KSCHOOL

KSCHOOL
KSCHOOL
KSCHOOL

INTRODUCCIÓN AL ÁLGEBRA LINEAL

www.kschool.com

Deep Learning

An MIT Press book in preparation

Ian Goodfellow, Aaron Courville, and Yoshua Bengio

Python & SciPy

Python has >15 year history in scientific computing

- IPython Notebook: 2005-2011
- pandas: 2008-2009
- scikit-learn: 2007
- NumPy: 2006
- matplotlib: 2002
- IPython: 2001
- Numarray: 2001
- SciPy: 1999
- Numeric: 1995

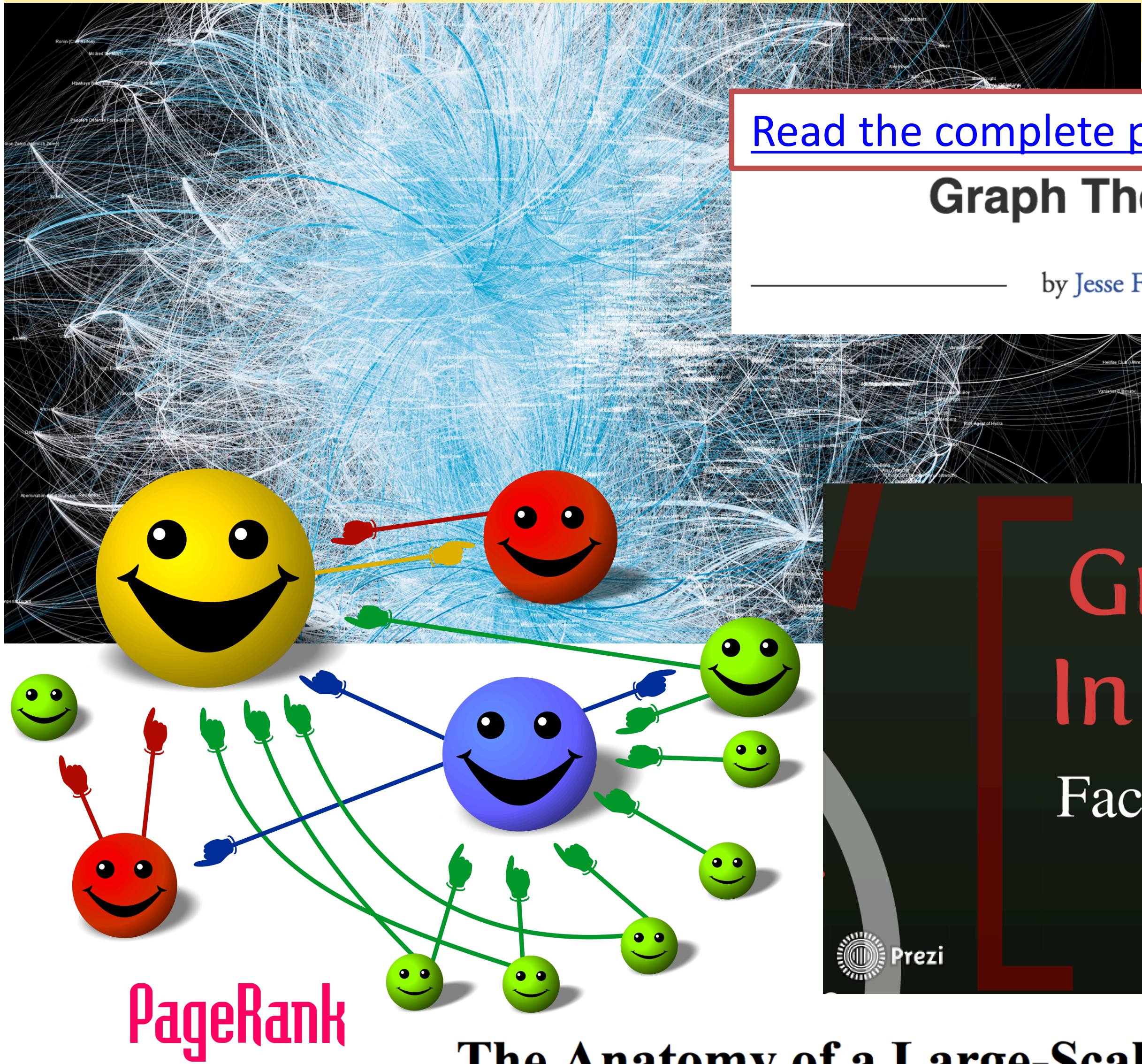
- High performance linear algebra, image processing, optimization via NumPy, optimized C++, FORTRAN
- Large structured data via HDF5, memmap
- Out of core processing, streaming & realtime
- Distributed computing via MPI, IPython Parallel, etc.
- GPU & heterogenous via OpenCL, PyCUDA, others
- Massive adoption in research, national labs, industry (engineering, finance, etc.)

CONTINUUM ANALYTICS

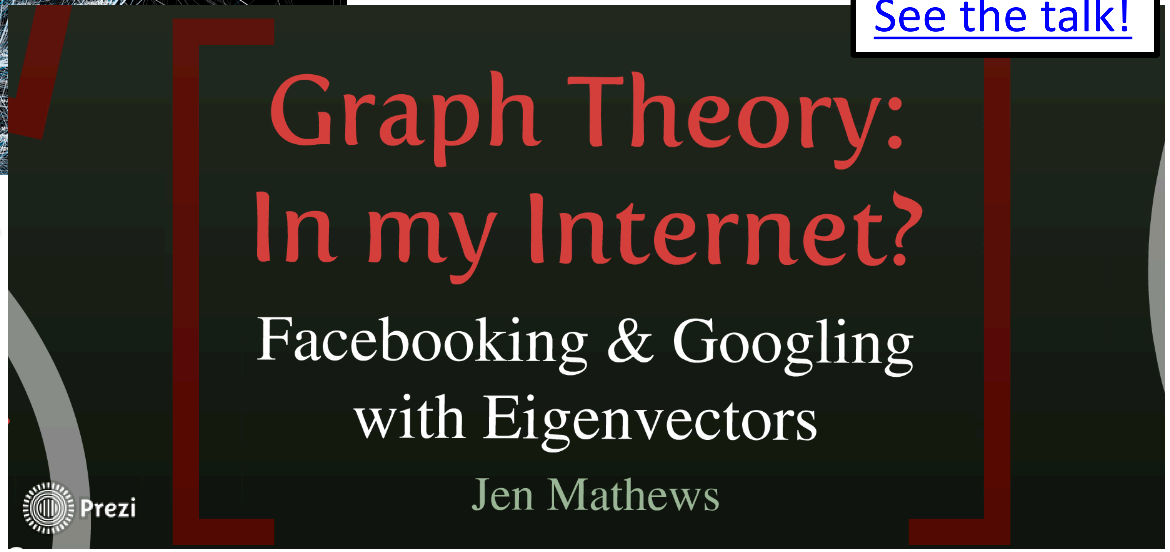
Read the complete post

Graph Theory: Part III (Facebook)

by Jesse Farmer on Wednesday, August 24, 2011



See the talk!



The Anatomy of a Large-Scale Hypertextual Web Search Engine

Sergey Brin and Lawrence Page

Computer Science Department,
Stanford University, Stanford, CA 94305, USA
sergey@cs.stanford.edu and page@cs.stanford.edu

Básicos: vectores y matrices

Trasponer vectores y matrices:

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}; \qquad \mathbf{x}^{\mathrm{T}} = [x_1, x_2, \ldots, x_n]$$

$$\mathbf{A} = \left(A_{ij}\right) = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ a_{31} & a_{32} \end{bmatrix}; \quad \mathbf{A}^{\mathrm{T}} = \mathbf{B} = \left(B_{ij}\right) = \begin{bmatrix} a_{11} & a_{21} & a_{31} \\ a_{12} & a_{22} & a_{32} \end{bmatrix} = \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \end{bmatrix}$$

Básicos: vectores y matrices

Algunas propiedades básicas:

- Productos y normas:

$$\mathbf{x}^{\mathrm{T}} \cdot \mathbf{y} = \sum_i x_i \cdot y_i$$

$$\mathbf{C} = \mathbf{AB} \Rightarrow c_{i,j} = (\mathbf{A} \cdot \mathbf{B})_{i,j} = \sum_k a_{i,k} \cdot b_{k,j}$$

$$\mathbf{x}^{\mathrm{T}} \cdot \mathbf{x} = \sum_i x_i^2 = \|\mathbf{x}\|_2^2; \quad \mathbf{x}^{\mathrm{T}} \cdot \mathbf{y} = \|\mathbf{x}\|_2 \|\mathbf{y}\|_2 \cos\theta$$

$$L^p = \|\mathbf{x}\|_p = \left(\sum_i |x_i|^p\right)^{1/p}; \quad \|\mathbf{A}\|_F = \sqrt{\sum_{i,j} a_{i,j}^2}$$

- Productos de matrices:

$$\mathbf{A}(\mathbf{B}+\mathbf{C}) = \mathbf{AB}+\mathbf{AC}$$
$$\mathbf{A}(\mathbf{BC}) = (\mathbf{AB})\mathbf{C}$$
$$(\mathbf{AB})^{\mathrm{T}} = \mathbf{B}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}}$$

- Matriz unitaria, inversa y sistema de ecuaciones:

$$\forall \mathbf{x} \in \mathrm{R}^n, \mathbf{I}_n \mathbf{x} = \mathbf{x}$$
$$\mathrm{A} \in \mathrm{R}^{nxn}, |\mathrm{A}| \neq 0 \Rightarrow \mathbf{A}^{-1}\mathbf{A} = \mathbf{A}\mathbf{A}^{-1} = \mathbf{I}_n$$

$$\text{Sean } \mathbf{x} \in \mathrm{R}^m, \mathbf{b} \in \mathrm{R}^n, \mathbf{A} \in \mathrm{R}^{nxm}:$$
$$\mathbf{Ax} = \mathbf{b}$$
$$\mathbf{A}^{-1}\mathbf{Ax} = \mathbf{A}^{-1}\mathbf{b}$$
$$\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$$

Más sobre matrices (cuadradas!)

- Traza:

$$Tr(\mathbf{A}) = \sum_i a_{i,i}$$

$$Tr(\mathbf{A}) = Tr(\mathbf{A}^{\mathrm{T}})$$

$$Tr(\mathbf{A}^{-1}) = 1 / Tr(\mathbf{A})$$

$$Tr(\mathbf{XY}) = \sum_i \left(\sum_k x_{i,k} \cdot y_{k,i} \right) = \sum_k \left(\sum_i x_{i,k} \cdot y_{k,i} \right) = Tr(\mathbf{YX})$$

$$Tr(\mathbf{A}^{\mathrm{T}}\mathbf{B}) = Tr(\mathbf{B}^{\mathrm{T}}\mathbf{A}) = Tr(\mathbf{AB}^{\mathrm{T}}) = Tr(\mathbf{BA}^{\mathrm{T}})$$

$$Tr(\mathbf{ABC}) = Tr(\mathbf{BCA}) = Tr(\mathbf{CBA})$$

$$Tr(\mathbf{P}^{-1}\mathbf{AP}) = Tr(\mathbf{APP}^{-1}) = Tr(\mathbf{A})$$

- Determinante:

$$\det(\mathbf{A}) = \sum_{\sigma \in S_n} \operatorname{sgn}(\sigma) \prod_{i=1}^{n} a_{i,\sigma_i}$$ ← wikipedia!

$$\det(\mathbf{A}) = \det(\mathbf{A}^{\mathrm{T}})$$

$$\det(\mathbf{A}^{-1}) = \det(\mathbf{A})^{-1}$$

$$\det(\mathbf{AB}) = \det(\mathbf{A}) \cdot \det(\mathbf{B})$$

Básicos: vectores y matrices

Algunas matrices especiales:

- Matriz diagonal:

$$\mathbf{D} / d_{i,j} = 0 \quad \forall i \neq j$$

$$\mathbf{D}^{-1} = \mathrm{diag}(\mathbf{v})^{-1} = \mathrm{diag}\left(\left[1/v_1, \ldots, 1/v_n\right]^{\mathrm{T}}\right)$$

$$Tr(\mathbf{D}) = \sum v_i; \quad \det(\mathbf{D}) = \prod v_i$$

$$\mathbf{D} = \mathrm{diag}(\mathbf{v}) = \begin{bmatrix} v_1 & 0 & \cdots & 0 \\ 0 & v_2 & 0 & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & 0 & v_n \end{bmatrix}$$

- Matriz simétrica: $\mathbf{S} = \mathbf{S}^{\mathrm{T}} \Rightarrow s_{i,j} = s_{j,i}$

- Matriz ortogonal:

$$\mathbf{M}^{\mathrm{T}}\mathbf{M} = \mathbf{M}\mathbf{M}^{\mathrm{T}} = \mathbf{I} \Rightarrow \mathbf{M}^{-1} = \mathbf{M}^{\mathrm{T}}$$

$$\det(\mathbf{M}) = 1$$

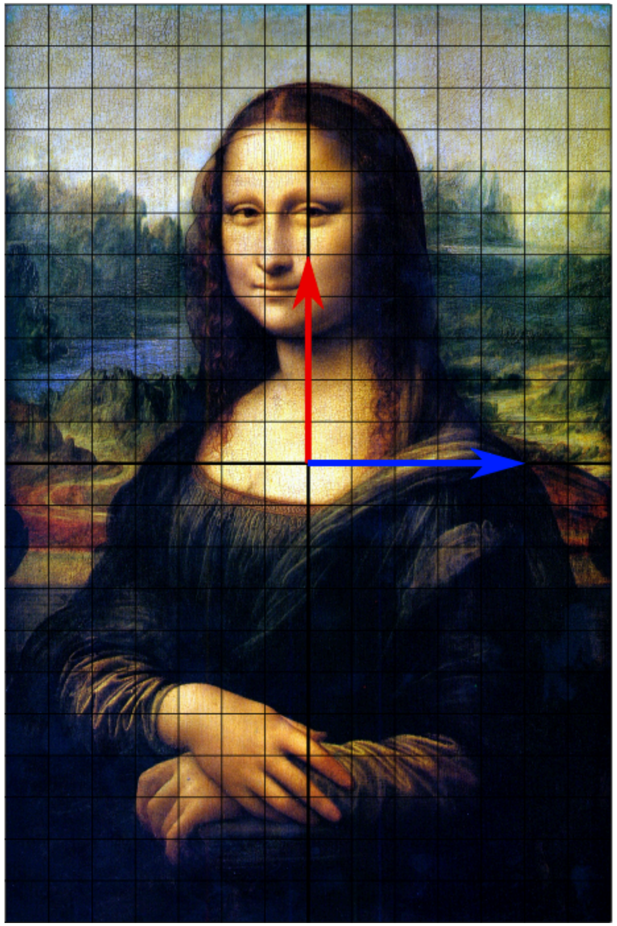
$$\mathbf{M} = \begin{bmatrix} \mathbf{m}_1^{\mathrm{T}} \\ \vdots \\ \mathbf{m}_n^{\mathrm{T}} \end{bmatrix}; \mathbf{M}^{\mathrm{T}} = [\mathbf{m}_1, \ldots, \mathbf{m}_n]; \mathbf{m}_i^{\mathrm{T}}\mathbf{m}_j = \delta_{i,j}$$

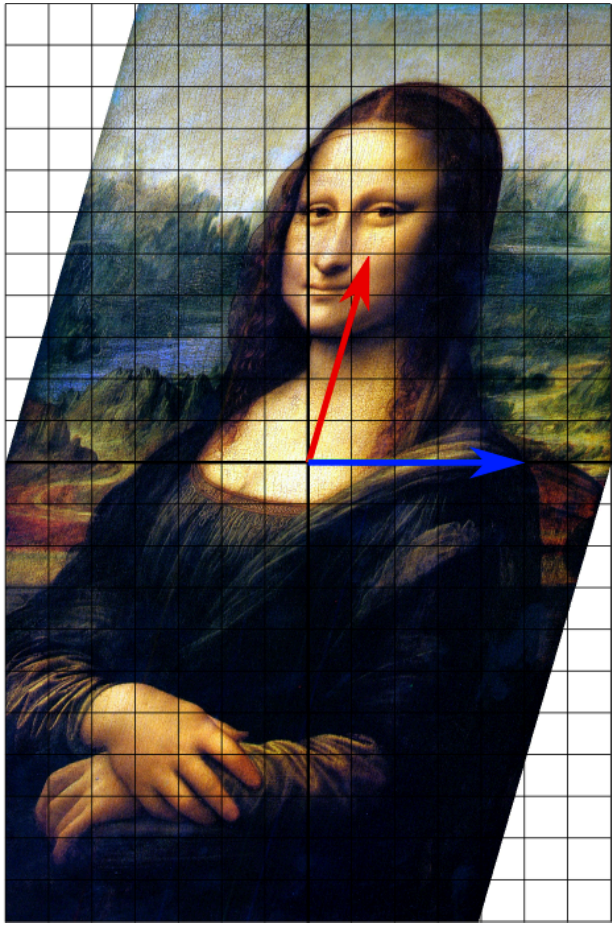
- Matriz semi-definida positiva: $\forall \mathbf{x} \in R^n, \mathbf{x}^{\mathrm{T}}\mathbf{P}\mathbf{x} \geq 0$


WSWCM

No sería brutal escribir una matriz en términos de estas matrices especiales???

Todo es cuestión de la representación que escojamos!





Eigen-decomposition

"Cualquier número es primo o está medido por un primo"
- Eculides, *Los Elementos,* Libro VII, Proposición 32

- Un vector no cero es autovector de una matriz cuadrada si:

$$\mathbf{A}\mathbf{v} = \lambda\mathbf{v}$$

- El número de autovalores de una matriz viene dado por la ecuación:

$$\left(\mathbf{A} - \lambda\right)\mathbf{v} = 0 \Rightarrow \det(\mathbf{A} - \lambda) = 0$$

- Si los autovectores correspondientes son linealmente independientes, entonces:

$$\mathbf{A} = \mathbf{V}\mathbf{D}\mathbf{V}^{-1}, \text{ siendo } \mathbf{V} = [\mathbf{v}_1, \dots, \mathbf{v}_n]; \mathbf{D} = \operatorname{diag}(\lambda)$$

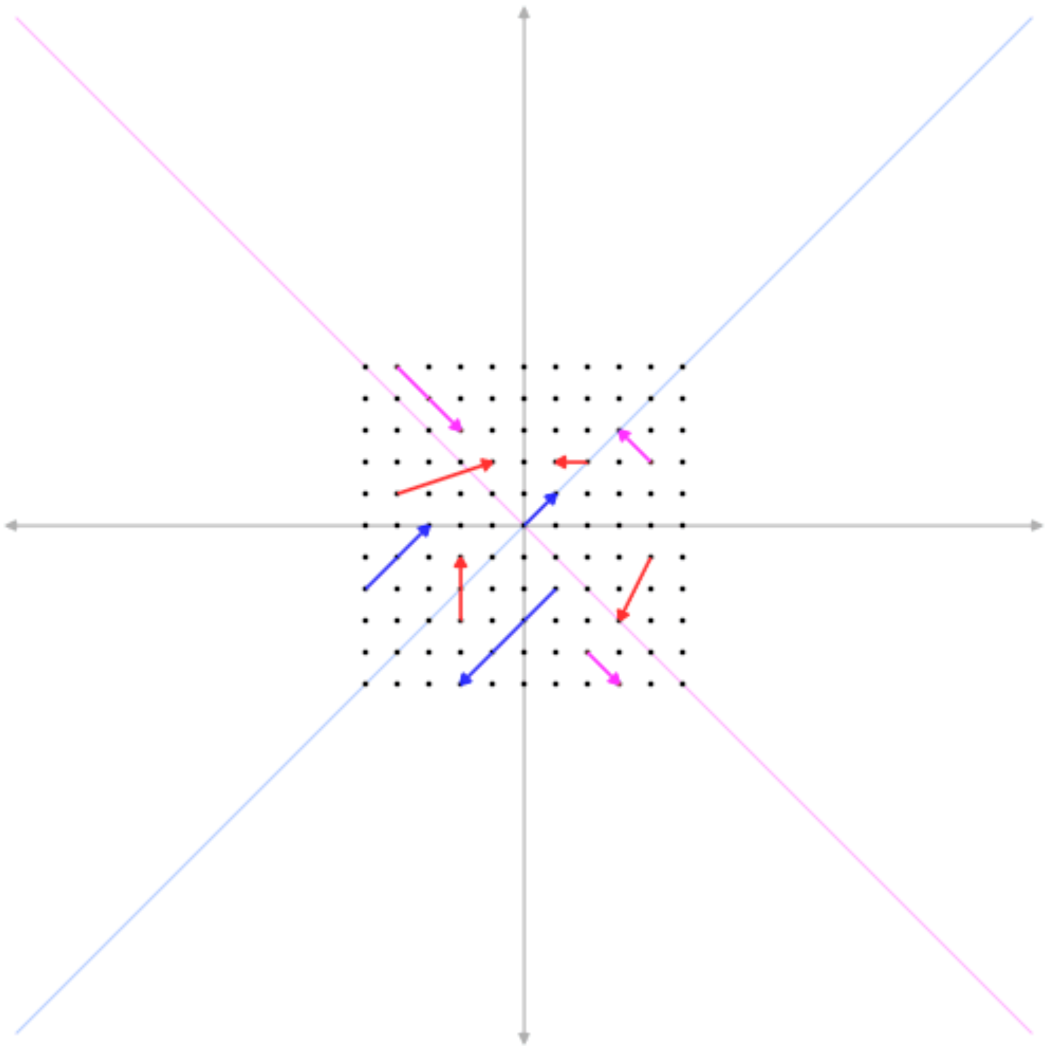
- Ejemplo:

$$\mathbf{A} = \begin{bmatrix} 1 & 0 \\ 1 & 3 \end{bmatrix}$$

Eigen-decomposition

"Cualquier número es primo o está medido por un primo"
- Eculides, *Los Elementos,* Libro VII, Proposición 32

$$\mathbf{A}\mathbf{v} = \lambda\mathbf{v}$$



Eigen-decomposition

- Algunas propiedades importantes:
  - La traza de la matriz es simplemente:

$$\mathrm{Tr}(\mathbf{A}) = \mathrm{Tr}\left(\mathbf{V}\mathbf{D}\mathbf{V}^{-1}\right) = \mathrm{Tr}\left(\mathbf{D}\mathbf{V}^{-1}\mathbf{V}\right) = \mathrm{Tr}\left(\mathbf{D}\right) = \sum \lambda_i$$

  - Y el determinante:

$$\det(\mathbf{A}) = \det\left(\mathbf{V}\mathbf{D}\mathbf{V}^{-1}\right) = \det\left(\mathbf{V}\right)\det\left(\mathbf{D}\right)\det\left(\mathbf{V}^{-1}\right) = \det\left(\mathbf{D}\right) = \prod \lambda_i$$

  - Matriz inversa:

$$\mathbf{A}^{-1} = \left(\mathbf{V}\mathbf{D}\mathbf{V}^{-1}\right)^{-1} = \mathbf{V}\mathbf{D}^{-1}\mathbf{V}^{-1}$$

  - Si **A** es simétrica:

$$\mathbf{A} = \mathbf{Q}\Lambda\mathbf{Q}^{\mathrm{T}};\ \text{siendo } \mathbf{Q}\mathbf{Q}^{\mathrm{T}} = \mathbf{I} \text{ y } \Lambda=\mathrm{diag}(\lambda),\ \lambda \in R$$

  - Si $f \in C^n$

$$f(\mathbf{A}) = \mathbf{V}f(\mathbf{D})\mathbf{V}^{-1}$$

Eigen-decomposition

Generalización

- Hasta ahora nos hemos ceñido a la descomposición (representación) de matrices cuadradas.

- El SVD (*Singular value decomposition*) es un método para factorizar cualquier matriz de números reales.

  - Fundamental en matrices no cuadradas!!!! $\mathbf{A} \in \mathrm{R}^{n \mathrm{x} m}$

- El PCA (*Principal Component Analysis*) es otra técnica para obtener información de una matriz. En este caso, simetrizamos el problema original, de manera que en vez de estudiar la matriz $\mathbf{A} \in \mathrm{R}^{n \mathrm{x} m}$, se estudia la matriz:

$$\mathbf{A}\mathbf{A}^{\mathrm{T}} \in \mathrm{R}^{n \mathrm{xn}}$$

- Por último, el método *Moore-Penrose psuedoinverse* es una forma de obtener soluciones de un sistema de ecuaciones lineales con n≠m (solución aproximada si n<m, o la solución con norma mínima si n>m).


access
hollywood
Liam: Are you ready?
Louis: I'm ready.

- En el eigendecomposition buscábamos una factorización de una matriz cuadrada en la forma:

$$\mathbf{A} = \mathbf{V}\mathrm{diag}(\lambda)\mathbf{V}^{-1}$$

- En particular, la aplicábamos a matrices simétricas, siendo la decomposición:

$$\mathbf{A} = \mathbf{Q}\Lambda\mathbf{Q}^{\mathrm{T}}$$

- SVD se aplica a matrices nxm. La factorización que buscamos es:

$$\mathbf{A} = \mathbf{U}\Sigma\mathbf{V}^{\mathrm{T}}$$

  - **U** es la matriz de autovectores de $\mathbf{AA}^{\mathrm{T}}$ (*left-singular vectors*):

$$\left(\mathbf{AA}^{\mathrm{T}}\right)\mathbf{U}_{:,j} = \lambda_j \mathbf{U}_{:,j}$$

  - **V** es la matriz de autovectores de $\mathbf{A}^{\mathrm{T}}\mathbf{A}$ (*right-singular vectors*):

$$\left(\mathbf{A}^{\mathrm{T}}\mathbf{A}\right)\mathbf{V}_{:,j} = \lambda_j \mathbf{V}_{:,j}$$

  - Los valores singulares σ_i son la raíz cuadrada de los autovalores λ_i de la matriz $\mathbf{A}^{\mathrm{T}}\mathbf{A}$ (iguales a los de $\mathbf{AA}^{\mathrm{T}}$).

$$\Sigma = \mathrm{diag}(\sigma) = \mathrm{diag}(\lambda)$$

SVD

- En el eigendecomposition buscábamos una factorización de una matriz cuadrada en la forma:

$$\mathbf{A} = \mathbf{V}\mathrm{diag}(\lambda)\mathbf{V}^{-1}$$

- En particular, la aplicábamos a matrices simétricas, siendo la decomposición:

$$\mathbf{A} = \mathbf{Q}\Lambda\mathbf{Q}^{\mathrm{T}}$$

Nota: es usual utilizar una versión reducida de SVD, the *Low-Rank* approximation, donde:

$$\mathbf{A} \in \mathrm{R}^{nxm}; \mathbf{U} \in \mathrm{R}^{nxr}; \mathbf{\Sigma} \in \mathrm{R}^{rxr}; \mathbf{V} \in \mathrm{R}^{mxr}$$

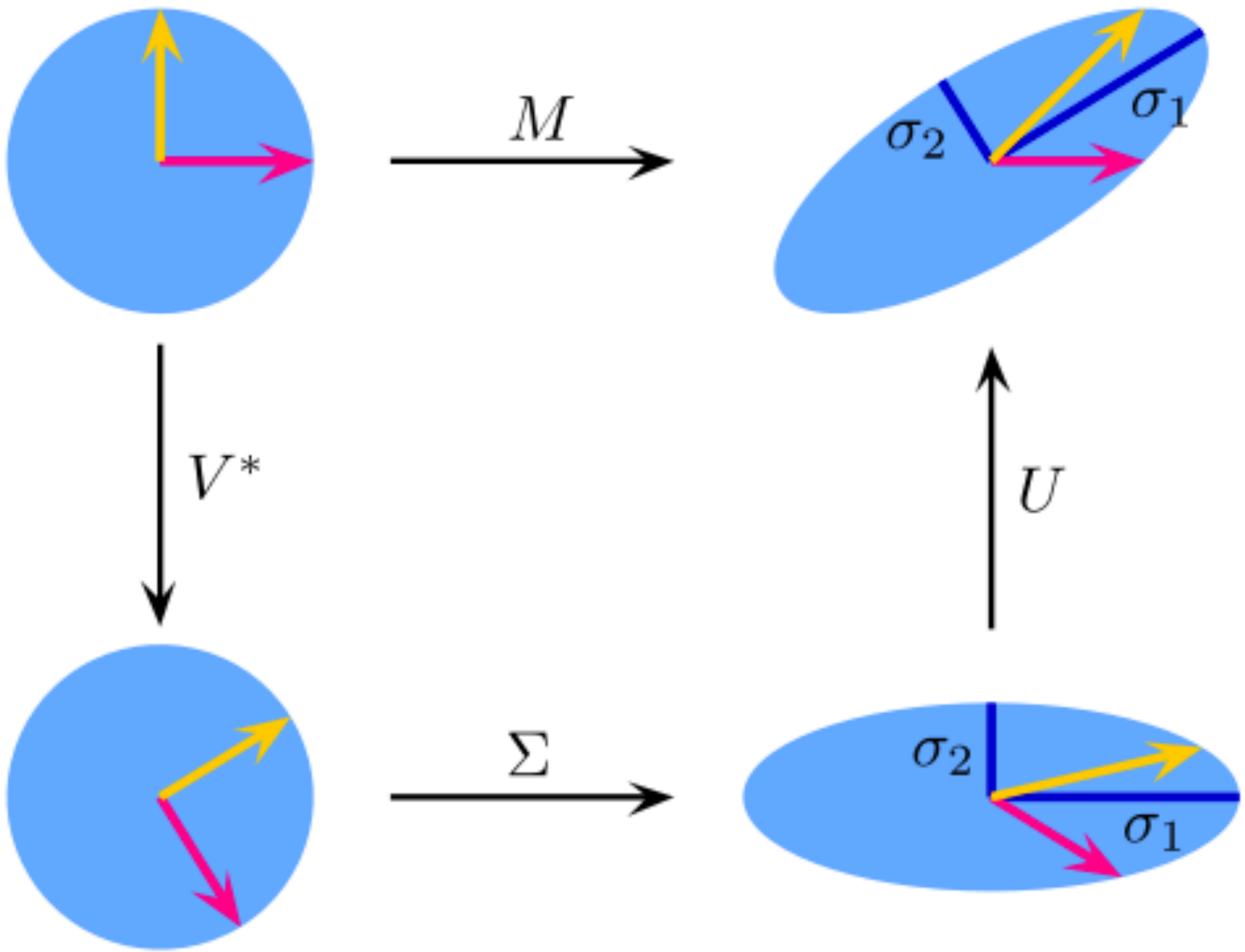
V es la matriz de autovectores de **A**$^{\mathrm{T}}$**A** (*right-singular vectors*):

$$\left(\mathbf{A}^{\mathrm{T}}\mathbf{A}\right)\mathbf{V}_{:,j} = \lambda_j \mathbf{V}_{:,j}$$

- Los valores singulares σ_i son la raíz cuadrada de los autovalores λ_i de la matriz $\mathbf{A}^{\mathrm{T}}\mathbf{A}$ (iguales a los de $\mathbf{A}\mathbf{A}^{\mathrm{T}}$).

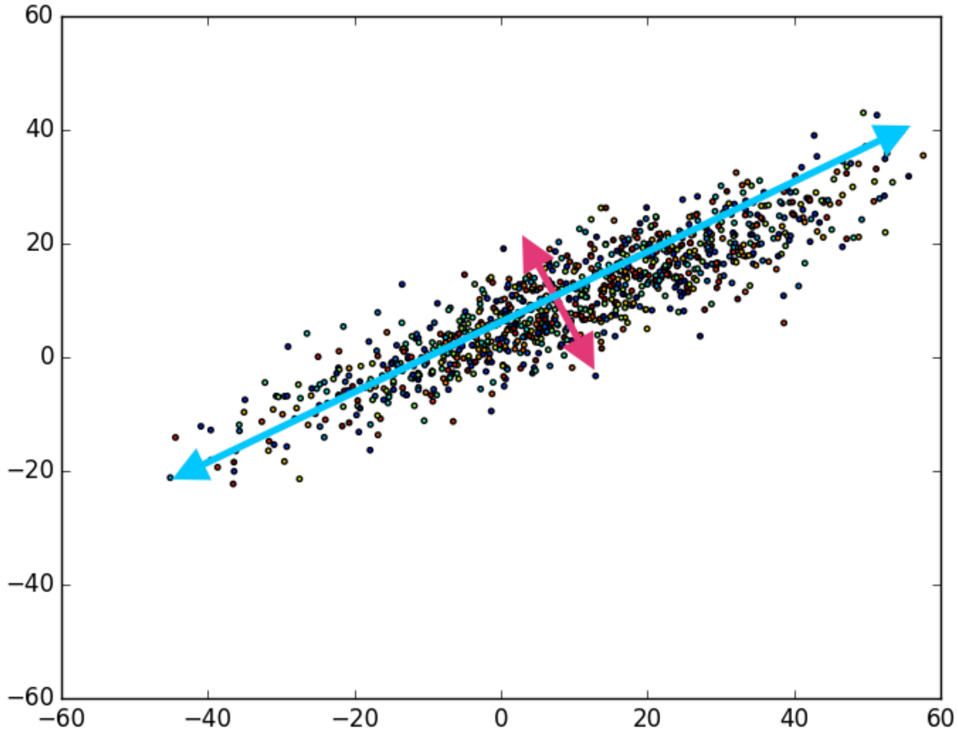
$$\Sigma = \mathrm{diag}(\sigma) = \mathrm{diag}(\lambda)$$

SVD



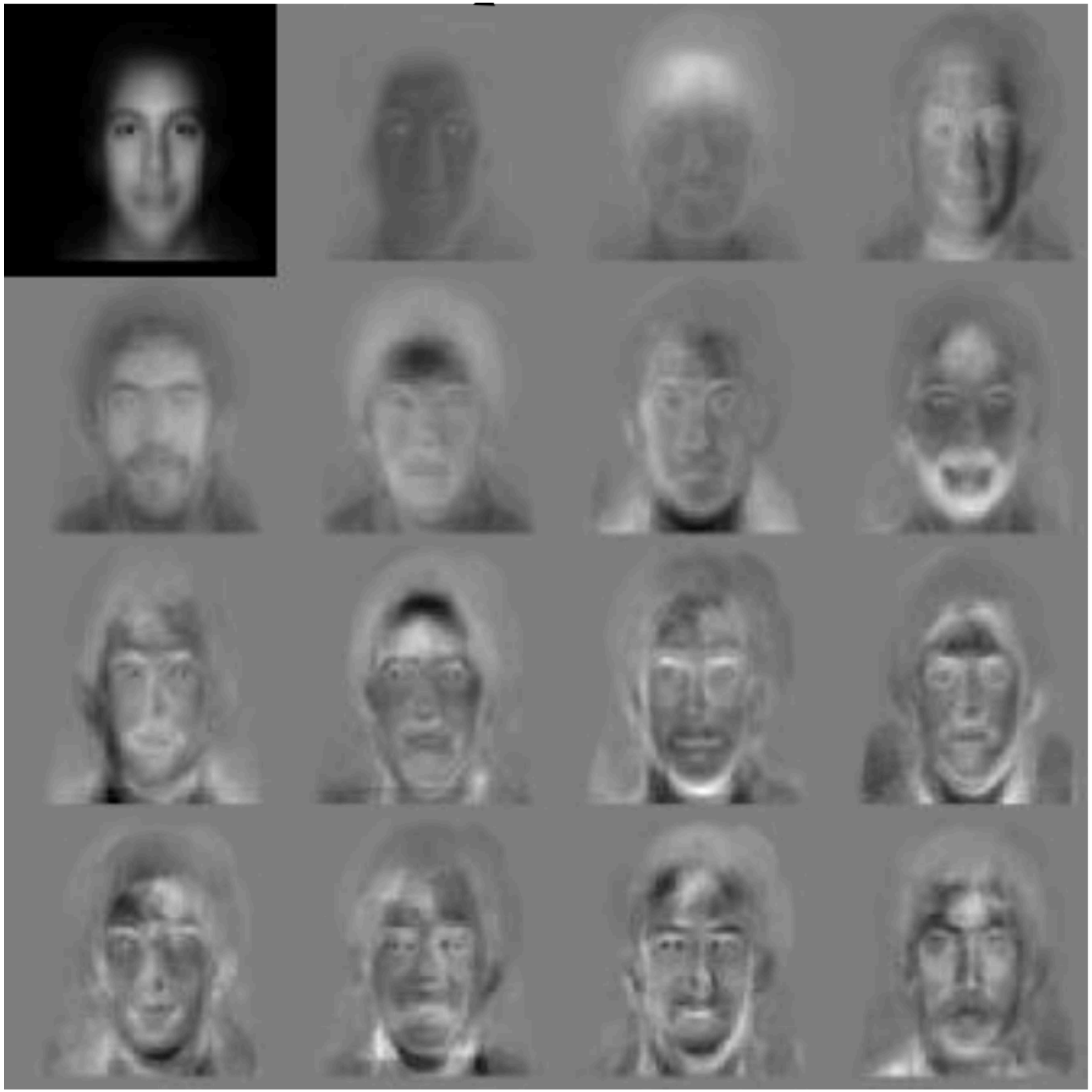
$$M = U \cdot \Sigma \cdot V^*$$

Applications of matrix decomposition

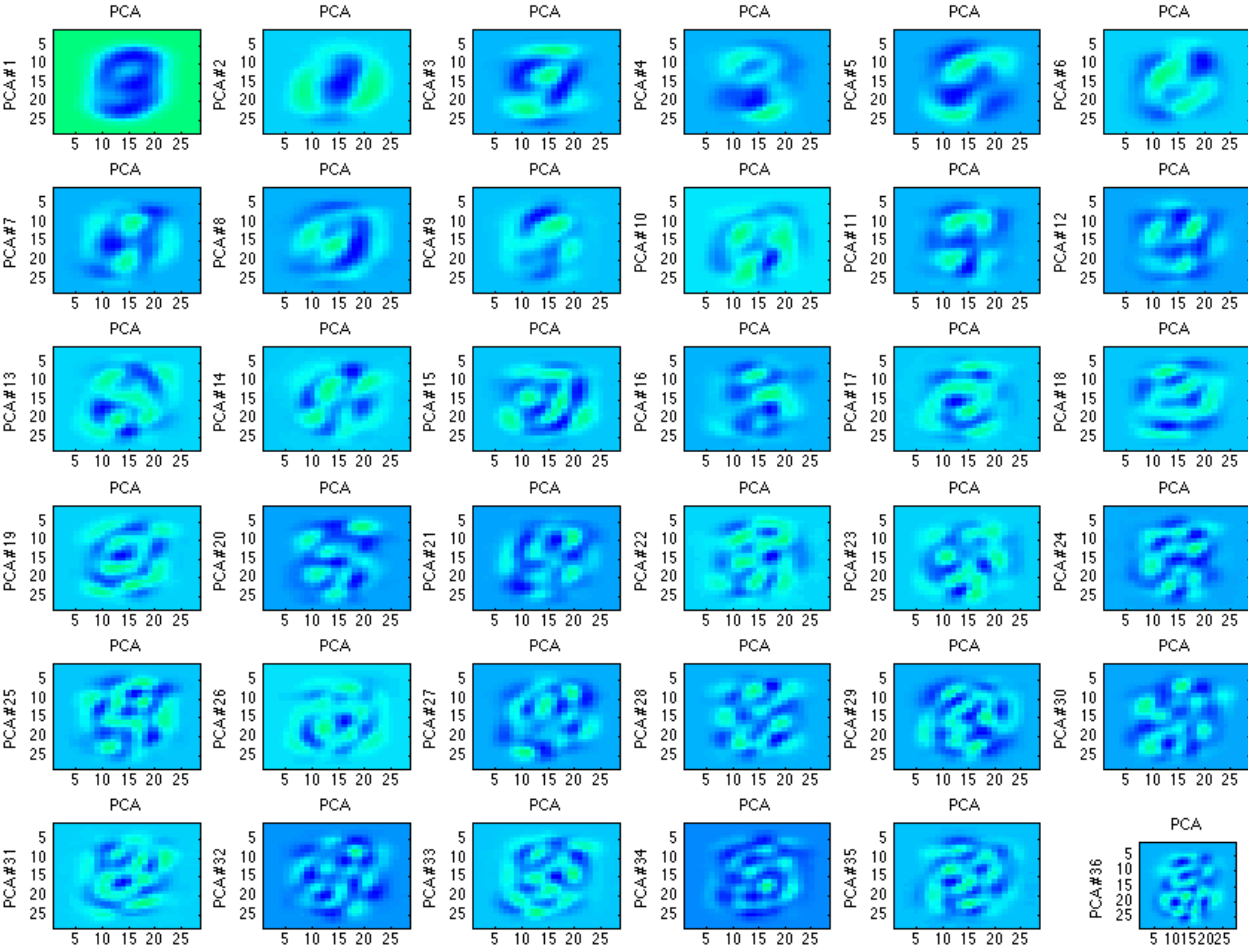


- SVD nos permite detectar las principales direcciones de simetría de nuestros problema (*singular vectors*), y reducirlas.

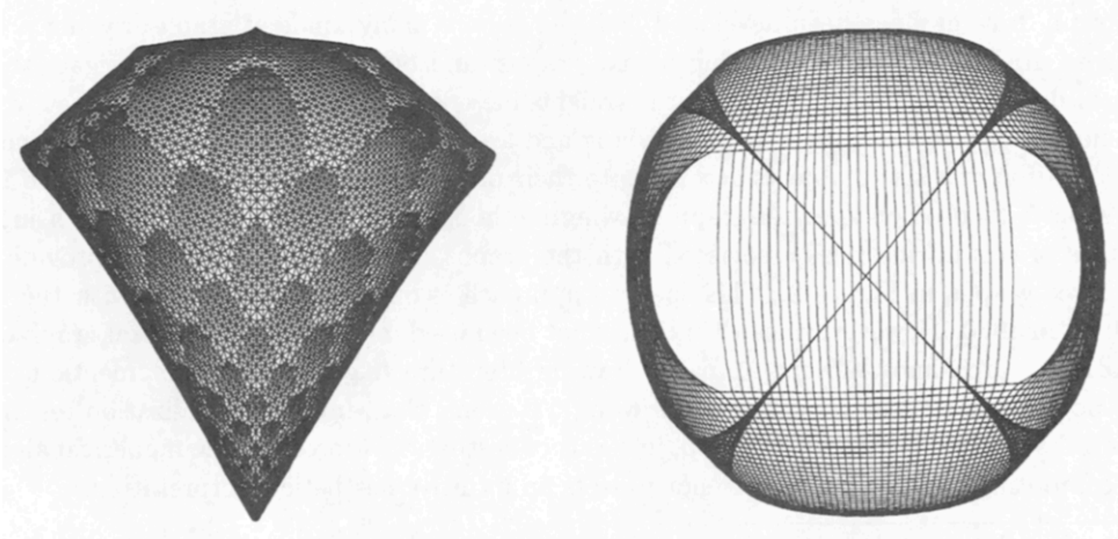
Applications of matrix decomposition



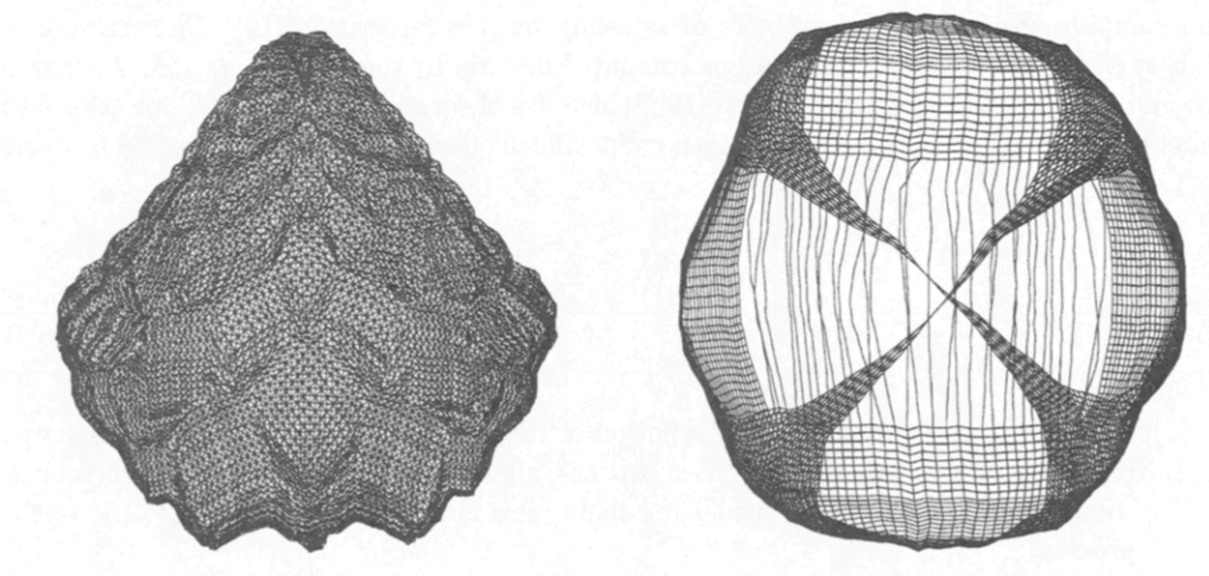
Applications of matrix decomposition



Applications of matrix decomposition



(c) The crack graph [10]. $|V| = 10,240$, $|E| = 30,380$. (d) A 100×100 folded grid with central horizontal edge removed. $|V| = 10,000$, $|E| = 18,713$.



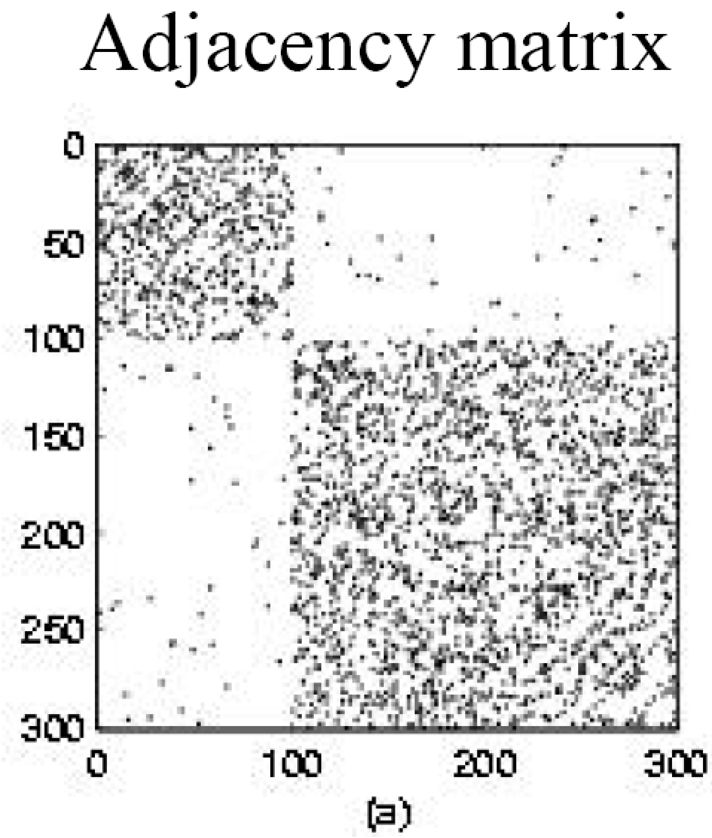
(c) The Crack graph [10] $|V| = 10,240$, $|E| = 30,380$ (d) A 100×100 folded grid with central horizontal edge removed. $|V| = 10,000$, $|E| = 18,713$

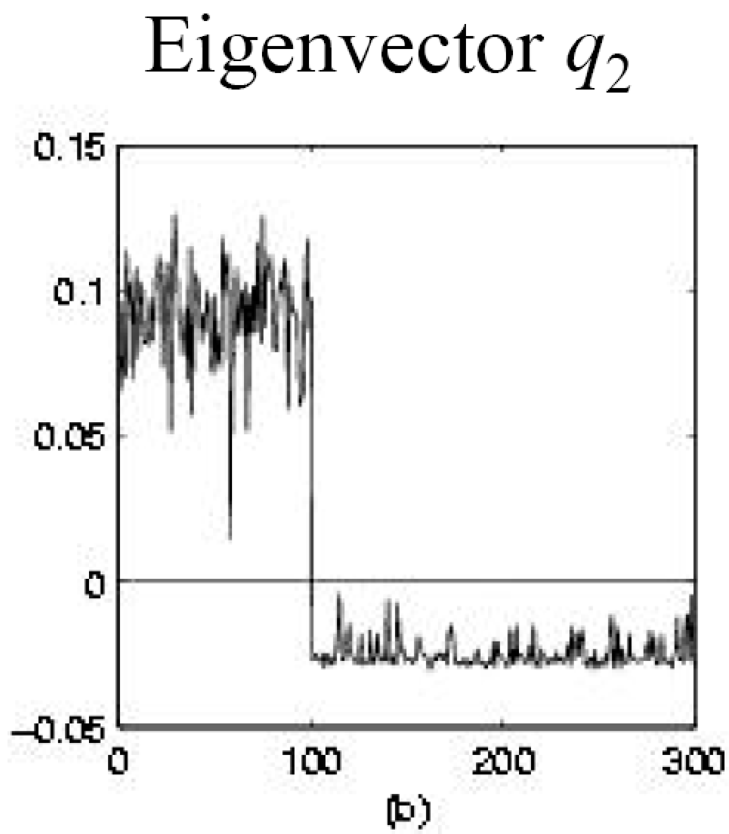
https://www.math.ucdavis.edu/~saito/data/acha.read.w12/koren-graph-drawing.pdf

Applications of matrix decomposition

Find 2nd eigenvector of graph Laplacian (think of it as adjacency) matrix

Cluster based on 2nd eigevector





Applications of matrix decomposition

Applications of matrix decomposition

users

items

1		3			5			5		4	
		5	4			4			2	1	3
2	4		1	2		3		4	3	5	
	2	4		5			4			2	
		4	3	4	2					2	5
1		3		3			2			4	

~

~ items

.1	-.4	.2
-.5	.6	.5
-.2	.3	.5
1.1	2.1	.3
-.7	2.1	-2
-1	.7	.3

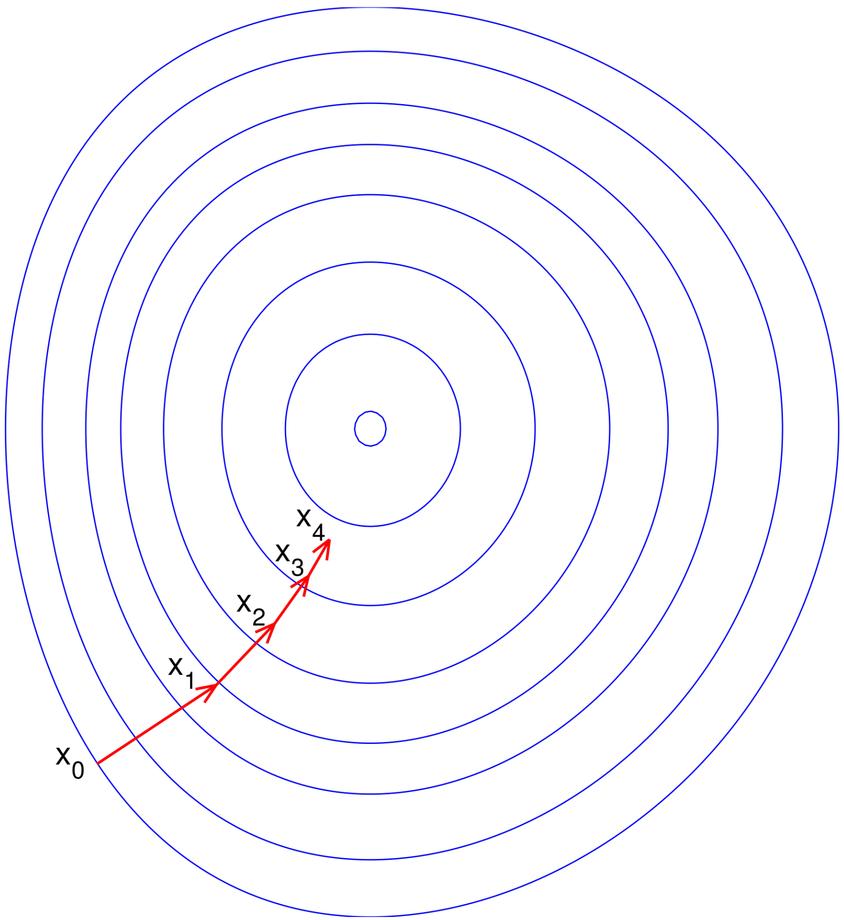
•

users

1.1	-.2	.3	.5	-2	-.5	.8	-.4	.3	1.4	2.4	-.9
-.8	.7	.5	1.4	.3	-1	1.4	2.9	-.7	1.2	-.1	1.3
2.1	-.4	.6	1.7	2.4	.9	-.3	.4	.8	.7	-.6	.1

A rank-3 SVD approximation

What´s next??



- Casi cualquier problema de **machine learning** se puede entender como una clasificación o una regresión.
- Ambos, están basados en la optimización (minimización) de un problema del tipo **Ax** =**b**.
- A la factorización de la matriz **A** (eiegndecomposition, SVD, PCA, LU, etc.) hay que añadirle la forma de minimizar el error en el resultado: **Gradient descent.**

Algunos enlaces de interés

- Deep learning:
  - http://goodfeli.github.io/dlbook/
  - http://deeplearning.net/

- Machine Learning:
  - Wikipedia!!!!! Buscar supervised, unsupervised, classification, regression, collaborative filtering, gradient descent, etc.
    Y recordad buscar en inglés ;)
  - Curso clásico de Coursera: https://www.coursera.org/learn/machine-learning
  - Libro *Mining of massive datasets*: http://www.mmds.org/

- Google ML library, *Tensor flow*: https://www.tensorflow.org/

- Una introducción a grafos que me pareció curiosa:
  http://20bits.com/article/graph-theory-part-i-introduction/

- Otra, relacionando lo que hemos visto con Google y Facebook, muy chula ☺
  https://prezi.com/knv3fxe7lytv/graph-theory-facebook/