Programación Lineal: Método Gráfico

Cesar Ricardo Altamirano Nava*

October 19, 2020

Programa Lineal

Se entiende por programa lineal aquel que optimiza:

$$Z_{opt} = \begin{pmatrix} c_1 & c_2 & \dots & c_n \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$$

Sujeto a las siguientes condiciones:

$$\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \gtreqless \begin{pmatrix} b'_1 \\ b'_2 \\ \vdots \\ b'_m \end{pmatrix}$$

$$\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} > \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

Esta es la forma matricial de un programa lineal, sin embargo se puede rescribir de la siguiente forma:

$$Z_{opt} = c\mathbf{X}$$

Sujeto a:

$$\mathbf{AX} \gtreqless \mathbf{b}$$

$$\mathbf{X} > 0$$

Y se llama forma canónica de un programa lineal cuando:

$$Z_{opt} = c\mathbf{X}$$

Sujeto a:

$$\mathbf{AX} \leq \mathbf{b}$$

$$\mathbf{X} > 0$$

Método Gráfico

Notese que la matriz A es un sistema de n variables con m ecuaciones, de esta forma, para $n = 2$ tenemos un sistema de m ecuaciones donde cada ecuación es una recta en $\mathbb{R}^2$, de esta forma se puede visualizar en el plano el sistema de ecuaciones al gratificar cada recta correspondiente a cada ecuación.

De esta forma tenemos que la matriz A, sera una matriz de $m \times 2$, ademas de la matriz columna b de $m \times 1$:

$$A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ \vdots & \vdots \\ a_{m1} & a_{m2} \end{pmatrix} b' = \begin{pmatrix} b'_1 \\ b'_2 \\ \vdots \\ b'_m \end{pmatrix}$$

Cada renglón de la matriz A seria una recta con la forma:

$$a_1\ x + a_2\ y = b'$$

Dada la ecuación general de la recta:

$$y = mx + b$$

Se pueden deducir los coeficientes m y b aa partir de la primera ecuación:

$$m_i = -\frac{a_{i1}}{a_{i2}}$$

$$b_i = \frac{b'_i}{a_{i2}}$$

Donde i es un numero natural menor o igual a m y representa la i-esima recta del sistema de caciones, de esta formo podemos expresar las m ecuaciones con la ecuación general de la recta (Notese que cuando la recta es vertical, analíticamente significa dividir entre cero)

*Agradecimientos al Gran Búho por su aportación en la Implementación y mejoramiento del algoritmo

Algoritmo para encontrar el punto optimo

Para encontrar el punto máximo se siguen los siguientes pasos:

1. Encontrar el conjunto P = puntos de intersección entre las recatas de la matriz A
2. Encontrar el subconjunto de $P_e \subseteq P$, tal que, $P_e = \{p \in P$ / p es un punto extremo del programa lineal dado por la matriz A$\}$
3. Encontrar el punto optimo para la función objetivo Z en P_e

Encontrar puntos intersección

Dadas dos ecuaciones de A, para saber el valor de x en el cual se intersectan las igualamos:

$$m_i\ x + b_i = m_j\ x + b_j \quad para\ i \neq j$$

La cual, a la hora de despejar x, obtenemos:

$$x = \frac{b_j - b_i}{m_i - m_j}$$

Notese que en el caso de que dos rectas sean paralelas tendrán la misma pendiente, por lo que se dará una división entre cero y ademas el numero de intersecciones en un sistema de m ecuaciones sera:

$$numIntersecciones \leq m^2 - \sum_{i=1}^{m} i$$

Por otra parte, los datos requeridos para calcular las intersecciones son:

$$m = \begin{pmatrix} m_1 \\ m_2 \\ \vdots \\ m_m \end{pmatrix} b = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix}$$

Con m el numero de restricciones en el programa lineal o de ecuaciones en el sistema de ecuaciones

Algorithm 1: Obtener puntos de intersección

```
Data: m=(m_1, m_2, ..., m_m),b=(b_1, b_2, ..., b_m), m + 2
Result: Conjuto de puntos de interseccion P
begin
    Puntos[ ];
    numPuntos = 0;
    for i < m do
        if m_i ≠ ∞ then
            for j < m do
                if m_i ≠ m_j then
                    Puntos[]⟵ (b_i, b_i m_j + b_j);
                    numPuntos ⟵ numPuntos + 1;
                end
            end
        else
            for j < m − i − 1 do
                if m_i ≠ m_{j+i} then
                    Punto_x ⟵ (b_{j+i} − b_i)/(m_i − m_{j+i});
                    Punto_y ⟵ Punto_x m_{j+i} + b_{j+i};
                    Puntos[ ]⟵ (Punto_x, Punto_y);
                    numPuntos ⟵ numPuntos + 1;
                end
            end
        end
    end
end
```

Encontrar Puntos Extremos

Dadas las matriz de $m \times 2$ A, y la matriz de $m \times 1$ b'

$$A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ \vdots & \vdots \\ a_{m1} & a_{m2} \end{pmatrix} b' = \begin{pmatrix} b'_1 \\ b'_2 \\ \vdots \\ b'_m \end{pmatrix}$$

podemos formar m ecuaciones de rectas de la forma:

$$a_{i1}x + a_{i2}y = b'_i$$

Entonces, el conjunto P_e de los puntos extremos, serán aquellos puntos $(x, y) \in P$, que cumplan las m ecuaciones $\forall i = 1, 2, \ldots, m$, si tomamos a a_1 y a_2 como las matrices columnas de a donde P esta dado por:

$$P = \begin{pmatrix} x_1 & y_1 \\ x_2 & y_2 \\ \vdots & \vdots \\ x_k & y_k \end{pmatrix} = \begin{pmatrix} P_1 \\ P_2 \\ \vdots \\ P_k \end{pmatrix}$$

Notese que k no necesariamente es igual a m

Algorithm 2: Obtener puntos de intersección

```
Data: P=(P_1, P_2, ..., P_k),b'=(b'_1, b'_2, ..., b'_m),a_1 = (a_11, a_12, ..., a_1m), a_2 = (a_21, a_22, ..., a_2m)
      ,m, numPuntos
Result: Conjuto de puntos extremo P_e, cardinalidad de P_e
begin
    P_e[ ];
    numPuntosExtremo = 0;
    for i < numPuntos do
        cont⟵ 0;
        for j < m do
            if P_i1 < 0 or  P_i2 < 0 then
                cont⟵cont+1;
            else
                if a1_j P_i1 + a2_j P_i2 > b'_j then
                    cont⟵cont+1;
        if cont = 0 then
            P_e[ ]⟵ P_i;
            numPuntosExtremo⟵numPuntosExtremo+1;
```

Encontrar Puntos Optimo

Por ultimo, para encontrar el punto optimo se evaluaran todos los puntos extremo en la función objetivo Z y guardaran los valores en un arreglo, luego este se ordenara de mayor a menor y se mandara el ultimo elemento del arreglo si se busca el punto máximo y el primer elemento del arreglo si se busca el punto mínimo, donde el conjunto de los puntos extremos

$$P_e = \begin{pmatrix} x_1 & y_1 \\ x_2 & y_2 \\ \vdots & \vdots \\ x_s & y_s \end{pmatrix} = \begin{pmatrix} P_{e1} \\ P_{e2} \\ \vdots \\ P_{es} \end{pmatrix}$$

Donde s es menor o igual a k

Algorithm 3: Obtener punto optimo del programa lineal

Data: **P**$_e$=$(P_{e1}, P_{e2}, \ldots, P_{es})$,**b'**=$(b'_1, b'_2, \ldots, b'_m)$,$a_1 = (a_{11}, a_{12}, \ldots, a_{1m})$, $a_2 = (a_{21}, a_{22}, \ldots, a_{2m})$,m, numPuntos

Result: Conjuto de puntos extremo P_e, cardinalidad de P_e

begin

  Z[ ];

  if *numPuntosExtremo = 0* **then**

    Terminar programa

  for $i < numPuntosExtremo$ **do**

    Z$\longleftarrow < z, P_{ei} >$;

  Ordenar(Z);

  if *min = true* **then**

    $P_{opt} \longleftarrow Z_1$;

  else

    $P_{opt} \longleftarrow Z_s$;

Implementación

```python
def Encontrar_Puntos(m,b,n):
    Puntos=[]
    numPuntos=0
    for i in range(n):
        if math.isinf(m[i]):

            for j in range(n):
                if m[i]!=m[j]:
                    if Puntos.count([b[i],(b[i]*m[j])+b[j]])==0 & math.isnan((b[i]*m[j])+b[j]) == False:
                        Puntos.append([b[i],(b[i]*m[j])+b[j]])
                        numPuntos=numPuntos+1
        else:
            for j in range(n-i-1):
                if m[i]!=m[j+i] :
                    Punto_x =(b[j+i]-b[i])/(m[i]-m[j+i])
                    Punto_y = (Punto_x*m[j+i])+b[j+i]

                    if Puntos.count([b[i],(b[i]*m[j])+b[j]])==0:
                        print(math.isnan(Punto_y))
                        Puntos.append([Punto_x,Punto_y])
                        numPuntos=numPuntos+1
    return Puntos,numPuntos
```

Figure 1: Obtener puntos intersección en ´python

```python
def Encontrar_Puntos_Extremo(Puntos, a1,a2, c, numPuntos, n):
    puntos_Extremos=[]
    numPuntos_Extremos=0
    cont=0
    for i in range(numPuntos):
        cont=0
        for j in range(n):
            if Puntos[i][0] < -0 or Puntos[i][1] < -0 or math.isnan(Puntos[i][0]) or math.isnan(Puntos[i][1]):
                cont=cont+1
            else:
                if (a1[j]*Puntos[i][0])+(a2[j]*Puntos[i][1]) > c[j]:
                    cont=cont+1
        if cont==0:
            puntos_Extremos.append(Puntos[i])
            numPuntos_Extremos=numPuntos_Extremos+1
    puntos_Extremos.sort()
    return puntos_Extremos,numPuntos_Extremos
```

Figure 2: Obtener puntos Extremo en ´python

```python
def Encontrar_Punto_Maximo(z,Puntos_Extremo,numPuntos_Extremo):
    Z=[]
    aux1=[]
    if numPuntos_Extremo==0:
        return False,0
    for i in range(numPuntos_Extremo):
        aux=[]
        Z.append((z[0]*Puntos_Extremo[i][0])+(z[1]*Puntos_Extremo[i][1]))
        aux.append(i)
        aux.append(Z[i])
        aux1.append(aux)

    cs = sorted(Z)
    for i in range(numPuntos_Extremo):
        if cs[numPuntos_Extremo-1] == aux1[i][1]:

            break
    print(cs[numPuntos_Extremo-1])

    return cs[numPuntos_Extremo-1],aux1[i][0];

def Encontrar_Punto_Minimo(z,Puntos_Extremo,numPuntos_Extremo):
    Z=[]
    aux1=[]
    if numPuntos_Extremo==0:
        return False,0
    for i in range(numPuntos_Extremo):
        aux=[]
        Z.append((z[0]*Puntos_Extremo[i][0])+(z[1]*Puntos_Extremo[i][1]))
        aux.append(i)
        aux.append(Z[i])
        aux1.append(aux)

    cs = sorted(Z)
    for i in range(numPuntos_Extremo):
        if cs[0] == aux1[i][1]:
            break

    return cs[0],i
```

Figure 3: Obtener punto Optimo python

Resaludo

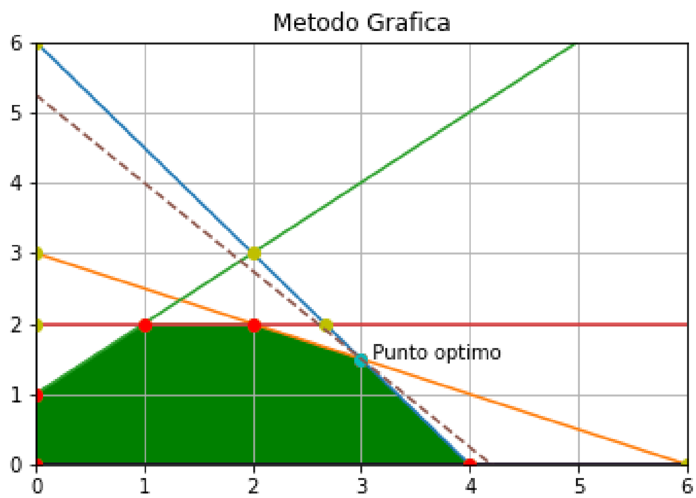

Figure 4: Gráfica del P.L. con el punto optimo