Redes Generativas Antagónicas

Antón Makarov Samusev

Universidad Complutense de Madrid

Universidad Politécnica de Madrid

11 de septiembre de 2019

Índice

Objetivos

- Historia de los métodos de punto interior.
- Descripción de los principales algoritmos.
- Descripción de algunas mejoras para los algoritmos de tipo primal-dual.
- Demostración de la complejidad.
- Implementación de algunos algoritmos en MATLAB.

Descripción del problema

- Problema de programación lineal (PPL): maximizar o minimizar una función lineal sujeta a un conjunto de restricciones lineales.

Descripción del problema

- Problema de programación lineal (PPL): maximizar o minimizar una función lineal sujeta a un conjunto de restricciones lineales.
- Forma primal estándar: mín $c^T x$ sujeto a: $Ax = b, x \geq 0$, donde $c, x \in \mathbb{R}^n, b \in \mathbb{R}^m$ y $A \in \mathcal{M}_{m \times n}(\mathbb{R})$.

Descripción del problema

- Problema de programación lineal (PPL): maximizar o minimizar una función lineal sujeta a un conjunto de restricciones lineales.
- Forma primal estándar: mín $c^T x$ sujeto a: $Ax = b, x \geq 0$, donde $c, x \in \mathbb{R}^n$, $b \in \mathbb{R}^m$ y $A \in \mathcal{M}_{m \times n}(\mathbb{R})$.
- Forma dual: máx $b^T y$ sujeto a: $A^T y + s = c, s \geq 0$, donde $y \in \mathbb{R}^m$ y $s \in \mathbb{R}^n$.

Descripción del problema

- Problema de programación lineal (PPL): maximizar o minimizar una función lineal sujeta a un conjunto de restricciones lineales.
- Forma primal estándar: mín $c^T x$ sujeto a: $Ax = b, x \geq 0$, donde $c, x \in \mathbb{R}^n$, $b \in \mathbb{R}^m$ y $A \in \mathcal{M}_{m \times n}(\mathbb{R})$.
- Forma dual: máx $b^T y$ sujeto a: $A^T y + s = c, s \geq 0$, donde $y \in \mathbb{R}^m$ y $s \in \mathbb{R}^n$.
- Condiciones KKT:

$$\begin{aligned} Ax &= b \\ A^T y + s &= c \\ XSe &= 0 \\ (x, s) &\geq 0. \end{aligned}$$

Historia

- Hasta 1979 $\rightarrow$ Algoritmo del símplex (Dantzig).

Historia

- Hasta 1979 $\rightarrow$ Algoritmo del símplex (Dantzig).
- 1979 $\rightarrow$ algoritmo del elipsoide (Khachiyan).

Historia

- Hasta 1979 $\rightarrow$ Algoritmo del símplex (Dantzig).
- 1979 $\rightarrow$ algoritmo del elipsoide (Khachiyan).
- 1984 $\rightarrow$ algoritmo proyectivo (Karmarkar).

Historia

- Hasta 1979 $\rightarrow$ Algoritmo del símplex (Dantzig).
- 1979 $\rightarrow$ algoritmo del elipsoide (Khachiyan).
- 1984 $\rightarrow$ algoritmo proyectivo (Karmarkar).
- 1985 $\rightarrow$ algoritmo primal de barrera logarítmica (Gill et. al.).

Historia

- Hasta 1979 $\rightarrow$ Algoritmo del símplex (Dantzig).
- 1979 $\rightarrow$ algoritmo del elipsoide (Khachiyan).
- 1984 $\rightarrow$ algoritmo proyectivo (Karmarkar).
- 1985 $\rightarrow$ algoritmo primal de barrera logarítmica (Gill et. al.).
- 1989 $\rightarrow$ algoritmo primal-dual (Meggido).

Historia

- Hasta 1979 $\rightarrow$ Algoritmo del símplex (Dantzig).
- 1979 $\rightarrow$ algoritmo del elipsoide (Khachiyan).
- 1984 $\rightarrow$ algoritmo proyectivo (Karmarkar).
- 1985 $\rightarrow$ algoritmo primal de barrera logarítmica (Gill et. al.).
- 1989 $\rightarrow$ algoritmo primal-dual (Meggido).
- 1992 $\rightarrow$ algoritmo primal-dual predictor-corrector (Mehrotra).

Algoritmo del elipsoide (Factibilidad)

Fórmulas de actualización del elipsoide (d es la fila de A correspondiente a la restricción que no se cumple).

$$x^{k+1} = x^k - \frac{1}{n+1}\frac{D_k d}{\sqrt{d^T D d}}, \tag{1}$$

$$D_{k+1} = \frac{n^2}{n^2-1}\left(D_k - \frac{2}{n+1}\frac{(D_k d)(D_k d)^T}{d^T D_k d}\right). \tag{2}$$

- Dado un punto inicial $x^0 \in \mathbb{R}^n$. Tomamos un elipsoide centrado en él que contenga al conjunto estrictamente factible $\hat{P}$.

Algoritmo del elipsoide (Factibilidad)

Fórmulas de actualización del elipsoide (d es la fila de A correspondiente a la restricción que no se cumple).

$$x^{k+1} = x^k - \frac{1}{n+1}\frac{D_k d}{\sqrt{d^T D d}}, \tag{1}$$

$$D_{k+1} = \frac{n^2}{n^2-1}\left(D_k - \frac{2}{n+1}\frac{(D_k d)(D_k d)^T}{d^T D_k d}\right). \tag{2}$$

- Dado un punto inicial $x^0 \in \mathbb{R}^n$. Tomamos un elipsoide centrado en él que contenga al conjunto estrictamente factible $\hat{P}$.
- El algoritmo comprueba si el punto es o no factible. Si lo es, termina.

Algoritmo del elipsoide (Factibilidad)

Fórmulas de actualización del elipsoide (d es la fila de A correspondiente a la restricción que no se cumple).

$$x^{k+1} = x^k - \frac{1}{n+1}\frac{D_k d}{\sqrt{d^T D d}}, \tag{1}$$

$$D_{k+1} = \frac{n^2}{n^2-1}\left(D_k - \frac{2}{n+1}\frac{(D_k d)(D_k d)^T}{d^T D_k d}\right). \tag{2}$$

- Dado un punto inicial $x^0 \in \mathbb{R}^n$. Tomamos un elipsoide centrado en él que contenga al conjunto estrictamente factible $\hat{P}$.
- El algoritmo comprueba si el punto es o no factible. Si lo es, termina.
- Si no es factible, busca cuál es la inecuación que no se cumple y toma como d la fila de A correspondiente.

Algoritmo del elipsoide (Factibilidad)

Fórmulas de actualización del elipsoide (d es la fila de A correspondiente a la restricción que no se cumple).

$$x^{k+1} = x^k - \frac{1}{n+1}\frac{D_k d}{\sqrt{d^T D d}}, \tag{1}$$

$$D_{k+1} = \frac{n^2}{n^2-1}\left(D_k - \frac{2}{n+1}\frac{(D_k d)(D_k d)^T}{d^T D_k d}\right). \tag{2}$$

- Dado un punto inicial $x^0 \in \mathbb{R}^n$. Tomamos un elipsoide centrado en él que contenga al conjunto estrictamente factible $\hat{P}$.
- El algoritmo comprueba si el punto es o no factible. Si lo es, termina.
- Si no es factible, busca cuál es la inecuación que no se cumple y toma como d la fila de A correspondiente.
- Se actualiza el elipsoide mediante las fórmulas (1) y (2).

Algoritmo del elipsoide (Factibilidad)

Fórmulas de actualización del elipsoide (d es la fila de A correspondiente a la restricción que no se cumple).

$$x^{k+1} = x^k - \frac{1}{n+1}\frac{D_k d}{\sqrt{d^T D d}}, \tag{1}$$

$$D_{k+1} = \frac{n^2}{n^2-1}\left(D_k - \frac{2}{n+1}\frac{(D_k d)(D_k d)^T}{d^T D_k d}\right). \tag{2}$$

- Dado un punto inicial $x^0 \in \mathbb{R}^n$. Tomamos un elipsoide centrado en él que contenga al conjunto estrictamente factible $\hat{P}$.
- El algoritmo comprueba si el punto es o no factible. Si lo es, termina.
- Si no es factible, busca cuál es la inecuación que no se cumple y toma como d la fila de A correspondiente.
- Se actualiza el elipsoide mediante las fórmulas (1) y (2).
- El proceso se repite hasta que se encuentra un punto factible o hasta que se alcanza el máximo número de iteraciones, en cuyo caso se concluye que $\hat{P} = \emptyset$.

Algoritmo del elipsoide (Optimización)

Queremos resolver el problema de programación lineal descrito por:

$$z = \text{máx}\{c^T x : x \in P\}, \quad P = \{x \in \mathbb{R}^n : Ax \leq b\}.$$

Podemos aplicar el algoritmo de factibilidad de manera iterada.
Cada vez que lleguemos a un punto factible, vamos a añadir al sistema $(A|b)$ una fila que proporcione una restricción con la misma forma de la función objetivo y que pase por dicho punto, reduciendo cada vez más el conjunto factible y por tanto la zona donde se busca el óptimo.
Ejemplo:

$$\begin{aligned} &\text{máx}\, x_1 + x_2 \\ \text{sujeto a:}\quad &-x_1 + 3x_2 \leq 3 \\ &2x_1 + x_2 \leq 8 \\ &-6x_1 + x_2 \leq -4 \\ &x_1, x_2 \geq 0. \end{aligned} \tag{3}$$

Ejemplo de aplicación (factibilidad)

Ejemplo de aplicación (Optimización)

Algoritmo primal

- Función de barrera logarítmica: $B(x, \mu) = c^T x - \mu \sum_{i=1}^{n} \log x_i$.

Algoritmo primal

- Función de barrera logarítmica: $B(x, \mu) = c^T x - \mu \sum_{i=1}^{n} \log x_i$.
- Problema a resolver: mín $B(x, \mu)$ sujeto a: $Ax = b$.

Algoritmo primal

- Función de barrera logarítmica: $B(x, \mu) = c^T x - \mu \sum_{i=1}^{n} \log x_i$.
- Problema a resolver: mín $B(x, \mu)$ sujeto a: $Ax = b$.
- Condiciones KKT:

$$\begin{aligned} Ax &= b \\ A^T y + s &= c \\ XSe &= \mu e \\ (x, s) &\geq 0. \end{aligned}$$

Algoritmo primal

- Función de barrera logarítmica: $B(x, \mu) = c^T x - \mu \sum_{i=1}^{n} \log x_i$.
- Problema a resolver: mín $B(x, \mu)$ sujeto a: $Ax = b$.
- Condiciones KKT:

$$\begin{aligned} Ax &= b \\ A^T y + s &= c \\ XSe &= \mu e \\ (x, s) &\geq 0. \end{aligned}$$

- Formulación:

$$\begin{pmatrix} H & A^T \\ A & 0 \end{pmatrix} \begin{pmatrix} -\Delta x \\ y \end{pmatrix} = \begin{pmatrix} c - \mu X^{-1} e \\ 0 \end{pmatrix}.$$

Algoritmo primal

- Función de barrera logarítmica: $B(x, \mu) = c^T x - \mu \sum_{i=1}^{n} \log x_i$.
- Problema a resolver: mín $B(x, \mu)$ sujeto a: $Ax = b$.
- Condiciones KKT:

$$\begin{aligned} Ax &= b \\ A^T y + s &= c \\ XSe &= \mu e \\ (x, s) &\geq 0. \end{aligned}$$

- Formulación:

$$\begin{pmatrix} H & A^T \\ A & 0 \end{pmatrix} \begin{pmatrix} -\Delta x \\ y \end{pmatrix} = \begin{pmatrix} c - \mu X^{-1} e \\ 0 \end{pmatrix}.$$

- Parámetros:
  - centralización $\sigma \rightarrow \mu_{k+1} = \sigma \mu_k$.
  - moderación $\alpha \rightarrow x^{k+1} = x^k + \alpha \Delta x^k$.

Algoritmo primal

- Función de barrera logarítmica: $B(x, \mu) = c^T x - \mu \sum_{i=1}^{n} \log x_i$.
- Problema a resolver: mín $B(x, \mu)$ sujeto a: $Ax = b$.
- Condiciones KKT:

$$\begin{aligned} Ax &= b \\ A^T y + s &= c \\ XSe &= \mu e \\ (x, s) &\geq 0. \end{aligned}$$

- Formulación:

$$\begin{pmatrix} H & A^T \\ A & 0 \end{pmatrix} \begin{pmatrix} -\Delta x \\ y \end{pmatrix} = \begin{pmatrix} c - \mu X^{-1} e \\ 0 \end{pmatrix}.$$

- Parámetros:
  - centralización $\sigma \rightarrow \mu_{k+1} = \sigma \mu_k$.
  - moderación $\alpha \rightarrow x^{k+1} = x^k + \alpha \Delta x^k$.
- Criterio de parada $\rightarrow$ cuando $\|\Delta x^k\|$ sea suficientemente pequeño.

Algoritmo primal-dual

- Aprovecha la información del problema dual.

Algoritmo primal-dual

- Aprovecha la información del problema dual.
- Formulación:

$$\begin{pmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S & 0 & X \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta y \\ \Delta s \end{pmatrix} = \begin{pmatrix} -(A^T y + s - c) \\ -(Ax - b) \\ -XSe + \sigma\mu e \end{pmatrix}.$$

Algoritmo primal-dual

- Aprovecha la información del problema dual.
- Formulación:

$$\begin{pmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S & 0 & X \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta y \\ \Delta s \end{pmatrix} = \begin{pmatrix} -(A^T y + s - c) \\ -(Ax - b) \\ -XSe + \sigma\mu e \end{pmatrix}.$$

- Precisa de los mismos parámetros que el algoritmo primal.

Algoritmo primal-dual

- Aprovecha la información del problema dual.
- Formulación:

$$\begin{pmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S & 0 & X \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta y \\ \Delta s \end{pmatrix} = \begin{pmatrix} -(A^T y + s - c) \\ -(Ax - b) \\ -XSe + \sigma\mu e \end{pmatrix}.$$

- Precisa de los mismos parámetros que el algoritmo primal.
- Criterio de parada $\rightarrow$ cuando $\mu = x^T s/n$ sea suficientemente pequeño. A este valor se le llama medida de dualidad.

Punto inicial I

¿Qué buscamos?

- $x^0, s^0 > 0$.
- Norma pequeña.
- Componentes no muy distintas y no demasiado cercanas a cero.

Punto inicial II

¿Cómo lo hacemos?

- Calculamos el vector $\tilde{x}$ de norma mínima que satisfaga la restricción primal y el vector $(\tilde{y}, \tilde{s})$ que satisfaga la restricción dual con s de norma mínima.

Punto inicial II

¿Cómo lo hacemos?

- Calculamos el vector $\tilde{x}$ de norma mínima que satisfaga la restricción primal y el vector $(\tilde{y}, \tilde{s})$ que satisfaga la restricción dual con s de norma mínima.
- En general $\tilde{x}$ y $\tilde{s}$ pueden tener componentes negativas. Lo evitamos definiendo los parámetros

$$\delta_x = \max(-(3/2)\min_i \tilde{x}_i, 0), \quad \delta_s = \max(-(3/2)\min_i \tilde{s}_i, 0).$$

  Hacemos $\hat{x} = \tilde{x} + \delta_x e, \hat{s} = \tilde{s} + \delta_s e$.

Punto inicial II

¿Cómo lo hacemos?

- Calculamos el vector $\tilde{x}$ de norma mínima que satisfaga la restricción primal y el vector $(\tilde{y}, \tilde{s})$ que satisfaga la restricción dual con s de norma mínima.
- En general $\tilde{x}$ y $\tilde{s}$ pueden tener componentes negativas. Lo evitamos definiendo los parámetros

$$\delta_x = \max(-(3/2)\min_i \tilde{x}_i, 0), \quad \delta_s = \max(-(3/2)\min_i \tilde{s}_i, 0).$$

  Hacemos $\hat{x} = \tilde{x} + \delta_x e, \hat{s} = \tilde{s} + \delta_s e$.
- Para que las componentes de x^0 y s^0 no sean demasiado distintas ni cercanas a cero, definimos dos parámetros más:

$$\hat{\delta_x} = \frac{1}{2}\frac{\hat{x}^T\hat{s}}{e^T\hat{s}}, \quad \hat{\delta_s} = \frac{1}{2}\frac{\hat{x}^T\hat{s}}{e^T\hat{x}}.$$

Punto inicial II

¿Cómo lo hacemos?

- Calculamos el vector $\tilde{x}$ de norma mínima que satisfaga la restricción primal y el vector $(\tilde{y}, \tilde{s})$ que satisfaga la restricción dual con s de norma mínima.
- En general $\tilde{x}$ y $\tilde{s}$ pueden tener componentes negativas. Lo evitamos definiendo los parámetros

$$\delta_x = \max(-(3/2)\min_i \tilde{x}_i, 0), \quad \delta_s = \max(-(3/2)\min_i \tilde{s}_i, 0).$$

  Hacemos $\hat{x} = \tilde{x} + \delta_x e, \hat{s} = \tilde{s} + \delta_s e$.
- Para que las componentes de x^0 y s^0 no sean demasiado distintas ni cercanas a cero, definimos dos parámetros más:

$$\hat{\delta}_x = \frac{1}{2}\frac{\hat{x}^T\hat{s}}{e^T\hat{s}}, \quad \hat{\delta}_s = \frac{1}{2}\frac{\hat{x}^T\hat{s}}{e^T\hat{x}}.$$

- El punto inicial será:

$$x^0 = \hat{x} + \hat{\delta}_x e, \quad y^0 = \tilde{y}, \quad s^0 = \hat{s} + \hat{\delta}_s e.$$

Predicción-corrección

- Obtener el paso afín o de predicción:

$$\begin{pmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S & 0 & X \end{pmatrix} \begin{pmatrix} \Delta x^{\mathrm{af}} \\ \Delta y^{\mathrm{af}} \\ \Delta s^{\mathrm{af}} \end{pmatrix} = \begin{pmatrix} -(A^T y + s - c) \\ -(Ax - b) \\ -XSe \end{pmatrix}.$$

Predicción-corrección

- Obtener el paso afín o de predicción:

$$\begin{pmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S & 0 & X \end{pmatrix} \begin{pmatrix} \Delta x^{\text{af}} \\ \Delta y^{\text{af}} \\ \Delta s^{\text{af}} \end{pmatrix} = \begin{pmatrix} -(A^T y + s - c) \\ -(Ax - b) \\ -XSe \end{pmatrix}.$$

- Utilizando el paso afín, obtener el paso de corrección:

$$\begin{pmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S & 0 & X \end{pmatrix} \begin{pmatrix} \Delta x^{\text{corr}} \\ \Delta y^{\text{corr}} \\ \Delta s^{\text{corr}} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ -\Delta X^{\text{af}} \Delta S^{\text{af}} e \end{pmatrix}.$$

Predicción-corrección

- Obtener el paso afín o de predicción:

$$\begin{pmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S & 0 & X \end{pmatrix} \begin{pmatrix} \Delta x^{\text{af}} \\ \Delta y^{\text{af}} \\ \Delta s^{\text{af}} \end{pmatrix} = \begin{pmatrix} -(A^T y + s - c) \\ -(Ax - b) \\ -XSe \end{pmatrix}.$$

- Utilizando el paso afín, obtener el paso de corrección:

$$\begin{pmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S & 0 & X \end{pmatrix} \begin{pmatrix} \Delta x^{\text{corr}} \\ \Delta y^{\text{corr}} \\ \Delta s^{\text{corr}} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ -\Delta X^{\text{af}} \Delta S^{\text{af}} e \end{pmatrix}.$$

- Combinar ambos pasos:

$$(\Delta x^{\text{af}}, \Delta y^{\text{af}}, \Delta s^{\text{af}}) + (\Delta x^{\text{corr}}, \Delta y^{\text{corr}}, \Delta s^{\text{corr}}).$$

Parámetro de centralización

- Heurística: elegir σ_k apropiado en cada iteración.

Parámetro de centralización

- Heurística: elegir σ_k apropiado en cada iteración.
- Si el paso de escalado afín reduce de manera significativa la medida de dualidad, podemos tomar un valor pequeño de σ_k.

Parámetro de centralización

- Heurística: elegir σ_k apropiado en cada iteración.
- Si el paso de escalado afín reduce de manera significativa la medida de dualidad, podemos tomar un valor pequeño de σ_k.
- Si no se puede hacer un paso grande sobre la dirección, tomamos un valor de σ_k grande, con el objetivo de centrar la iteración siguiente.

Parámetro de centralización

- Heurística: elegir σ_k apropiado en cada iteración.
- Si el paso de escalado afín reduce de manera significativa la medida de dualidad, podemos tomar un valor pequeño de σ_k.
- Si no se puede hacer un paso grande sobre la dirección, tomamos un valor de σ_k grande, con el objetivo de centrar la iteración siguiente.
- Para ello hacemos:

$$\mu_{\text{af}} = \frac{(x + \alpha_{\text{af}}^{\text{primal}} \Delta x)^T (s + \alpha_{\text{af}}^{\text{dual}} \Delta s)}{n}, \quad \sigma = \left(\frac{\mu_{\text{af}}}{\mu} \right)^3.$$

Longitud de paso

- Se calculan por separado las longitudes de paso más grandes posibles tales que las variables x y s se mantengan no negativas.

$$\alpha^{\text{primal}}_{k,\max} = \min_{i:\Delta x_i^k<0} -\frac{x_i^k}{\Delta x_i^k}, \quad \alpha^{\text{dual}}_{k,\max} = \min_{i:\Delta s_i^k<0} -\frac{s_i^k}{\Delta s_i^k}.$$

Longitud de paso

- Se calculan por separado las longitudes de paso más grandes posibles tales que las variables x y s se mantengan no negativas.

$$\alpha_{k,\max}^{\text{primal}} = \min_{i:\Delta x_i^k<0} -\frac{x_i^k}{\Delta x_i^k}, \quad \alpha_{k,\max}^{\text{dual}} = \min_{i:\Delta s_i^k<0} -\frac{s_i^k}{\Delta s_i^k}.$$

- Mejor utilizar pasos grandes $\rightarrow$ se toma un parámetro $\eta \in [0{,}9, 1)$:

$$\alpha_k^{\text{primal}} = \min(1, \eta\alpha_{k,\max}^{\text{primal}}), \quad \alpha_k^{\text{dual}} = \min(1, \eta\alpha_{k,\max}^{\text{dual}}).$$

Complejidad algoritmica

Pregunta

¿Cuál es el número de iteraciones necesario para alcanzar una brecha de dualidad $\leq \psi$?

Sea $\epsilon = \frac{\psi}{\mu_0}$, donde $\mu_0 = \frac{\sum_i x_i^0 s_i^0}{n}$.
Sea K el mínimo valor de $k = 1, 2, \ldots$ tal que $\mu_k \leq \psi$. Se tiene que:

$$K = \frac{n}{\delta} \log\left(\frac{1}{\epsilon}\right).$$

Como δ no depende de n, el número de iteraciones necesarias es:

$$\mathcal{O}\left(n \log\left(\frac{1}{\epsilon}\right)\right).$$

Además, en cada iteración las operaciones tienen complejidad polinomial.
Por tanto, el algoritmo tiene complejidad polinomial.

Rendimiento I

Problema (3) puesto en forma estándar:

$$\begin{aligned} & \text{mín } -x_1 - x_2 \\ \text{sujeto a: } & -x_1 + 3x_2 + h_1 = 3 \\ & 2x_1 + x_2 + h_2 = 8 \\ & -6x_1 + x_2 + h_3 = -4 \\ & x_1, x_2, h_1, h_2, h_3 \geq 0. \end{aligned}$$

Método	**Óptimo - Resultado**	**Iteraciones**	μ
Elipsoide	0.1402	12	-
Mehrotra	0	7	$1{,}1661 \times 10^{-7}$
EJOR	0	19	$1{,}9073 \times 10^{-6}$

Importante

Con estos resultados no queremos decir que un algoritmo sea mejor que otro, no obstante, es satisfactorio comprobar que se corresponden con lo previsto.

Rendimiento II

Comparativa del número de iteraciones necesarias para alcanzar distintas precisiones con el algoritmo de Mehrotra.

Valor de μ pedido	**Valor de μ alcanzado**	**Número de iteraciones**
1×10^{-1}	0,0344	2
1×10^{-2}	0,0029	3
1×10^{-3}	$2{,}2781 \times 10^{-4}$	4
1×10^{-4}	$1{,}8221 \times 10^{-5}$	5
1×10^{-5}	$1{,}4577 \times 10^{-6}$	6
1×10^{-6}	$1{,}1661 \times 10^{-7}$	7
1×10^{-9}	$7{,}4633 \times 10^{-10}$	9
1×10^{-12}	$3{,}8212 \times 10^{-13}$	12

Rendimiento III

Conclusión

- Los métodos de punto interior son una herramienta muy potente para resolver problemas de programación lineal.
- Son muy eficientes al ser aplicados a problemas reales de gran tamaño.
- Aseguran que resolveremos nuestros problemas de optimización en $\mathcal{O}(n \log(1/\epsilon))$ iteraciones.
- La implementación de los algoritmos ha sido una parte fundamental en el aprendizaje.
- La demostración de la complejidad ha revelado técnicas interesantes.

Referencias principales

- Gondzio, J. *Interior point methods 25 years later.* European Journal of Operational Research 218, 587–601, 2012.
- Nemhauser G. L., Wolsey, L. A. *Integer and Combinatorial Optimization.* Wiley Interscience Series in Discrete Mathematics and Optimization, New York, 1988.
- Nocedal, J., Wright, S. J. *Numerical Optimization.* Springer, 2006.
- Robere, R. *Interior Point Methods and Linear Programming.* University of Toronto, 2012.

Gracias por su atención.

¿Preguntas?