

## *Taller 2 - Cuadrados Mínimos*

Métodos Numéricos

Departamento de Computación, FCEyN, Universidad de Buenos Aires

20 de Octubre de 2017



## Repaso de cuadrados mínimos lineales

Qué es

Cómo lo resuelvo

## Segundo taller

Modelado del problema

Enunciado

## Hasta ahora...

Sabemos resolver sistemas de la forma  $Ax = b$  (con  $A \in \mathbb{R}^{m \times n}$  cuadrada).

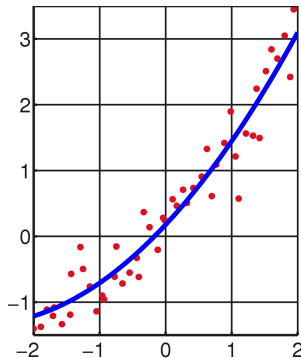
Sirve para muchos problemas y aplicaciones, pero siempre asumimos que  $A$  era cuadrada e inversible.

¿Qué pasa si quiero resolver sistemas con  $m \neq n$ ? Ya no se garantiza única solución.

¿En qué casos tiene sentido plantearse esto?

# Motivación

Producto de algún experimento obtengo varias medidas que se traducen en un conjunto de puntos  $(t_i, y_i)$ .



Quiero encontrar una función que relacione “más o menos bien” mis datos. Dicho de otra manera, quiero una  $f$  tal que  $f(t_i) \approx y_i$ .

## El problema de *data fitting*

Esto se conoce como el problema de *data fitting*, donde lo que buscamos es aproximar  $m$  muestras a partir de  $n$  variables o *features*.

Además, entre más pares  $(t_i, y_i) \implies$  más robusta es mi información sobre el experimento.

Nos vamos a restringir al caso donde  $m \gg n$ , y para resolver el problema vamos a usar el método de cuadrados mínimos lineales (CML).

## Cuadros mínimos lineales

Vamos a armar  $f$  combinando linealmente una familia de funciones  $\mathcal{F} = \{\delta_1, \dots, \delta_n\}$ :  
A las funciones  $\delta_j$  no tienen **ninguna** restricción.

$$f(t_i) = \sum_{j=1}^n x_j \cdot \delta_j(t_i) = \begin{bmatrix} \delta_1(t_i) & \cdots & \delta_n(t_i) \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \approx y_i$$

Entonces, si tengo  $m$  puntos  $(t_i, y_i)$

$$\begin{bmatrix} f(t_1) \\ \vdots \\ f(t_m) \end{bmatrix} = \underbrace{\begin{bmatrix} \delta_1(t_1) & \cdots & \delta_n(t_1) \\ \vdots & \ddots & \vdots \\ \delta_1(t_m) & \cdots & \delta_n(t_m) \end{bmatrix}}_A \cdot \underbrace{\begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}}_x \approx \underbrace{\begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix}}_b$$

## Cuadros mínimos lineales

Vamos a armar  $f$  combinando linealmente una familia de funciones  $\mathcal{F} = \{\delta_1, \dots, \delta_n\}$ :  
A las funciones  $\delta_j$  no tienen **ninguna** restricción.

$$f(t_i) = \sum_{j=1}^n x_j \cdot \delta_j(t_i) = \begin{bmatrix} \delta_1(t_i) & \cdots & \delta_n(t_i) \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \approx y_i$$

Entonces, si tengo  $m$  puntos  $(t_i, y_i)$

$$\begin{bmatrix} f(t_1) \\ \vdots \\ f(t_m) \end{bmatrix} = \underbrace{\begin{bmatrix} \delta_1(t_1) & \cdots & \delta_n(t_1) \\ \vdots & \ddots & \vdots \\ \delta_1(t_m) & \cdots & \delta_n(t_m) \end{bmatrix}}_A \cdot \underbrace{\begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}}_x \approx \underbrace{\begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix}}_b$$

## Cuadrados mínimos lineales

Vamos a armar  $f$  combinando linealmente una familia de funciones  $\mathcal{F} = \{\delta_1, \dots, \delta_n\}$ :  
A las funciones  $\delta_j$  no tienen **ninguna** restricción.

$$f(t_i) = \sum_{j=1}^n x_j \cdot \delta_j(t_i) = \begin{bmatrix} \delta_1(t_i) & \cdots & \delta_n(t_i) \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \approx y_i$$

Entonces, si tengo  $m$  puntos  $(t_i, y_i)$

$$\begin{bmatrix} f(t_1) \\ \vdots \\ f(t_m) \end{bmatrix} = \underbrace{\begin{bmatrix} \delta_1(t_1) & \cdots & \delta_n(t_1) \\ \vdots & \ddots & \vdots \\ \delta_1(t_m) & \cdots & \delta_n(t_m) \end{bmatrix}}_A \cdot \underbrace{\begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}}_x \approx \underbrace{\begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix}}_b$$



## Cuadrados mínimos lineales

¿Cómo sé cuál es la mejor función? Más específicamente, ¿qué solución del sistema anterior estoy buscando?

En el caso de CML buscamos minimizar la suma de los cuadrados de la diferencia entre los datos reales y los que se obtienen con nuestra función.

Es decir que buscamos una función  $f$  que garantice minimizar:

$$\arg \min_f \sum_{i=1}^m (y_i - f(t_i))^2 = \arg \min_x \|b - Ax\|_2^2$$

## Cuadrados mínimos lineales

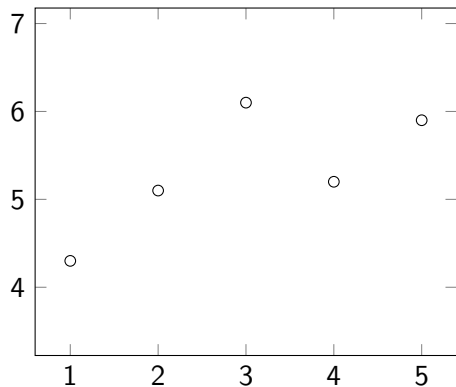
¿Cómo sé cuál es la mejor función? Más específicamente, ¿qué solución del sistema anterior estoy buscando?

En el caso de CML buscamos minimizar la suma de los cuadrados de la diferencia entre los datos reales y los que se obtienen con nuestra función.

Es decir que buscamos una función  $f$  que garantice minimizar:

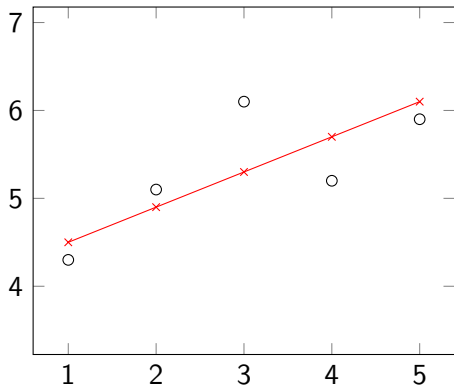
$$\arg \min_f \sum_{i=1}^m (y_i - f(t_i))^2 = \arg \min_x \|b - Ax\|_2^2$$

## Lo mismo gráficamente



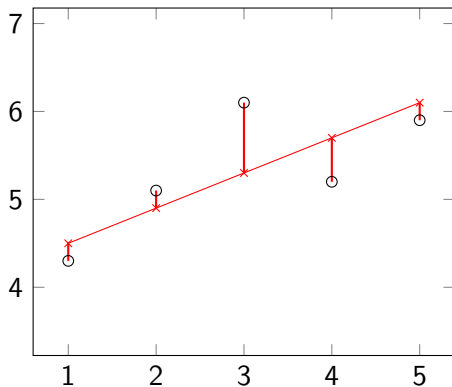
Dados una serie de  $m$  puntos  $(t_i, y_i)$

## Lo mismo gráficamente



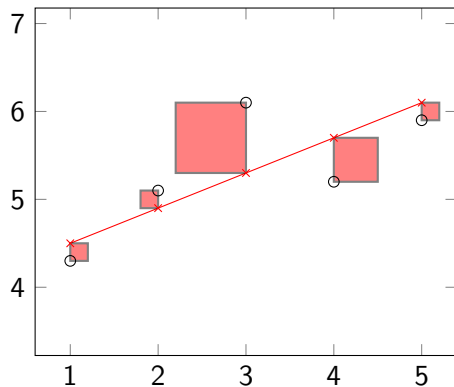
Queremos combinar linealmente  $n$  funciones para obtener  $f(t)$

## Lo mismo gráficamente



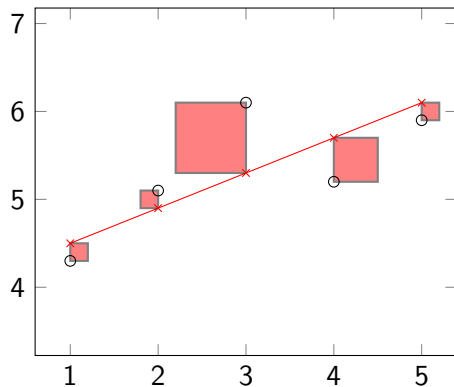
Y minimizar la suma de los errores...

## Lo mismo gráficamente



Y minimizar la suma de los errores **cuadrados**

## Lo mismo gráficamente



En este caso tomé  $\delta_1(t) = 1$  y  $\delta_2(t) = t$  y  $x = \begin{pmatrix} 4,1 \\ 0,4 \end{pmatrix}$ .

Me queda entonces  $f(t) = x_1 \cdot \delta_1(t) + x_2 \cdot \delta_2(t) = 4,1 \cdot 1 + 0,4 \cdot t$

## Recapitulando

$$\mathcal{F} = \{\delta_1, \dots, \delta_n\}$$

$$f(t_i) = \sum_{i=1}^n x_i \cdot \delta_i(t_i)$$

$$\arg \min_f \sum_{i=1}^m (y_i - f(t_i))^2 = \arg \min_x \|b - Ax\|_2^2$$

$$\underbrace{\begin{bmatrix} \delta_1(t_1) & \cdots & \delta_n(t_1) \\ \vdots & \ddots & \vdots \\ \delta_1(t_m) & \cdots & \delta_n(t_m) \end{bmatrix}}_A \cdot \underbrace{\begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}}_x \approx \underbrace{\begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix}}_b$$

¿Cómo podemos resolver esto? Seguramente el sistema ni tenga solución exacta porque tenemos más ecuaciones que incógnitas.



## Intuición

El problema se traduce en minimizar la norma cuadrada del vector residual  $r = b - Ax$

¿Qué vector  $x$  hace que  $Ax$  sea lo más parecido posible a  $b$ ?

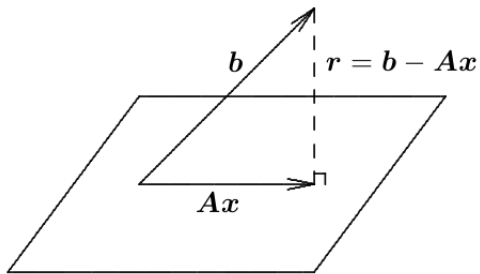


Lo que buscamos entonces es el  $x$  tal que  $Ax$  es la proyección ortogonal de  $b$  sobre  $\text{Im}(A)$

## Intuición

El problema se traduce en minimizar la norma cuadrada del vector residual  $r = b - Ax$

¿Qué vector  $x$  hace que  $Ax$  sea lo más parecido posible a  $b$ ?



Lo que buscamos entonces es el  $x$  tal que  $Ax$  es la proyección ortogonal de  $b$  sobre  $\text{Im}(A)$

## Ecuaciones normales

En base a lo anterior se puede ver que la solución a nuestro problema de cuadrados mínimos está dada por:

$$A^T A x = A^T b \quad (1)$$

Además si  $A$  tiene rango columna completo:

- ▶  $A^T A$  es no singular. De esta manera resolviendo (1) obtenemos la solución al problema de cuadrados mínimos.

- ▶  $A^T A$  es simétrica y definida positiva, por lo que podemos resolver por Cholesky.

## Ecuaciones normales

En base a lo anterior se puede ver que la solución a nuestro problema de cuadrados mínimos está dada por:

$$A^T A x = A^T b \quad (1)$$

Además si  $A$  tiene rango columna completo:

- ▶  $A^T A$  es no singular. De esta manera resolviendo (1) obtenemos la solución al problema de cuadrados mínimos.
- ▶  $A^T A$  es simétrica y definida positiva, por lo que podemos resolver por Cholesky.

# Ecuaciones normales

Ventajas de resolver CML usando ecuaciones normales:

- ▶ Si  $m \gg n$  (cosa muy común), resolvemos un sistema en  $n \times n$
- ▶ Fácil de implementar: calcular  $A^T A$  y resolver por Cholesky

Sin embargo, al resolver de esta manera nos podríamos encontrar con algunos problemas:

- ▶ Pérdida de precisión al calcular  $A^T A$  y  $A^T b$   
 $\text{cond}(A^T A) = \text{cond}(A)^2$

¿Qué otra opción tenemos para evitar estos problemas?

## Ecuaciones normales

Ventajas de resolver CML usando ecuaciones normales:

- ▶ Si  $m \gg n$  (cosa muy común), resolvemos un sistema en  $n \times n$
- ▶ Fácil de implementar: calcular  $A^T A$  y resolver por Cholesky

Sin embargo, al resolver de esta manera nos podríamos encontrar con algunos problemas:

- ▶ Pérdida de precisión al calcular  $A^T A$  y  $A^T b$
- ▶  $\text{cond}(A^T A) = \text{cond}(A)^2$

¿Qué otra opción tenemos para evitar estos problemas?

## Resolución usando QR

- ▶ Si factorizamos  $A$  obtenemos  $A = Q \begin{bmatrix} R' \\ 0 \end{bmatrix}$  con  $Q \in \mathbb{R}^{m \times m}$  ortogonal y  $R' \in \mathbb{R}^{n \times n}$  triangular superior.

- ▶ Como  $Q$  es ortogonal mantiene la norma:

$$\|r\|_2^2 = \|Q^T r\|_2^2 = \|Q^T(b - Ax)\|_2^2 = \left\| Q^T b - \begin{bmatrix} R' \\ 0 \end{bmatrix} x \right\|_2^2$$

- ▶ Si llamamos  $Q^T b = \begin{bmatrix} b'_1 \\ b'_2 \end{bmatrix}$  con  $b'_1 \in \mathbb{R}^n$  y  $b'_2 \in \mathbb{R}^{m-n}$

entonces

$$\|r\|_2^2 = \left\| \begin{bmatrix} b'_1 \\ b'_2 \end{bmatrix} - \begin{bmatrix} R'x \\ 0 \end{bmatrix} \right\|_2^2 = \left\| \begin{bmatrix} b'_1 - R'x \\ b'_2 \end{bmatrix} \right\|_2^2 = \|b'_1 - R'x\|_2^2 + \|b'_2\|_2^2$$

Luego la solución del problema de cuadrados mínimos es  $R'x = b'_1$   
y el error de nuestra aproximación es  $\|r\|_2^2 = \|b'_2\|_2^2$

## Resolución usando QR

- ▶ Si factorizamos  $A$  obtenemos  $A = Q \begin{bmatrix} R' \\ 0 \end{bmatrix}$  con  $Q \in \mathbb{R}^{m \times m}$  ortogonal y  $R' \in \mathbb{R}^{n \times n}$  triangular superior.

- ▶ Como  $Q$  es ortogonal mantiene la norma:

$$\|r\|_2^2 = \|Q^T r\|_2^2 = \|Q^T(b - Ax)\|_2^2 = \left\| Q^T b - \begin{bmatrix} R' \\ 0 \end{bmatrix} x \right\|_2^2$$

- ▶ Si llamamos  $Q^T b = \begin{bmatrix} b'_1 \\ b'_2 \end{bmatrix}$  con  $b'_1 \in \mathbb{R}^n$  y  $b'_2 \in \mathbb{R}^{m-n}$

entonces

$$\|r\|_2^2 = \left\| \begin{bmatrix} b'_1 \\ b'_2 \end{bmatrix} - \begin{bmatrix} R'x \\ 0 \end{bmatrix} \right\|_2^2 = \left\| \begin{bmatrix} b'_1 - R'x \\ b'_2 \end{bmatrix} \right\|_2^2 = \|b'_1 - R'x\|_2^2 + \|b'_2\|_2^2$$

Luego la solución del problema de cuadrados mínimos es  $R'x = b'_1$   
y el error de nuestra aproximación es  $\|r\|_2^2 = \|b'_2\|_2^2$



## Resolución usando QR

- ▶ Si factorizamos  $A$  obtenemos  $A = Q \begin{bmatrix} R' \\ 0 \end{bmatrix}$  con  $Q \in \mathbb{R}^{m \times m}$  ortogonal y  $R' \in \mathbb{R}^{n \times n}$  triangular superior.

- ▶ Como  $Q$  es ortogonal mantiene la norma:

$$\|r\|_2^2 = \|Q^T r\|_2^2 = \|Q^T(b - Ax)\|_2^2 = \left\| Q^T b - \begin{bmatrix} R' \\ 0 \end{bmatrix} x \right\|_2^2$$

- ▶ Si llamamos  $Q^T b = \begin{bmatrix} b'_1 \\ b'_2 \end{bmatrix}$  con  $b'_1 \in \mathbb{R}^n$  y  $b'_2 \in \mathbb{R}^{m-n}$

entonces

$$\|r\|_2^2 = \left\| \begin{bmatrix} b'_1 \\ b'_2 \end{bmatrix} - \begin{bmatrix} R'x \\ 0 \end{bmatrix} \right\|_2^2 = \left\| \begin{bmatrix} b'_1 - R'x \\ b'_2 \end{bmatrix} \right\|_2^2 = \|b'_1 - R'x\|_2^2 + \|b'_2\|_2^2$$

Luego la solución del problema de cuadrados mínimos es  $R'x = b'_1$   
y el error de nuestra aproximación es  $\|r\|_2^2 = \|b'_2\|_2^2$

## Resolución usando QR

- ▶ Si factorizamos  $A$  obtenemos  $A = Q \begin{bmatrix} R' \\ 0 \end{bmatrix}$  con  $Q \in \mathbb{R}^{m \times m}$  ortogonal y  $R' \in \mathbb{R}^{n \times n}$  triangular superior.

- ▶ Como  $Q$  es ortogonal mantiene la norma:

$$\|r\|_2^2 = \|Q^T r\|_2^2 = \|Q^T(b - Ax)\|_2^2 = \left\| Q^T b - \begin{bmatrix} R' \\ 0 \end{bmatrix} x \right\|_2^2$$

- ▶ Si llamamos  $Q^T b = \begin{bmatrix} b'_1 \\ b'_2 \end{bmatrix}$  con  $b'_1 \in \mathbb{R}^n$  y  $b'_2 \in \mathbb{R}^{m-n}$

entonces

$$\|r\|_2^2 = \left\| \begin{bmatrix} b'_1 \\ b'_2 \end{bmatrix} - \begin{bmatrix} R'x \\ 0 \end{bmatrix} \right\|_2^2 = \left\| \begin{bmatrix} b'_1 - R'x \\ b'_2 \end{bmatrix} \right\|_2^2 = \|b'_1 - R'x\|_2^2 + \|b'_2\|_2^2$$

- ▶ Luego la solución del problema de cuadrados mínimos es  $R'x = b'_1$  y el error de nuestra aproximación es  $\|r\|_2^2 = \|b'_2\|_2^2$

## Resolución usando QR

Ventajas de resolver CML usando QR:

- ▶ Evitamos los posibles errores numéricos de calcular  $A^T A$  y  $A^T b$ .
- ▶ Fácil de implementar: obtener factorización QR, calcular  $Q^T b$  y resolvemos  $R'x = b'_1$ . Recordar que  $\text{cond}(Q) = 1$  y  $R'$  es triangular superior.

Por otro lado, si  $m \gg n$ , resolver por QR requiere aproximadamente el doble de trabajo computacional (si lo resolvemos de manera inteligente).

En general, la elección del método de resolución a utilizar dependerá las características del problema particular que se quiera resolver.

## Resolución usando QR

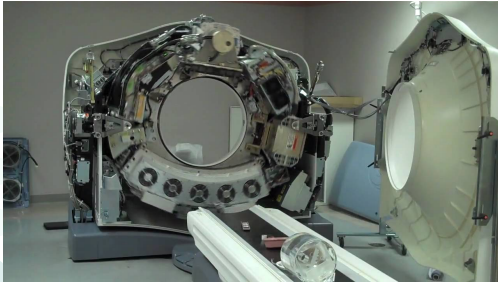
Ventajas de resolver CML usando QR:

- ▶ Evitamos los posibles errores numéricos de calcular  $A^T A$  y  $A^T b$ .
- ▶ Fácil de implementar: obtener factorización QR, calcular  $Q^T b$  y resolvemos  $R'x = b'_1$ . Recordar que  $\text{cond}(Q) = 1$  y  $R'$  es triangular superior.

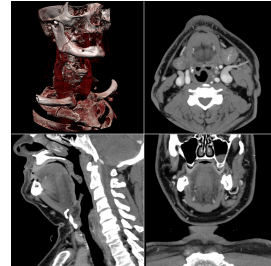
Por otro lado, si  $m \gg n$ , resolver por QR requiere aproximadamente el doble de trabajo computacional (si lo resolvemos de manera inteligente).

En general, la elección del método de resolución a utilizar dependerá las características del problema particular que se quiera resolver.

# Modelemos un problema usando CML

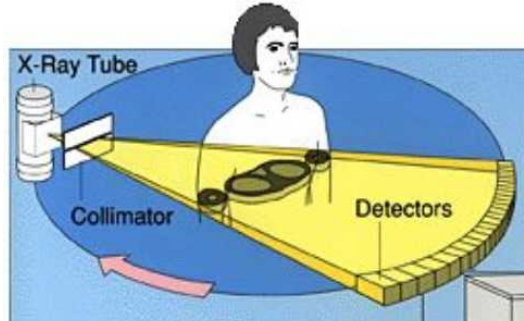


La **tomografía computada** (TC) es una tecnología que utiliza rayos X para producir una imagen de un corte del objeto de estudio. Nos permite “ver” lo que hay adentro.



## ¿Cómo funciona?

El aparato de TC emite múltiples haces de rayos X que inciden sobre el objeto a estudiar. Distintos tejidos absorben los rayos de distintas maneras: vamos a medir cuánto tarda en llegar cada rayo al otro lado <sup>1</sup>.



<sup>1</sup>Esto es una simplificación. En realidad se mide la intensidad de los rayos.

## ¿Cómo lo modelamos con CML?

Vamos a simular lo anterior tratando de reconstruir una TC de un corte de cráneo. Se emiten  $m$  rayos desde distintos ángulos.

Para reconstruir la TC discretizamos el espacio con una grilla de  $p \times q$  y nuestro objetivo va a ser averiguar cuánto tiempo toma a un rayo atravesar cada posición. Decidimos algunos nombres:

- ▶  $b_k$  al tiempo total que toma al rayo  $k$  atravesar al sujeto.
- ▶  $t_{ij}^k$  que va a ser 1 o 0 según si el rayo  $k$  atraviesa la posición  $i, j$
- ▶  $x_{ij}$  al tiempo que toma atravesar la posición  $i, j$  (nuestras incógnitas).

Relacionamos las variables:

$$b_k \approx \sum_{i,j} x_{ij} \cdot t_{ij}^k$$

## ¿Cómo lo modelamos con CML?

Vamos a simular lo anterior tratando de reconstruir una TC de un corte de cráneo. Se emiten  $m$  rayos desde distintos ángulos.

Para reconstruir la TC discretizamos el espacio con una grilla de  $p \times q$  y nuestro objetivo va a ser averiguar cuánto tiempo toma a un rayo atravesar cada posición. Decidimos algunos nombres:

- ▶  $b_k$  al tiempo total que toma al rayo  $k$  atravesar al sujeto.
- ▶  $t_{ij}^k$  que va a ser 1 o 0 según si el rayo  $k$  atraviesa la posición  $i, j$
- ▶  $x_{ij}$  al tiempo que toma atravesar la posición  $i, j$  (nuestras incógnitas).

Relacionamos las variables:

$$b_k \approx \sum_{i,j} x_{ij} \cdot t_{ij}^k$$



## ¿Cómo lo modelamos con CML?

Vamos a simular lo anterior tratando de reconstruir una TC de un corte de cráneo. Se emiten  $m$  rayos desde distintos ángulos.

Para reconstruir la TC discretizamos el espacio con una grilla de  $p \times q$  y nuestro objetivo va a ser averiguar cuánto tiempo toma a un rayo atravesar cada posición. Decidimos algunos nombres:

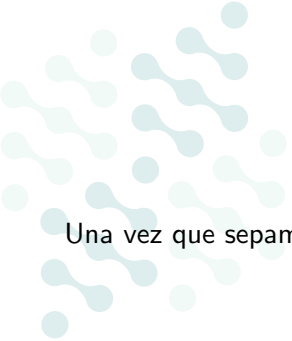
- ▶  $b_k$  al tiempo total que toma al rayo  $k$  atravesar al sujeto.
- ▶  $t_{ij}^k$  que va a ser 1 o 0 según si el rayo  $k$  atraviesa la posición  $i, j$
- ▶  $x_{ij}$  al tiempo que toma atravesar la posición  $i, j$  (nuestras incógnitas).

Relacionamos las variables:

$$b_k \approx \sum_{i,j} x_{ij} \cdot t_{ij}^k$$

## Nuestro modelo

Usando las variables definidas anteriormente, nuestro sistema a resolver queda definido de la siguiente manera:


$$\underbrace{\begin{bmatrix} t_{11}^1 & t_{12}^1 & \cdots & t_{21}^1 & \cdots & t_{pq}^1 \\ t_{11}^2 & t_{12}^2 & \cdots & t_{21}^2 & \cdots & t_{pq}^2 \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ t_{11}^m & t_{12}^m & \cdots & t_{21}^m & \cdots & t_{pq}^m \end{bmatrix}}_A \cdot \underbrace{\begin{bmatrix} x_{11} \\ \vdots \\ x_{pq} \end{bmatrix}}_x \approx \underbrace{\begin{bmatrix} b_1 \\ \vdots \\ b_m \end{bmatrix}}_b$$

Una vez que sepamos los valores  $x_{ij}$  podemos reconstruir nuestra imagen.

Su tarea va a ser responder algunas preguntas sobre CML y completar el código que lo resuelve.

Debe implementarse la resolución con ecuaciones normales y la resolución con QR.

Algunas posibles preguntas para experimentar:

- ▶ ¿Qué tamaño debe tener una buena discretización?
- ▶ ¿Cuántos rayos necesito emitir para que la aproximación sea buena?
- ▶ ¿Qué cambia entre cada método? ¿Tiempo? ¿Calidad?