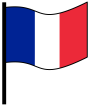
Procesamiento de Lenguaje Natural

Olivia Gutú y Julio Waissman

Maestría en Ciencia de Datos
Semana 4: Vecinos próximos aproximados

$\longrightarrow$



donna → femme

$[1, 2, 1]$ $[3, 2, 2]$ $[0, 0, 3]$ $[1, 2, 5]$ ↓ $[1, 2, 5]$

$[3, 2, 1]$ $[5, 2, 0]$ $[3, 0, 1]$ $[1, 3, 2]$

$$[1,2,5] \xrightarrow[\text{función/algoritmo}]{\text{¿existe una transformación?}} [5,2,0]$$

Introducción: traducción de textos

Lo que busco en abstracto, es una función T:

$$T : \mathbb{R}^3 \to \mathbb{R}^3$$

$$\begin{array}{ccc}
\text{vettore} & \xrightarrow{T} & \text{vectour} \\
\text{vettore} & \xrightarrow{T} & \text{vectour} \\
\text{vettore} & \xrightarrow{T} & \text{vectour} \\
\vdots & \vdots & \vdots \\
\text{vettore} & \xrightarrow{T} & \text{vectour}
\end{array}$$

¿Cuál es el tipo de función más sencilla de este tipo?... sí, ¡una matriz!

Introducción: traducción de textos

La aplicación simultánea de esta transformación lineal se puede escribir como una multiplicación de matrices:

$\mathrm{T}(\text{vettore ☕}^T \ \text{vettore 🚲}^T \cdots \text{vettore 🌍}^T) = (\text{vectour ☕}^T \ \text{vectour 🚲}^T \cdots \text{vectour 🌍}^T)$

Como quiero que mis vectores sean renglones y no columnas, saco transpuesta de ambos lados:

$$\begin{pmatrix} \text{vettore ☕} \\ \text{vettore 🚲} \\ \vdots \\ \text{vettore 🌍} \end{pmatrix} T^T = \begin{pmatrix} \text{vectour ☕} \\ \text{vectour 🚲} \\ \vdots \\ \text{vectour 🌍} \end{pmatrix}$$

Introducción: traducción de textos

La ecuación

$$\begin{pmatrix} \text{vettore ☕} \\ \vdots \\ \text{vettore 🌍} \end{pmatrix} T^T = \begin{pmatrix} \text{vectour ☕} \\ \vdots \\ \text{vectour 🌍} \end{pmatrix}$$

la escribo como $XR = Y$. El problema es encontrar a la matriz R tal que:

$$XR \approx Y$$

Introducción: traducción de textos

- Se tiene que partir de un «diccionario de palabras tipo Python»
- A partir de este diccionario construir R
- Se verifica que tan bien lo hace R
- Si no funciona bien, propongo una nueva R y vuelvo a iterar
- ¿Les suena este razonamiento? ... sí descenso del gradiente.

Puedo decir que R funciona bien si

$$\|XR - Y\|$$

es un valor pequeño para alguna norma, de hecho quiero que Loss sea lo más pequeño posible. Entramos entonces al terreno de la optimización.

Introducción: traducción de textos

$\|A\|_F^2 = \text{traza}(AA^T)$

e.g. Si

$$A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

entonces

$$\|A\|_F^2 = a^2 + b^2 + c^2 + d^2$$

Introducción: traducción de textos

Se considera entonces:

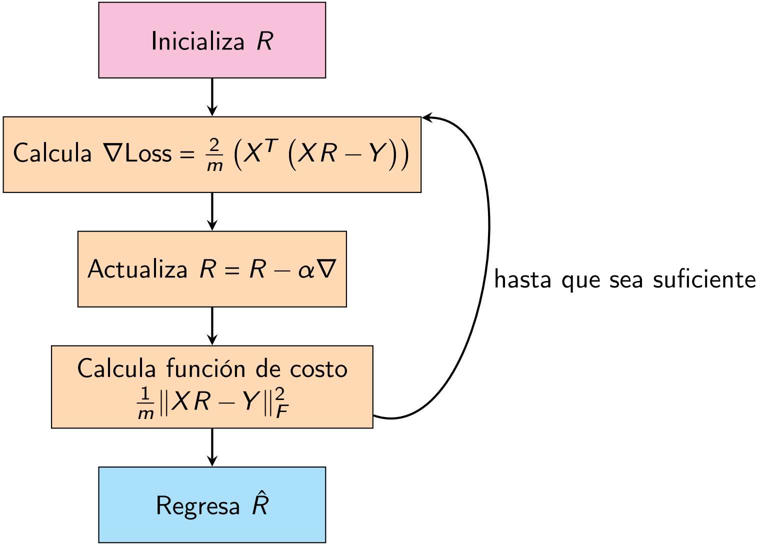
m el número de palabras en el conjunto de entrenamiento

$$\text{Loss} = \frac{1}{m}\|XR - Y\|_F^2$$

$$\nabla\text{Loss} = \frac{2}{m}(X^T(XR - Y))$$

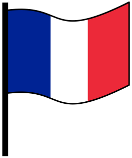
https://math.stackexchange.com/questions/2128462/derivative-of-squared-frobenius-norm-of-a-matrix

Introducción: traducción de textos



Vecinos próximos

$$\xrightarrow{T=R^T}$$



$$\text{vettore}_{\text{ciao}}\ R = \hat{y}$$

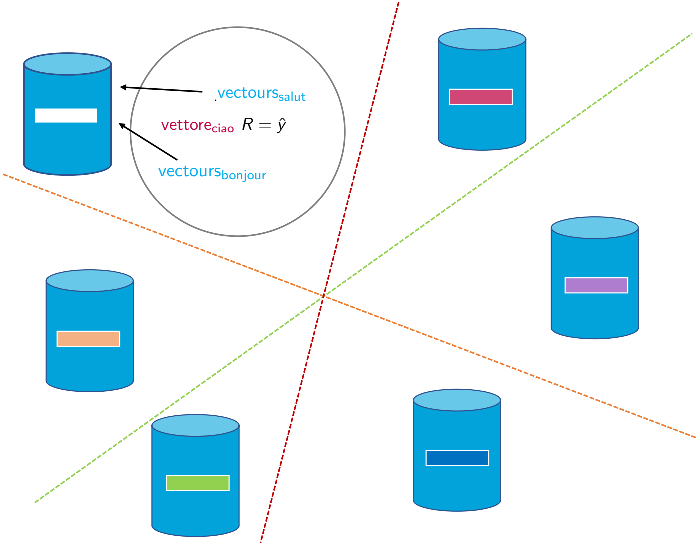
$\hat{y}$ es un vector que no está en la lista de vectours, ¿cuál está más cerca?
e. g. ¿$\text{vectours}_{\text{salut}}$ o $\text{vectours}_{\text{bonjour}}$?

Respuesta platónica: pues simplemente calculo la distancia entre todos (norma entre vectores) y tomo el que esté más cerca.

Después de la respuesta naïve, la reflexión: esto es computacionalmente muy costoso.

La idea es buscar en un subconjunto que escogí anticipadamente, descartando los que ya sé que no tienen posibilidad de ser vectores cercanos.

Vecinos próximos



Tablas hash

$$\text{hash function}\ (\text{vector}) = \text{hash value}$$

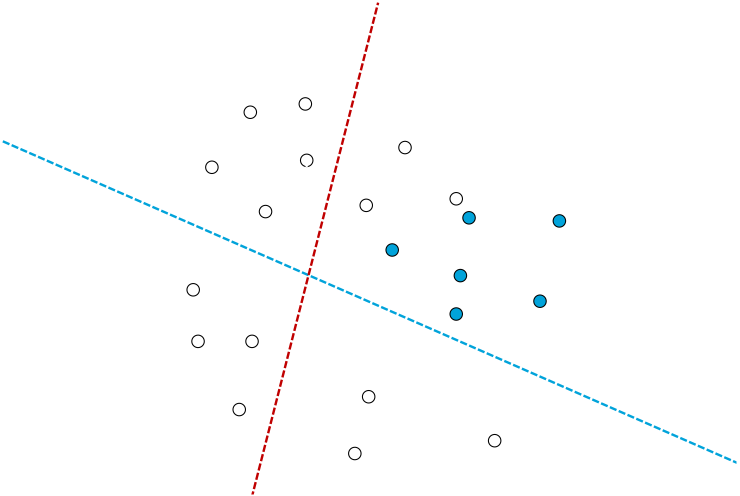
0	1	2	3	4	5	6	7	8	9
100 10		12 42 62		24	15 35			78	69 99

e.g.

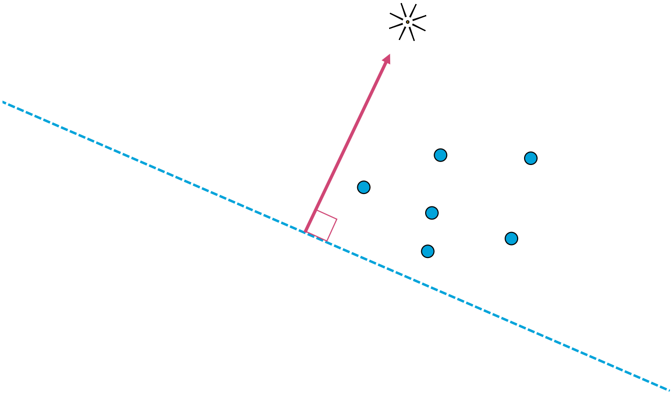
$$\text{hash function}(v) = v \mod 10$$

Esta tabla no es sensible a la localidad: 69 está muy lejos de 99

Tablas hash: sensibles a la localidad

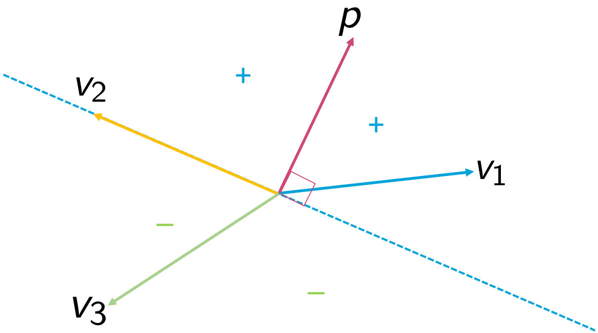


Un plano está definido por un solo vector (el vector normal al plano)



Tablas hash: sensibles a la localidad

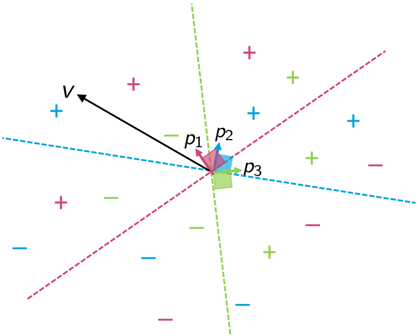
$$
\begin{aligned}
p \text{ dot } v_1 &> 0 \\
p \text{ dot } v_2 &= 0 \\
p \text{ dot } v_3 &< 0
\end{aligned}
$$



Tablas hash: sensibles a la localidad

$$\begin{aligned} p_1 \text{ dot } v &\geq 0 \implies h_1 = 1 \\ p_2 \text{ dot } v &\geq 0 \implies h_2 = 1 \\ p_3 \text{ dot } v &< 0 \implies h_3 = 0 \end{aligned}$$

$$\text{hash value} = 2^0 \times h_1 + 2^1 \times h_2 + 2^2 \times h_3$$



Tablas hash: sensibles a la localidad aproximado

- ¿Cuál es la mejor manera de dividir el espacio por planos?
- n opciones aleatorias de divisiones (por cada división una tabla hash)
- Al final nos quedamos con el que tiene la menor distancia de todos