Introduction to Optimization

K. R. Sahasranand

Data Science

sahasranand@iitpkd.ac.in

Optimization so far...

Given $A \in \mathbb{R}^{m \times n}, m \geq n$ and $b \in \mathbb{R}^m$, Least squares

$$\min_{x \in \mathbb{R}^n} \|Ax - b\|^2$$

Optimization so far...

Given $A \in \mathbb{R}^{m \times n}, m \geq n$ and $b \in \mathbb{R}^m$, Least squares

$$\min_{x \in \mathbb{R}^n} \|Ax - b\|^2$$

Given $A \in \mathbb{R}^{m \times n}, m < n$ and $b \in \mathbb{R}^m$, Minimum ℓ_2 norm

$$\min_{x \in \mathbb{R}^n} x_1^2 + \cdots + x_n^2 \text{ subject to } Ax = b.$$

Optimization so far...

Given $A \in \mathbb{R}^{m \times n}, m \geq n$ and $b \in \mathbb{R}^m$, Least squares

$$\min_{x \in \mathbb{R}^n} \|Ax - b\|^2$$

Given $A \in \mathbb{R}^{m \times n}, m < n$ and $b \in \mathbb{R}^m$, Minimum ℓ_2 norm

$$\min_{x \in \mathbb{R}^n} x_1^2 + \cdots + x_n^2 \text{ subject to } Ax = b.$$

Given $A \in \mathbb{R}^{m \times n}, m < n$ and $b \in \mathbb{R}^m$, Minimum ℓ_1 norm?

$$\min_{x \in \mathbb{R}^n} |x_1| + \cdots + |x_n| \text{ subject to } Ax = b.$$

Optimization so far...

Given $A \in \mathbb{R}^{m\times n}, m \geq n$ and $b \in \mathbb{R}^m$, Least squares

$$\min_{x\in\mathbb{R}^n} \|Ax - b\|^2$$

Given $A \in \mathbb{R}^{m\times n}, m < n$ and $b \in \mathbb{R}^m$, Minimum ℓ_2 norm

$$\min_{x\in\mathbb{R}^n} x_1^2 + \cdots + x_n^2 \text{ subject to } Ax = b.$$

Given $A \in \mathbb{R}^{m\times n}, m < n$ and $b \in \mathbb{R}^m$, Minimum ℓ_1 norm?

$$\min_{x\in\mathbb{R}^n} |x_1| + \cdots + |x_n| \text{ subject to } Ax = b.$$

Given $A \in \mathbb{R}^{m\times n}, m < n$ and $b \in \mathbb{R}^m$, A general function?

$$\min_{x\in\mathbb{R}^n} f(x) \text{ subject to } Ax = b.$$

Linear program

An optimization problem of the form:

$$\begin{aligned} \text{maximize } & c^\top x \\ \text{subject to } & Ax \leq b \end{aligned}$$

where $c \in \mathbb{R}^n$, $b \in \mathbb{R}^m$, and $A \in \mathbb{R}^{m \times n}$.

Linear program

An optimization problem of the form:

$$\text{maximize } c^\top x$$
$$\text{subject to } Ax \leq b$$

where $c \in \mathbb{R}^n$, $b \in \mathbb{R}^m$, and $A \in \mathbb{R}^{m \times n}$.

(includes $\min$ instead of $\max$, $Ax = b$, $Ax \geq b$ etc. also)

Linear program

An optimization problem of the form:

$$\text{maximize } c^\top x$$
$$\text{subject to } Ax \le b$$

$$\text{where } c \in \mathbb{R}^n,\ b \in \mathbb{R}^m, \text{ and } A \in \mathbb{R}^{m \times n}.$$

(includes $\min$ instead of $\max$, $Ax = b$, $Ax \ge b$ etc. also)

Feasible solutions: All x that satisfy $Ax \le b$

Optimal solution: a *feasible solution* that maximizes $c^\top x$
(a.k.a. *Optimum*)

An example

- A factory has 20 machines and 20 parts

An example

- A factory has 20 machines and 20 parts

- Any machine can produce any part.

An example

- A factory has 20 machines and 20 parts
- Any machine can produce any part.
- Each part-machine production time is known.

An example

- A factory has 20 machines and 20 parts
- Any machine can produce any part.
- Each part-machine production time is known.
- Goal: assign one part to each machine to
  minimize total production time.

An example

- A factory has 20 machines and 20 parts
- Any machine can produce any part.
- Each part-machine production time is known.
- Goal: assign one part to each machine to
  minimize total production time.
- Total possible assignments: $20!$

An example

- A factory has 20 machines and 20 parts
- Any machine can produce any part.
- Each part-machine production time is known.
- Goal: assign one part to each machine to
  minimize total production time.
- Total possible assignments: $20!$
- Brute-force approach checks all $20!$ assignments.

An example

- A factory has 20 machines and 20 parts
- Any machine can produce any part.
- Each part-machine production time is known.
- Goal: assign one part to each machine to
  minimize total production time.
- Total possible assignments: $20!$
- Brute-force approach checks all $20!$ assignments.
- With a computer taking $1\mu s$ per assignment:

An example

- A factory has 20 machines and 20 parts
- Any machine can produce any part.
- Each part-machine production time is known.
- Goal: assign one part to each machine to
  minimize total production time.
- Total possible assignments: $20!$
- Brute-force approach checks all $20!$ assignments.
- With a computer taking $1\mu\mathrm{s}$ per assignment:

  Total time $= 20! \times 10^{-6}$ seconds $\approx 77,147$ years!

Another example

- A company must allocate resources to 100 different projects.

Another example

- A company must allocate resources to 100 different projects.
- Each project uses a portion of 10 shared resources (e.g., budget, manpower).

Another example

- A company must allocate resources to 100 different projects.
- Each project uses a portion of 10 shared resources (e.g., budget, manpower).
- Objective: Maximize total profit across all projects.

Another example

- A company must allocate resources to 100 different projects.
- Each project uses a portion of 10 shared resources (e.g., budget, manpower).
- Objective: Maximize total profit across all projects.
- Constraints: Linear resource limits (e.g., total budget $\leq B$).

Another example

- A company must allocate resources to 100 different projects.
- Each project uses a portion of 10 shared resources (e.g., budget, manpower).
- Objective: Maximize total profit across all projects.
- Constraints: Linear resource limits (e.g., total budget $\leq B$).
- Decision variables: Fraction of resources to allocate to each project (continuous).

Another example

- A company must allocate resources to 100 different projects.
- Each project uses a portion of 10 shared resources (e.g., budget, manpower).
- Objective: Maximize total profit across all projects.
- Constraints: Linear resource limits (e.g., total budget $\leq B$).
- Decision variables: Fraction of resources to allocate to each project (continuous).
- Even with discretized choices (e.g., 100 levels per variable), there are 100^{100} combinations.

Another example

- A company must allocate resources to 100 different projects.
- Each project uses a portion of 10 shared resources (e.g., budget, manpower).
- Objective: Maximize total profit across all projects.
- Constraints: Linear resource limits (e.g., total budget $\leq B$).
- Decision variables: Fraction of resources to allocate to each project (continuous).
- Even with discretized choices (e.g., 100 levels per variable), there are 100^{100} combinations.
- Brute-force evaluation at $1\mu s$ per combination takes $\approx 3.2 \times 10^{213}$ years!

History

- Kantorovich (1939) and Koopmans (WWII)
  - resource and transportation planning

History

- Kantorovich (1939) and Koopmans (WWII)
    - resource and transportation planning
- 1975 Nobel Prize in Economics

History

- Kantorovich (1939) and Koopmans (WWII)
  - resource and transportation planning

- 1975 Nobel Prize in Economics

- Dantzig (1947) – simplex method, a foundational LP algorithm

History

- Kantorovich (1939) and Koopmans (WWII)
  - resource and transportation planning
- 1975 Nobel Prize in Economics
- Dantzig (1947) – simplex method, a foundational LP algorithm
- Efficient in practice but exponential worst-case complexity

History

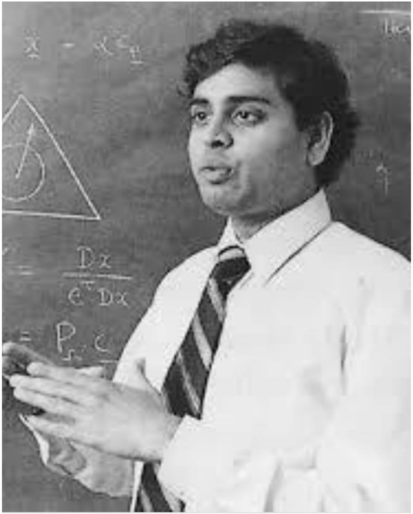
- Kantorovich (1939) and Koopmans (WWII)
    - resource and transportation planning
- 1975 Nobel Prize in Economics
- Dantzig (1947) – simplex method, a foundational LP algorithm
- Efficient in practice but exponential worst-case complexity
- Search for polynomial-time algorithms for LP

History

- Kantorovich (1939) and Koopmans (WWII)
  - resource and transportation planning
- 1975 Nobel Prize in Economics
- Dantzig (1947) – simplex method, a foundational LP algorithm
- Efficient in practice but exponential worst-case complexity
- Search for polynomial-time algorithms for LP
- Khachiyan (1979) – first polynomial-time LP algorithm
  (ellipsoid method)

History

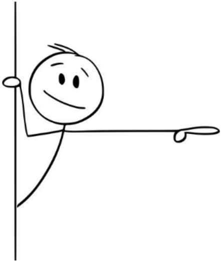
- Kantorovich (1939) and Koopmans (WWII)
  - resource and transportation planning
- 1975 Nobel Prize in Economics
- Dantzig (1947) – simplex method, a foundational LP algorithm
- Efficient in practice but exponential worst-case complexity
- Search for polynomial-time algorithms for LP
- Khachiyan (1979) – first polynomial-time LP algorithm (ellipsoid method)
- Karmarkar (1984) – first practical polynomial-time algorithm

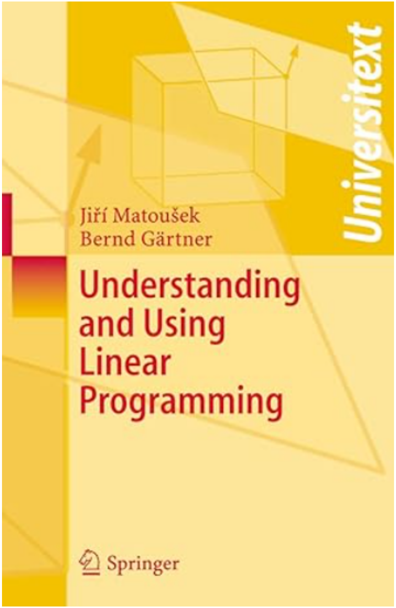


Narendra Krishna Karmarkar (b. 1956)

IITB-CalTech-UC Berkeley

Reference




Universitext
Jiří Matoušek
Bernd Gärtner
Understanding and Using Linear Programming
Springer

Optimized diet – wholesome & cheap

Design a daily diet that satisfies basic nutritional requirements

Nutrient	Minimum Required	Unit
Protein	50	g
Iron	18	mg
Fiber	25	g

Optimized diet – wholesome & cheap

Design a daily diet that satisfies basic nutritional requirements

Nutrient	Minimum Required	Unit
Protein	50	g
Iron	18	mg
Fiber	25	g

Available Food Items:

x_1 : Sprouts $\quad x_2$: Spinach $\quad x_3$: Rice $\quad x_4$: Fish

Optimized diet – wholesome & cheap

Design a daily diet that satisfies basic nutritional requirements

Nutrient	**Minimum Required**	**Unit**
Protein	50	g
Iron	18	mg
Fiber	25	g

Available Food Items:

x_1 : Sprouts x_2 : Spinach x_3 : Rice x_4 : Fish

Nutrient Content per 100g:

Nutrient	Sprouts	Spinach	Rice	Fish
Protein (g)	22	2.9	2.6	20
Iron (mg)	3.9	2.7	0.4	0.7
Fiber (g)	7	2.2	1.8	0

Optimized diet – wholesome & cheap

Design a daily diet that satisfies basic nutritional requirements

Nutrient	**Minimum Required**	**Unit**
Protein	50	g
Iron	18	mg
Fiber	25	g

Available Food Items:

x_1 : Sprouts x_2 : Spinach x_3 : Rice x_4 : Fish

Nutrient Content per 100g:

Nutrient	Sprouts	Spinach	Rice	Fish
Protein (g)	22	2.9	2.6	20
Iron (mg)	3.9	2.7	0.4	0.7
Fiber (g)	7	2.2	1.8	0

Constraints:

$$22x_1 + 2.9x_2 + 2.6x_3 + 20x_4 \geq 50 \qquad \text{(Protein)}$$

Optimized diet – wholesome & cheap

Design a daily diet that satisfies basic nutritional requirements

Nutrient	**Minimum Required**	**Unit**
Protein	50	g
Iron	18	mg
Fiber	25	g

Available Food Items:

$$x_1 : \text{Sprouts} \qquad x_2 : \text{Spinach} \qquad x_3 : \text{Rice} \qquad x_4 : \text{Fish}$$

Nutrient Content per 100g:

Nutrient	Sprouts	Spinach	Rice	Fish
Protein (g)	22	2.9	2.6	20
Iron (mg)	3.9	2.7	0.4	0.7
Fiber (g)	7	2.2	1.8	0

Constraints:

$$22x_1 + 2.9x_2 + 2.6x_3 + 20x_4 \geq 50 \qquad \text{(Protein)}$$

$$3.9x_1 + 2.7x_2 + 0.4x_3 + 0.7x_4 \geq 18 \qquad \text{(Iron)}$$

Optimized diet – wholesome & cheap

Design a daily diet that satisfies basic nutritional requirements

Nutrient	Minimum Required	Unit
Protein	50	g
Iron	18	mg
Fiber	25	g

Available Food Items:

x_1 : Sprouts $\quad x_2$: Spinach $\quad x_3$: Rice $\quad x_4$: Fish

Nutrient Content per 100g:

Nutrient	Sprouts	Spinach	Rice	Fish
Protein (g)	22	2.9	2.6	20
Iron (mg)	3.9	2.7	0.4	0.7
Fiber (g)	7	2.2	1.8	0

Constraints:

$$
\begin{aligned}
22x_1 + 2.9x_2 + 2.6x_3 + 20x_4 &\geq 50 && \text{(Protein)}\\
3.9x_1 + 2.7x_2 + 0.4x_3 + 0.7x_4 &\geq 18 && \text{(Iron)}\\
7x_1 + 2.2x_2 + 1.8x_3 &\geq 25 && \text{(Fiber)}
\end{aligned}
$$

Objective: Minimize Diet Cost

... while meeting the dietary requirements

Each food item has an associated cost per 100g:

Food Item	**Variable**	**Cost (Rs. per 100g)**
Sprouts	x_1	50
Spinach	x_2	20
Rice	x_3	30
Fish	x_4	120

Objective: Minimize Diet Cost

... while meeting the dietary requirements

Each food item has an associated cost per 100g:

Food Item	Variable	Cost (Rs. per 100g)
Sprouts	x_1	50
Spinach	x_2	20
Rice	x_3	30
Fish	x_4	120

Objective: minimize total cost of the diet

Objective: Minimize Diet Cost

... while meeting the dietary requirements

Each food item has an associated cost per 100g:

Food Item	**Variable**	**Cost (Rs. per 100g)**
Sprouts	x_1	50
Spinach	x_2	20
Rice	x_3	30
Fish	x_4	120

Objective: minimize total cost of the diet

$$\min \; 50x_1 + 20x_2 + 30x_3 + 120x_4$$

Objective: Minimize Diet Cost

... while meeting the dietary requirements

Each food item has an associated cost per 100g:

Food Item	**Variable**	**Cost (Rs. per 100g)**
Sprouts	x_1	50
Spinach	x_2	20
Rice	x_3	30
Fish	x_4	120

Objective: minimize total cost of the diet

$$\min\ 50x_1 + 20x_2 + 30x_3 + 120x_4$$

Nonnegativity Constraints: $x_1, x_2, x_3, x_4 \geq 0$

Objective: Minimize Diet Cost

... while meeting the dietary requirements

Each food item has an associated cost per 100g:

Food Item	**Variable**	**Cost (Rs. per 100g)**
Sprouts	x_1	50
Spinach	x_2	20
Rice	x_3	30
Fish	x_4	120

Objective: minimize total cost of the diet

$$\min \; 50x_1 + 20x_2 + 30x_3 + 120x_4$$

Nonnegativity Constraints: $x_1, x_2, x_3, x_4 \geq 0$

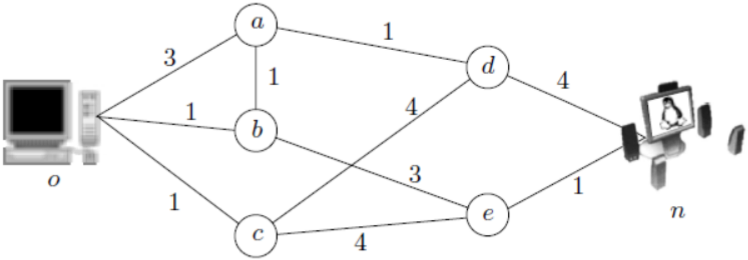
(continue on board)

Flow in a network

You want to transfer your music collection from an old computer to a new one, using a local network.

Flow in a network

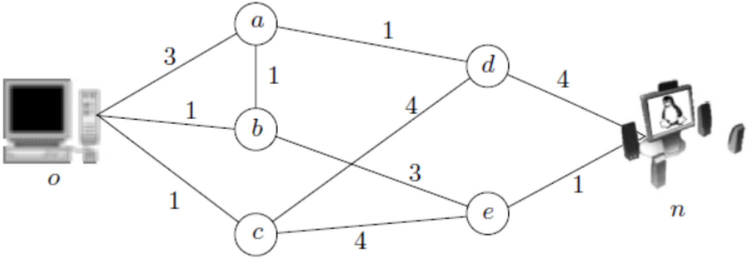
You want to transfer your music collection from an old computer to a new one, using a local network.



The numbers near each data link specify the maximum transfer rate of that link (in Mbps, say)

Flow in a network

You want to transfer your music collection from an old computer to a new one, using a local network.

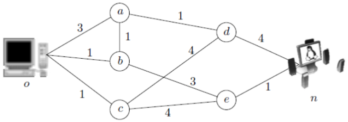


The numbers near each data link specify the maximum transfer rate of that link (in Mbps, say)

→ What is the maximum transfer rate from computer o to computer n?

Flow in a network

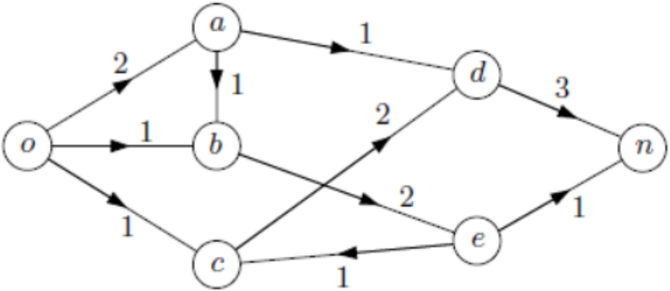
You want to transfer your music collection from an old computer to a new one, using a local network.



$$
\begin{aligned}
\text{maximize } & x_{\mathrm{oa}} + x_{\mathrm{ob}} + x_{\mathrm{oc}} \\
\text{subject to } & -3 \leq x_{\mathrm{oa}} \leq 3, \quad -1 \leq x_{\mathrm{ob}} \leq 1, \quad -1 \leq x_{\mathrm{oc}} \leq 1, \\
& -1 \leq x_{\mathrm{ab}} \leq 1, \quad -1 \leq x_{\mathrm{ad}} \leq 1, \quad -3 \leq x_{\mathrm{be}} \leq 3, \\
& -4 \leq x_{\mathrm{cd}} \leq 4, \quad -4 \leq x_{\mathrm{ce}} \leq 4, \quad -4 \leq x_{\mathrm{dn}} \leq 4, \\
& -1 \leq x_{\mathrm{en}} \leq 1, \\
x_{\mathrm{oa}} &= x_{\mathrm{ab}} + x_{\mathrm{ad}} \\
x_{\mathrm{ob}} + x_{\mathrm{ab}} &= x_{\mathrm{be}} \\
x_{\mathrm{oc}} &= x_{\mathrm{cd}} + x_{\mathrm{ce}} \\
x_{\mathrm{ad}} + x_{\mathrm{cd}} &= x_{\mathrm{dn}} \\
x_{\mathrm{be}} + x_{\mathrm{ce}} &= x_{\mathrm{en}}.
\end{aligned}
$$

Flow in a network

You want to transfer your music collection from an old computer to a new one, using a local network.



The number near each link is the transfer rate on that link, and the arrow determines the direction of the data flow.

Linear regression

.. when square isn't fair

Linear regression

.. when square isn't fair

Instead of least squares:

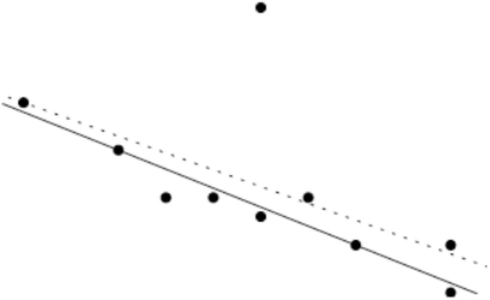
$$\min \sum_{i=1}^{n} \left(mx_i + c - y_i\right)^2$$

suppose we want:

$$\min \sum_{i=1}^{n} \left|mx_i + c - y_i\right|$$

Linear regression

.. when square isn't fair



Instead of least squares:

$$\min \sum_{i=1}^{n} \left(mx_i + c - y_i\right)^2$$

suppose we want:

$$\min \sum_{i=1}^{n} \left|mx_i + c - y_i\right|$$

Least Absolute Error

$\min \sum_{i=1}^{n} \left| mx_i + c - y_i \right|$

Find the line $y = mx + c$ that *minimizes the sum of absolute errors* using LP:

Least Absolute Error

$\min \sum_{i=1}^{n} |mx_i + c - y_i|$

Find the line $y = mx + c$ that *minimizes the sum of absolute errors* using LP:

$$
\begin{aligned}
&\text{minimize } e_1 + \cdots + e_n \\
&\text{subject to} \\
&\qquad e_i \geq mx_i + c - y_i \qquad\qquad \text{for } i \in [n], \\
&\qquad e_i \geq -(mx_i + c - y_i) \qquad \text{for } i \in [n].
\end{aligned}
$$

Least Absolute Error

$\min \sum_{i=1}^{n} \left| mx_i + c - y_i \right|$

Find the line $y = mx + c$ that *minimizes the sum of absolute errors* using LP:

$$\begin{aligned}
&\text{minimize } e_1 + \cdots + e_n \\
&\text{subject to} \\
&\qquad e_i \geq mx_i + c - y_i \qquad \text{for } i \in [n], \\
&\qquad e_i \geq -(mx_i + c - y_i) \quad \text{for } i \in [n].
\end{aligned}$$

The constraints guarantee that

$$e_i \geq \max\{mx_i + c - y_i, -(mx_i + c - y_i)\} = \left| mx_i + c - y_i \right|.$$

Separation of points

A computer-controlled **rabbit trap** should be programmed so that

it catches rabbits,

but if a weasel wanders in, it is released.

Separation of points

A computer-controlled **rabbit trap** should be programmed so that

it catches rabbits,

but if a weasel wanders in, it is released.

Separation of points

A computer-controlled **rabbit trap** should be programmed so that

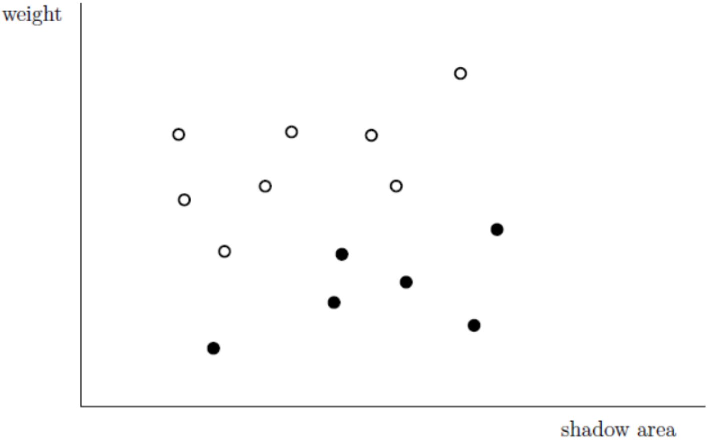
it catches rabbits,

but if a weasel wanders in, it is released.

The trap can **weigh the animal** inside and also can determine the **area of its shadow**.

Separation of points

Find a separating line?



○ Rabbit ● Weasel

Separation of points

Find a separating line – Using LP

$$
\begin{aligned}
\text{maximize } & \delta \\
\text{subject to } & y(p_i) \geq ax(p_i) + b + \delta && ;\ i \in [m], \\
& y(q_j) \leq ax(q_j) + b - \delta && ;\ j \in [n].
\end{aligned}
$$

Separation of points

Find a separating line – Using LP

$$\begin{aligned} &\text{maximize } \delta \\ &\text{subject to } y(p_i) \geq ax(p_i) + b + \delta \quad ; \ i \in [m], \\ &\qquad\qquad\quad y(q_j) \leq ax(q_j) + b - \delta \quad ; \ j \in [n]. \end{aligned}$$