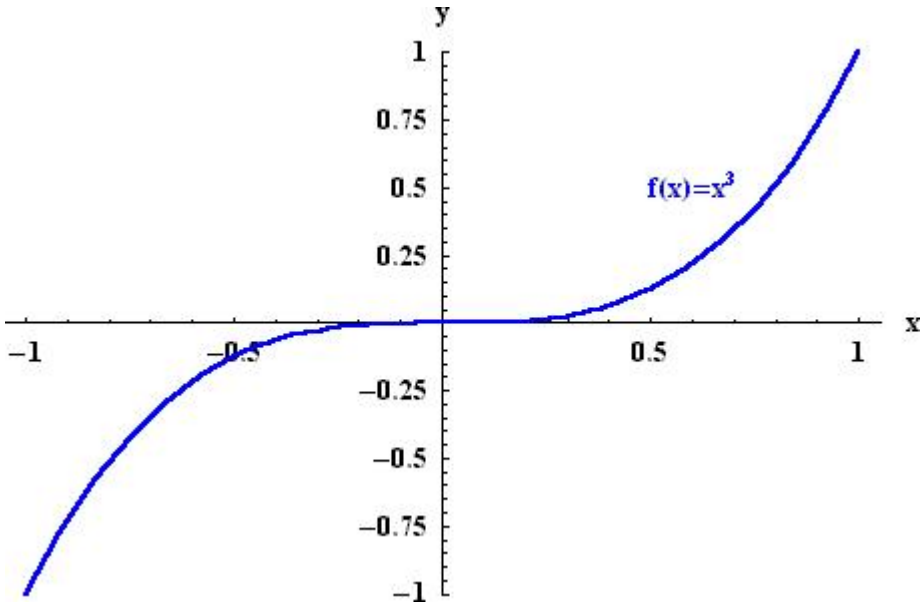
Since determinant and trace are greater than 0 this function is convex. Since the function is convex, global minimum is $x_1 = 4$ *and* $x_2 = 3.71428571$.

This is exactly what we got in Python program.

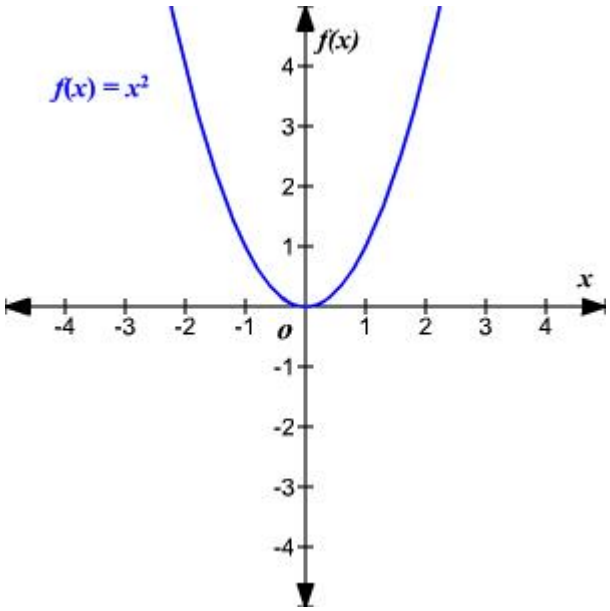
2.2 Problem 2

This is convex function, when $x_1 \geq 0$.*Because*,

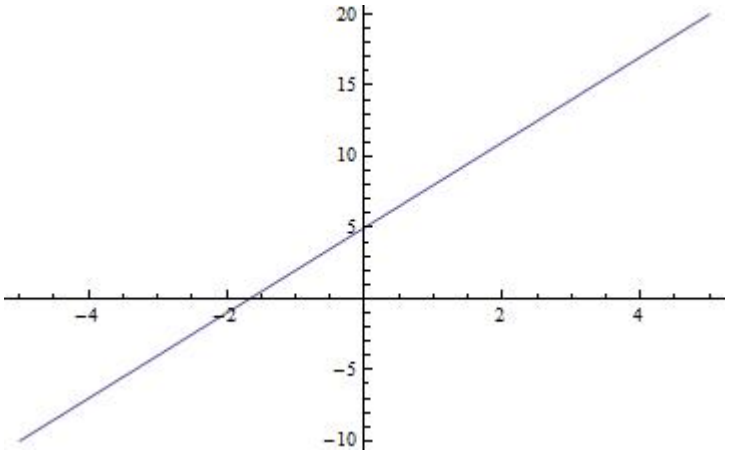
1. $f(x_1) = x_1^3$, *is convex when* $x_1 \geq 0$



2. $f(x_2, x_3) = (x_2 - x_3)^2$ *is convex because* $f(x) = x^2$, *is convex*



3. f $(x_3) = x_3 + 2$ *is convex, since a linear function is always convex*



Summation of convex functions is convex. Because of that, function
f $(x_1, x_2, x_3) = x_1^3 + (x_2 - x_3)^2 + x_3^3 + 2$ *is convex when* $x_1 \geq 0$
Analytical solution
f($x_1, x_2, x_3) = x_1^3 + (x_2 - x_3)^2 + x_3^3 + 2$
f($x_1, x_2, x_3) = x_1^3 + x_2^2 + x_3^2 - 2x_2x_3 + x_3^3 + 2$
$f' -> 3x_1^2, 2x_2 - 2x_3, 2x_3 - 2x_2 + 3x_3^2$
$H \mapsto [[6x_1, 0, 0], [0, 2, 0], [0, 0, 2 - 6x_3]]$
Determinant$_1(H) = 6x_1$
$Determinant_2(H) = 12x_1$
$Determinant_3(H) = 24x_1 - 72x_1x_3$
$f(x_1, x_2, x_3)$*to be convex, when all* 3 *determinants should be greater than or equals to* 0.

Moreover, f'$(x_1) = 0, f'(x_2) = 0, f'(x_3) = 0$; *in order to get the local minima* $f'(x_1) = 3x_1^2 = 0$ *and so* $x_1 = 0$
Now $f'(x_2) = 2x_2 - 2x_3 = 0$ *and so* $x_2 = x_3$
Finally $f'(x_3) = 2x_3 - 2x_2 + 3x_3^2 = 0$ *and substituting* $x_2 = x_3$, *we get* $2x_3 + 3x_3^2 - 2x_3 = 0$ *thus* $x_3 = 0$ *therefore,* x_2 *is also equals to* 0. *Thus our point of local minima is* $(0, 0, 0)$.
Substituting $(0, 0, 0)$ *in our equation, we get* 2; *and that's what we get in our Python*

```
import cvxpy as cp

# Create two scalar optimization variables.
x1 = cp.Variable()
x2 = cp.Variable()
x3 = cp.Variable()


obj = cp.Minimize(x1**3+(x2-x3)**2+x3**3+2)
```

```
prob = cp.Problem(obj)
prob.solve()
print("status:", prob.status)
print("optimal value", prob.value)
print("optimal var", x1.value, x2.value, x3.value)

#status: optimal
#optimal value 1.9999999993629338
#optimal var 0.0008258815459611968 0.0008258861793499652 0.0008258861755015717
```

2.3 Problem 3

```
import cvxpy as cp

# Create two scalar optimization variables.
x1 = cp.Variable()
x2 = cp.Variable()
t =100
constraints=[-x1 - x2 + 4 <= 0]

obj = cp.Minimize ((x1 - 2)**2 + 3 * x2 - (1/t)* cp.log(-x1 - x2 +4))
# Form and solve problem.
prob = cp.Problem(obj,constraints)
prob.solve() # Returns the optimal value.
print("status:", prob.status)
print("optimal value", prob.value)
print("optimal var", x1.value, x2.value)
#status: optimal
#optimal value 3.922011151970246
#optimal var 3.4999999744531194 0.5000000237666717
```

We have checked the minimum obj for different t values. What we have noticed is when t increases, 'obj' is decreasing.

Solution by Python program:

t	Obj
1	(not accurate optimal)17.8574
2	(not accurate optimal)11.3549
3	(not accurate optimal)8.5412
5	(not accurate optimal)6.6692
10	(not accurate optimal)5.2566
100	3.9220

Solution by self calculation:

t	Obj
1	23.8965
2	13.8233
3	10.4655
5	7.7793
10	5.7647
100	3.9515

3 Modeling constrained problems:

We now focus on modeling some kind of real problems.

3.1 Problem: Water Resources

The amount of water, city will drawn from reservoir in L: a
The amount of water, city will drawn from stream in L: b
Constraints:
b$\leq$ 100
$(50 * a + 250 * b)/500 \leq 100$
$a + b = 500$
Objective is to reduce the cost. Minimize the obj. TotalCost : $obj : 100 * a + 50 * b$

```
import cvxpy as cp
a = cp.Variable()
b = cp.Variable()

constraint = [b<=100,(50*a+250*b)/500<=100,a+b==500]
obj = cp.Minimize(100*a+50*b)
prob = cp.Problem(obj,constraint)
prob.solve()
print("status:", prob.status)
print("optimal value", prob.value)
print("optimal var", a.value, b.value)
#status: optimal
#optimal value 44999.999979746055
#optimal var 399.9999995949225 100.00000040507622
```

a and b are 400 and 100 respectively. These values validates the constraints.
This means that we need 400,000 litres from reservoir and 100,000 litres from stream.

3.2 Good-smelling perfume design:

The portion of blend 1: a
The portion of blend 2: b
The portion of blend 3: c
The portion of blend 4: d

The Total of all portion= a+b+c+d
Constraints:
b $\geq 0.05 * Total$
$b \leq 0.2 * Total$
$c \geq 0.3 * Total$
$a \geq 0.1 * Total$
$a \leq 0.25 * Total$
$0.35 * a + 0.6 * b + 0.35 * c + 0.4 * d \leq 0.5 * total$
$0.15 * a + 0.05 * b + 0.2 * c + 0.1 * d \geq 0.08 * total$
$0.15 * a + 0.05 * b + 0.2 * c + 0.1 * d \leq 0.13 * total$
$0.3 * a + 0.2 * b + 0.4 * c + 0.2 * d \leq 0.35 * total$
$0.2 * a + 0.15 * b + 0.05 * c + 0.3 * d \geq 0.19 * total$
$total \geq 1$
$Objective\ function\ is, minimize\ obj : 55 * a + 65 * b + 35 * c + 85 * d$

```
import cvxpy as cp
a = cp.Variable()
b = cp.Variable()
c = cp.Variable()
d = cp.Variable()
total=a+b+c+d

constraint = [b>=0.05*total,
b<=0.2*total,
c>=0.3*total,
a>=0.1*total,
a<=0.25*total,
0.35*a+0.6*b+0.35*c+0.4*d<=0.5*total,
0.15*a+0.05*b+0.2*c+0.1*d>=0.08*total,
0.15*a+0.05*b+0.2*c+0.1*d<=0.13*total,
0.3*a+0.2*b+0.4*c+0.2*d<=0.35*total,
0.2*a+0.15*b+0.05*c+0.3*d>=0.19*total,
total>=1]
obj = cp.Minimize(55*a+65*b+35*c+85*d)
prob = cp.Problem(obj,constraint)
prob.solve()
print("status:", prob.status)
print("optimal value", prob.value)
print("optimal var", a.value, b.value, c.value, d.value)

#status: optimal
#optimal value 62.999999996553605
#optimal var 0.13999999986708458 0.13999999998214976 0.30000000007485483 0.4200000000282858
```

Optimal solutions of a,b,c and d are 0.14, 0.14, 0.3 and 0.42 respectively.
These values validates all the constraints. Optimal cost is around 62.

3.3 Roadway expenses:

Amount of money spend for rural: x_{rural}
$Amount\ of\ money\ spend\ for\ urban:\ x_{urban}$
$Amount\ of\ money\ benefit\ from\ rural:\ B_{rural}$
$Amount\ of\ money\ benefit\ from\ urban:\ B_{urban}$
$B_{rural} = 7000 * log(1 + x_rural)$
$B_{urban} = 5000 * log(1 + x_urban)$
Constraints:
$x_{rural} \geq 0$;
$x_{urban} \geq 0$;
$x_{rural} + x_{urban} \leq 200$;
$Objective\ function\ is\ maximize\ obj: (B_{rural} + B_{urban} - x_{rural} - x_{urban})$;

```
import cvxpy as cp


x_rural= cp.Variable()
x_urban= cp.Variable()

B_rural=7000* (cp.log(1+x_rural))
B_urban=5000* (cp.log(1+x_urban))


constraint = [x_rural>=0, x_urban>=0, x_rural+x_urban<=200]

obj = cp.Maximize(B_rural+B_urban-x_rural-x_urban)
prob = cp.Problem(obj,constraint)
prob.solve()
print("status:", prob.status)
print("optimal value", prob.value)
print("optimal var", x_rural.value, x_urban.value, obj.value)
#status: optimal
#optimal value 55348.89313228272
#optimal var 116.83333815650475 83.16666070848763 55348.89311062496
```

3.4 Design you own optimization problem

Explanation: The problem is about the demand and supply in the market and how can a provider compete the demand provided maximizing his profit at the same time.

Branch	Centre1	Centre2	Centre3
Kolkata	400 INR	480 INR	320 INR
Jalpaiguri	480 INR	240 INR	540 INR

Lets say the provider has the production branch located at Kolkata and Jalpaiguri, and he want to distribute the product, the price for product from Jalpaigur to different distribution centres is given and so is for the price of transportation from Kolkata to centres.
The limit of products Kolkata can supply is 300
The limit of product Jalpaiguri can supply is 500
Also the centre has demands:
Centre 1 with 200
Centre 2 with 300
Centre 3 with 250
supply of unit of product from Kolkata to centre1: x_1
$supply\ of\ unit\ of\ product\ from\ Kolkata\ to\ centre2 : x_2$
$supply\ of\ unit\ of\ product\ from\ Kolkata\ to\ centre3 : x_3$
$supply\ of\ unit\ of\ product\ from\ Jalpaiguri\ to\ centre1 : y_1$
$supply\ of\ unit\ of\ product\ from\ Jalpaiguri\ to\ centre2 : y_2$
$supply\ of\ unit\ of\ product\ from\ Jalpaiguri\ to\ centre3 : y_3$
Constraints:
$x1 + x2 + x3 \leq 300,$
$y1 + y2 + y3 \leq 500,$

$x1 + y1 = 200,$
$x2 + y2 = 300,$
$x3 + y3 = 250,$
$x1 \geq 0,$
$x2 \geq 0,$
$x3 \geq 0,$
$y1 \geq 0,$
$y2 \geq 0,$
$y3 \geq 0$
$Objective\ function\ is\ maximize\ obj : (400 * x_1 + 480 * x_2 + 320 * x_3 + 480 * y_1 + 240 * y_2 + 540 * y_3)$

Program:

```
import cvxpy as cp
import math

x1 = cp.Variable()
x2 = cp.Variable()
x3 = cp.Variable()
y1 = cp.Variable()
y2 = cp.Variable()
y3 = cp.Variable()

constraint = [x1+x2+x3<=300,
              y1+y2+y3<=500,
              x1+y1==200,
              x2+y2==300,
```

```
                x3+y3==250,
                x1>=0,
                x2>=0,
                x3>=0,
                y1>=0,
                y2>=0,
                y3>=0]
obj = cp.Minimize(400*x1 + 480*x2 + 320*x3 + 480*y1 + 240*y2 +540*y3)
prob = cp.Problem(obj,constraint)
prob.solve()
print("optimal var", x1.value, x2.value, x3.value, y1.value, y2.value, y3.value, obj.value)
print("status:", prob.status)
print("optimal value", prob.value)
print("The x2 value upto 2 decimal places is",round(6.000299442986485e-09,2))
print("The y3 value upto 2 decimal places is",round(9.304159351672448e-08,2))

#optimal var 49.99999993124693 6.000299442986485e-09 249.999999906963

#150.00000006875356 299.99999999400063 9.304159351672448e-08 244000.0000274114
#status: optimal
#optimal value 244000.0000274114
#The x2 value upto 2 decimal places is 0.0
#The y3 value upto 2 decimal places is 0.0
```

Optimal Solution found:3 iteration
Objective: 244000
Conclusion: to optimize they wouldn't sent the packet from Kolkata to centre 2, and from Jalpaiguri to centre 3.