Evolutive EigenGame: Resolviendo PCA con un algoritmo evolutivo

Tesis de Licenciatura

Ezequiel Diego Cribioli

Dirección: Dr. Juan Pablo Pinasco

Departamento de Matematica - FCEyN - UBA

12 de agosto de 2022

Agradecimientos

Plan

Motivación

Motivación

- Imaginemos que estamos trabajando en una herramienta en la que tenemos que lidiar con una base de datos de fotos de perritos.

- Una más....

- Cada foto es un dato muy grande. Por ejemplo, si están en calidad 1080p, tenemos $1920 \times 1080 \times 3 = 6,220,800$ números.

- Cada foto es un dato muy grande. Por ejemplo, si están en calidad 1080p, tenemos $1920 \times 1080 \times 3 = 6,220,800$ números.

- A lo mejor podemos describir las fotos con muchas menos variables.

- Cada foto es un dato muy grande. Por ejemplo, si están en calidad 1080p, tenemos $1920 \times 1080 \times 3 = 6,220,800$ números.
- A lo mejor podemos describir las fotos con muchas menos variables.
- ¿Cuántos parámetros necesitamos para describir la cara de un perro ?

$\rightarrow$

tamaño de las orejas :	1
largo de la trompa :	3
largo del pelo :	7
...	...
ternura :	100

- Hacer esto a mano no es una buena idea.

- Hacer esto a mano no es una buena idea.
- Necesitamos una forma **inteligente** de comprimir las fotos en menos variables, donde por inteligente nos referimos a :

- Hacer esto a mano no es una buena idea.
- Necesitamos una forma **inteligente** de comprimir las fotos en menos variables, donde por inteligente nos referimos a :
    1. Automatizada : Más rápido y sin errores.

- Hacer esto a mano no es una buena idea.
- Necesitamos una forma **inteligente** de comprimir las fotos en menos variables, donde por inteligente nos referimos a :
  1. Automatizada : Más rápido y sin errores.
  2. Fiel : Que no perdamos mucha información al convertir los datos.

- Recortar las muestras quedándonos con algunas de las $6,220,800$ variables no sirve :

- Recortar las muestras quedándonos con algunas de las $6,220,800$ variables no sirve :

- Recortar las muestras quedándonos con algunas de las $6,220,800$ variables no sirve :

$\rightarrow$

- Es por esto que se estudian las llamadas **técnicas de reducción de dimensión**, que nos ayudan en esta clase de problemas.

Análisis de Componentes Principales

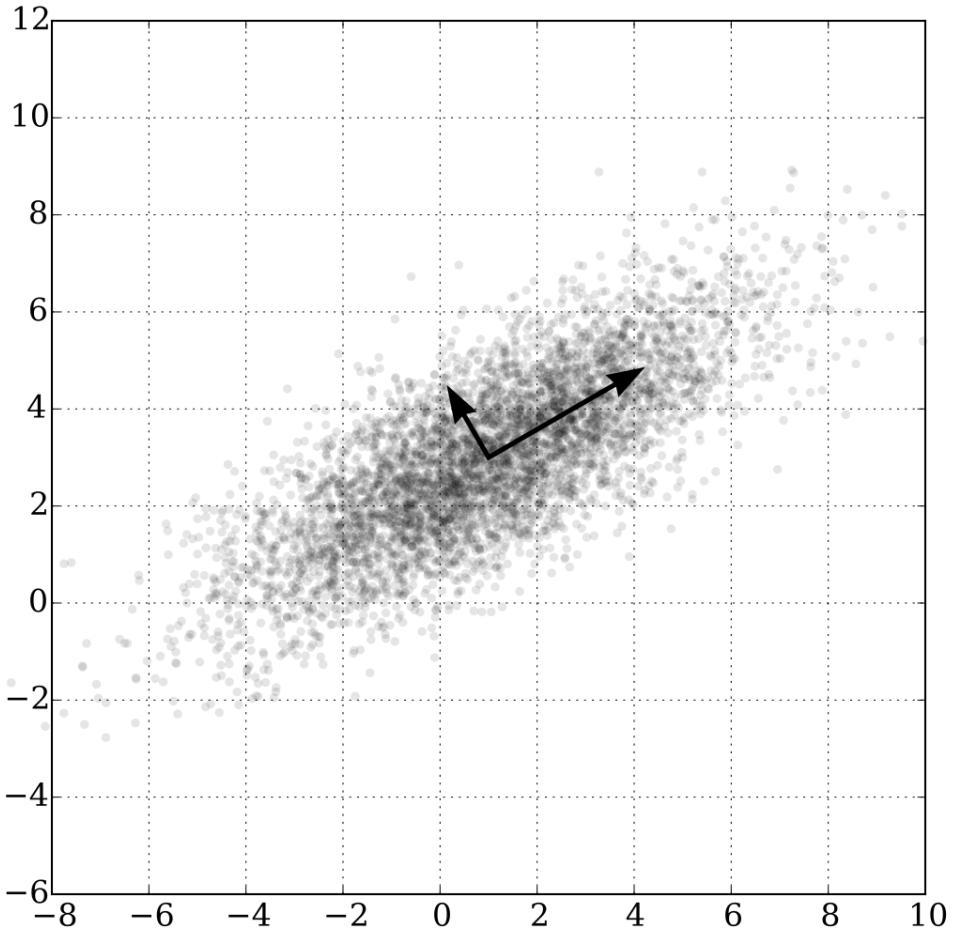
PCA

- El *Análisis de Componentes Principales* (PCA por sus siglas en inglés) es una de estas técnicas de reducción de dimensión que busca identificar las direcciones que mejor describen nuestro set de datos según cómo estos se correlacionen.

PCA

- El *Análisis de Componentes Principales* (PCA por sus siglas en inglés) es una de estas técnicas de reducción de dimensión que busca identificar las direcciones que mejor describen nuestro set de datos según cómo estos se correlacionen.

- Estas direcciones se denominan "componentes" y van a ser las nuevas variables en las que vamos a describir nuestro dataset.


12
10
8
6
4
2
0
−2
−4
−6
−8
−6
−4
−2
0
2
4
6
8
10

Método

- Si nuestras muestras son un conjunto de k puntos en n dimensiones $\{x_i\}_{i=1}^{k} \subset \mathbb{R}^n$, consideramos la matriz $X \in \mathbb{R}^{k \times n}$ cuyas filas son cada una de nuestras k muestras.

Método

- Si nuestras muestras son un conjunto de k puntos en n dimensiones $\{x_i\}_{i=1}^{k} \subset \mathbb{R}^n$, consideramos la matriz $X \in \mathbb{R}^{k \times n}$ cuyas filas son cada una de nuestras k muestras.

- Con esta matriz X, calculamos la *matriz de covarianza*

$$M = X^T X \in \mathbb{R}^{n \times n}$$

que nos dice cuán relacionadas están las muestras entre ellas.

Método

- Si nuestras muestras son un conjunto de k puntos en n dimensiones $\{x_i\}_{i=1}^{k} \subset \mathbb{R}^n$, consideramos la matriz $X \in \mathbb{R}^{k \times n}$ cuyas filas son cada una de nuestras k muestras.

- Con esta matriz X, calculamos la *matriz de covarianza*

$$M = X^T X \in \mathbb{R}^{n \times n}$$

que nos dice cuán relacionadas están las muestras entre ellas.

- Esto es asi ya que $M_{ij} = \langle x_i, x_j \rangle$, que nos dice cuán alineados están x_i y x_j.

- **Teorema :** Esta matriz M es simétrica y semi-definida positiva, por lo que existe una base $\{v_1, v_2, \cdots, v_n\}$ tal que $M \cdot v_i = \lambda_i v_i$ con $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n \geq 0$.

- **Teorema :** Esta matriz M es simétrica y semi-definida positiva, por lo que existe una base $\{v_1, v_2, \cdots, v_n\}$ tal que $M \cdot v_i = \lambda_i v_i$ con $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n \geq 0$.

- PCA consiste en encontrar estos autovectores, que son las *componentes*. Cuanto más grande es λ_i, más información aporta v_i, por lo que vamos a querer encontrar a las componentes en orden.

- **Teorema :** Esta matriz M es simétrica y semi-definida positiva, por lo que existe una base $\{v_1, v_2, \cdots, v_n\}$ tal que $M \cdot v_i = \lambda_i v_i$ con $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n \geq 0$.

- PCA consiste en encontrar estos autovectores, que son las *componentes*. Cuanto más grande es λ_i, más información aporta v_i, por lo que vamos a querer encontrar a las componentes en orden.

- No vamos a querer encontrar todas las componentes, sino unas pocas que aporten mucha información (las primeras).

Ejemplo

- Se realizó PCA a las calificaciones escolares de 15 estudiantes en 8 materias : lengua, matemáticas, física, inglés, filosofía, historia, química, gimnasia.

Ejemplo

- Se realizó PCA a las calificaciones escolares de 15 estudiantes en 8 materias : lengua, matemáticas, física, inglés, filosofía, historia, química, gimnasia.

- Las dos primeras componentes principales explicaban juntas el 82,1% de la varianza :

Ejemplo

- Se realizó PCA a las calificaciones escolares de 15 estudiantes en 8 materias : lengua, matemáticas, física, inglés, filosofía, historia, química, gimnasia.

- Las dos primeras componentes principales explicaban juntas el 82,1% de la varianza : Humanidades (lengua, inglés, filosofía, historia)

Ejemplo

- Se realizó PCA a las calificaciones escolares de 15 estudiantes en 8 materias : lengua, matemáticas, física, inglés, filosofía, historia, química, gimnasia.

- Las dos primeras componentes principales explicaban juntas el 82,1% de la varianza : Humanidades (lengua, inglés, filosofía, historia) y Ciencias (matemáticas, física, química).

Ejemplo

- Se realizó PCA a las calificaciones escolares de 15 estudiantes en 8 materias : lengua, matemáticas, física, inglés, filosofía, historia, química, gimnasia.
- Las dos primeras componentes principales explicaban juntas el 82,1% de la varianza : Humanidades (lengua, inglés, filosofía, historia) y Ciencias (matemáticas, física, química).
- Estos dos conjuntos de habilidades son estadísticamente independientes.

EigenGame

- Para hacer PCA, necesitamos calcular los autovectores de una matriz. Y para eso se recurre a algoritmos de aproximación.

- Para hacer PCA, necesitamos calcular los autovectores de una matriz. Y para eso se recurre a algoritmos de aproximación.

- Acá es donde entra el concepto de EigenGame, introducido en 2021 por un grupo de investigadores de DeepMind.

- Para hacer PCA, necesitamos calcular los autovectores de una matriz. Y para eso se recurre a algoritmos de aproximación.
- Acá es donde entra el concepto de EigenGame, introducido en 2021 por un grupo de investigadores de DeepMind.
- En vez de intentar resolver el problema directamente, verlo como un juego de múltiples jugadores.

- Para entender esto, necesitamos conocer el concepto de **Equilibrio de Nash :**

- Para entender esto, necesitamos conocer el concepto de **Equilibrio de Nash :**
  - Dado un juego de 2 o más jugadores, un conjunto de estrategias para los jugadores es un *Equilibrio de Nash* si cada jugador tiene una estrategia óptima dada la de sus rivales. En otras palabras, es una situación en la que ninguno de los jugadores tiene incentivo a cambiar.

- Para entender esto, necesitamos conocer el concepto de **Equilibrio de Nash :**

  - Dado un juego de 2 o más jugadores, un conjunto de estrategias para los jugadores es un *Equilibrio de Nash* si cada jugador tiene una estrategia óptima dada la de sus rivales. En otras palabras, es una situación en la que ninguno de los jugadores tiene incentivo a cambiar.

- Los equilibrios de Nash están presentes es muchos modelos muy usados (por ejemplo la economía) ya que resultan muy naturales.

Dilema del prisionero

	Yo confieso	**Yo no confieso**
Él confiesa	6 años para ambos	10 años para mí y 0 para él
Él no confiesa	10 años para él y 0 para mí	1 año para ambos

- Volviendo al EigenGame : imaginemos que queremos encontrar las primeras d componentes de una matriz M. Planteamos el siguiente juego de d jugadores :

- Volviendo al EigenGame : imaginemos que queremos encontrar las primeras d componentes de una matriz M. Planteamos el siguiente juego de d jugadores :
  - Cada jugador tiene un vector $\hat{v_i}$ de norma 1 y una *función de utilidad* u_i que depende de los vectores en juego, y el objetivo de cada jugador es maximizar su función utilidad.

- Volviendo al EigenGame : imaginemos que queremos encontrar las primeras d componentes de una matriz M. Planteamos el siguiente juego de d jugadores :

  - Cada jugador tiene un vector $\hat{v_i}$ de norma 1 y una *función de utilidad* u_i que depende de los vectores en juego, y el objetivo de cada jugador es maximizar su función utilidad.

  - En cada turno, todos los jugadores modifican sus vectores al mismo tiempo para que cada jugador busque que la función u_i sea lo más grande posible.

- Volviendo al EigenGame : imaginemos que queremos encontrar las primeras d componentes de una matriz M. Planteamos el siguiente juego de d jugadores :

  - Cada jugador tiene un vector $\hat{v}_i$ de norma 1 y una *función de utilidad* u_i que depende de los vectores en juego, y el objetivo de cada jugador es maximizar su función utilidad.

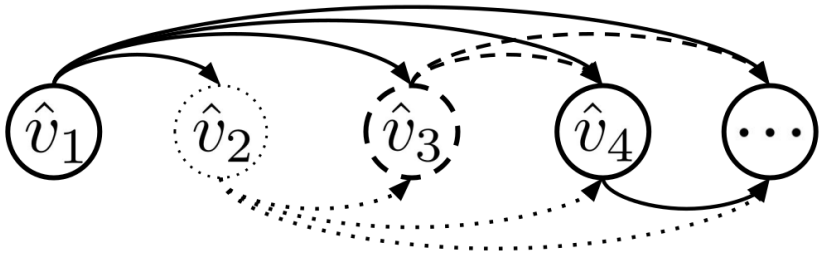
  - En cada turno, todos los jugadores modifican sus vectores al mismo tiempo para que cada jugador busque que la función u_i sea lo más grande posible.

  - Para el i-esimo jugador, su función es

$$u_i(\hat{v}_i \mid \hat{v}_{j<i}) = \langle \hat{v}_i, M\hat{v}_i \rangle - \sum_{j<i} \frac{\langle \hat{v}_i, M\hat{v}_j \rangle^2}{\langle \hat{v}_j, M\hat{v}_j \rangle}$$

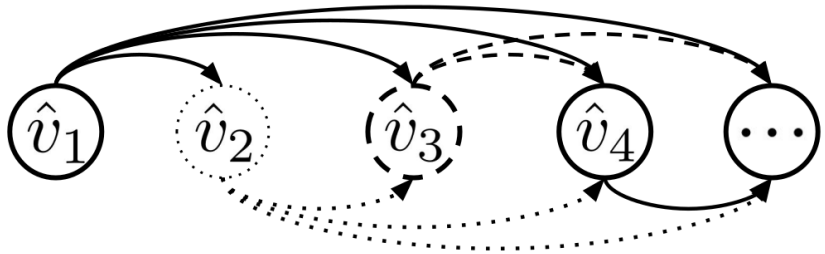
Observaciones

- u_i depende únicamente de los primeros i vectores. A esto se lo conoce como *jerarquía* (!).



Observaciones

- u_i depende únicamente de los primeros i vectores. A esto se lo conoce como *jerarquía* (!).



- El equilibrio de Nash del EigenGame es **único** y se da cuando $\hat{v_1} = v_1$, $\hat{v_2} = v_2$, $\cdots$, $\hat{v_d} = v_d$. Esta es justamente la razón por la que consideramos el juego.

Algoritmo Evolutivo

Diagrama

- Recapitulemos en donde estamos parados.

Diagrama

- Recapitulemos en donde estamos parados.
  - Nuestra motivación era estudiar una técnica de reducción de dimensión (PCA).

Diagrama

- Recapitulemos en donde estamos parados.
  - Nuestra motivación era estudiar una técnica de reducción de dimensión (PCA).
  - Para hacer PCA, tenemos que encontrar los primeros d autovectores de una matriz.

Diagrama

- Recapitulemos en donde estamos parados.
  - Nuestra motivación era estudiar una técnica de reducción de dimensión (PCA).
  - Para hacer PCA, tenemos que encontrar los primeros d autovectores de una matriz.
  - Para encontrar los primeros d autovectores, queremos encontrar un equilibrio de Nash en un juego (EigenGame).

Diagrama

- Recapitulemos en donde estamos parados.
  - Nuestra motivación era estudiar una técnica de reducción de dimensión (PCA).
  - Para hacer PCA, tenemos que encontrar los primeros d autovectores de una matriz.
  - Para encontrar los primeros d autovectores, queremos encontrar un equilibrio de Nash en un juego (EigenGame).
- Encontrar equilibrios de Nash es muy difícil en general. En este trabajo nos dedicamos a diseñar y estudiar un algoritmo para eso.

- Existen algoritmos basados en el método del Ascenso del Gradiente.

- Existen algoritmos basados en el método del Ascenso del Gradiente.

- Nosotros proponemos un algoritmo del tipo *evolutivo*. Esta perspectiva nos va a ofrecer una alternativa sin muchas de las desventajas de los algoritmos anteriores.

- Los algoritmos evolutivos son métodos basados en los postulados de la evolución biológica. En ellos se genera un conjunto de variantes de un estado, las cuales se mezclan, y compiten entre sí, sobreviviendo solamente la mejor.

- Los algoritmos evolutivos son métodos basados en los postulados de la evolución biológica. En ellos se genera un conjunto de variantes de un estado, las cuales se mezclan, y compiten entre sí, sobreviviendo solamente la mejor.

- Por ejemplo, se puede usar uno de estos algoritmos para encontrar la mejor forma de preparar un budín.

- Un dato clave para nuestra estrategia es la ya mencionada jerarquía.

- Un dato clave para nuestra estrategia es la ya mencionada jerarquía.

- Si nos centramos en el primer jugador, su función de utilidad u_1 depende solo de sí mismo.

- Un dato clave para nuestra estrategia es la ya mencionada jerarquía.
- Si nos centramos en el primer jugador, su función de utilidad u_1 depende solo de sí mismo.
- Entonces, si queremos encontrar el equilibrio de Nash del juego, primero debemos encontrar el vector v_1 que maximiza u_1.

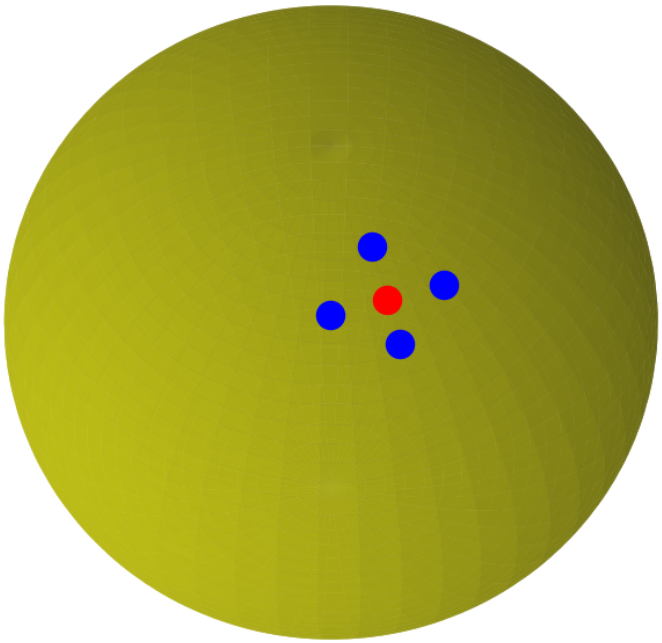
- Ahora la función u_2 depende solo de $\hat{v_1}$ y de $\hat{v_2}$. Pero ya sabemos que $\hat{v_1}$ tiene que ser el óptimo de u_1. Entonces, conociendo v_1 del paso anterior, podemos buscar v_2 como el óptimo de u_2 considerando $\hat{v_1} = v_1$.

- Ahora la función u_2 depende solo de $\hat{v_1}$ y de $\hat{v_2}$. Pero ya sabemos que $\hat{v_1}$ tiene que ser el óptimo de u_1. Entonces, conociendo v_1 del paso anterior, podemos buscar v_2 como el óptimo de u_2 considerando $\hat{v_1} = v_1$.

- Así siguiendo, cuando queramos calcular v_j ya vamos a tener calculados todos los vectores anteriores. Por lo tanto, encontrar los vectores es simplemente encontrar óptimos de funciones.

- En este caso, el espacio donde estamos buscando valores óptimos es una esfera. Entonces, estando parados en un punto de la esfera, vamos a movernos una cierta distancia *step* en ciertas direcciones, y quedarnos con el punto que tenga la mayor utilidad.

- En este caso, el espacio donde estamos buscando valores óptimos es una esfera. Entonces, estando parados en un punto de la esfera, vamos a movernos una cierta distancia *step* en ciertas direcciones, y quedarnos con el punto que tenga la mayor utilidad.



- Al principio, esta distancia *step* va a ser muy grande, para abarcar la mayor parte de la esfera.

- Al principio, esta distancia *step* va a ser muy grande, para abarcar la mayor parte de la esfera.

- Pero si ninguno de los puntos nuevos resulta mejor que en el que ya estábamos, entonces reducimos la distancia *step*.

- Al principio, esta distancia *step* va a ser muy grande, para abarcar la mayor parte de la esfera.
- Pero si ninguno de los puntos nuevos resulta mejor que en el que ya estábamos, entonces reducimos la distancia *step*.
- El espíritu es “si ningún punto te sirve, movete menos”.

- Por este motivo, el valor de *step* nos va a servir de termómetro.

- Por este motivo, el valor de *step* nos va a servir de termómetro.

- Como regla, el algoritmo termina cuando *step* empieza a ser menor que cierto valor ε elegido. Si queremos un resultado con más precisión, establecemos una cota inferior ε menor.

Performance

- ¿Cuánto tarda el algoritmo en encontrar una componente ? Es fácil calcular cuanto va a tardar el algoritmo en obtener la utilidad de algún vector.

Performance

- ¿Cuánto tarda el algoritmo en encontrar una componente ? Es fácil calcular cuanto va a tardar el algoritmo en obtener la utilidad de algún vector.

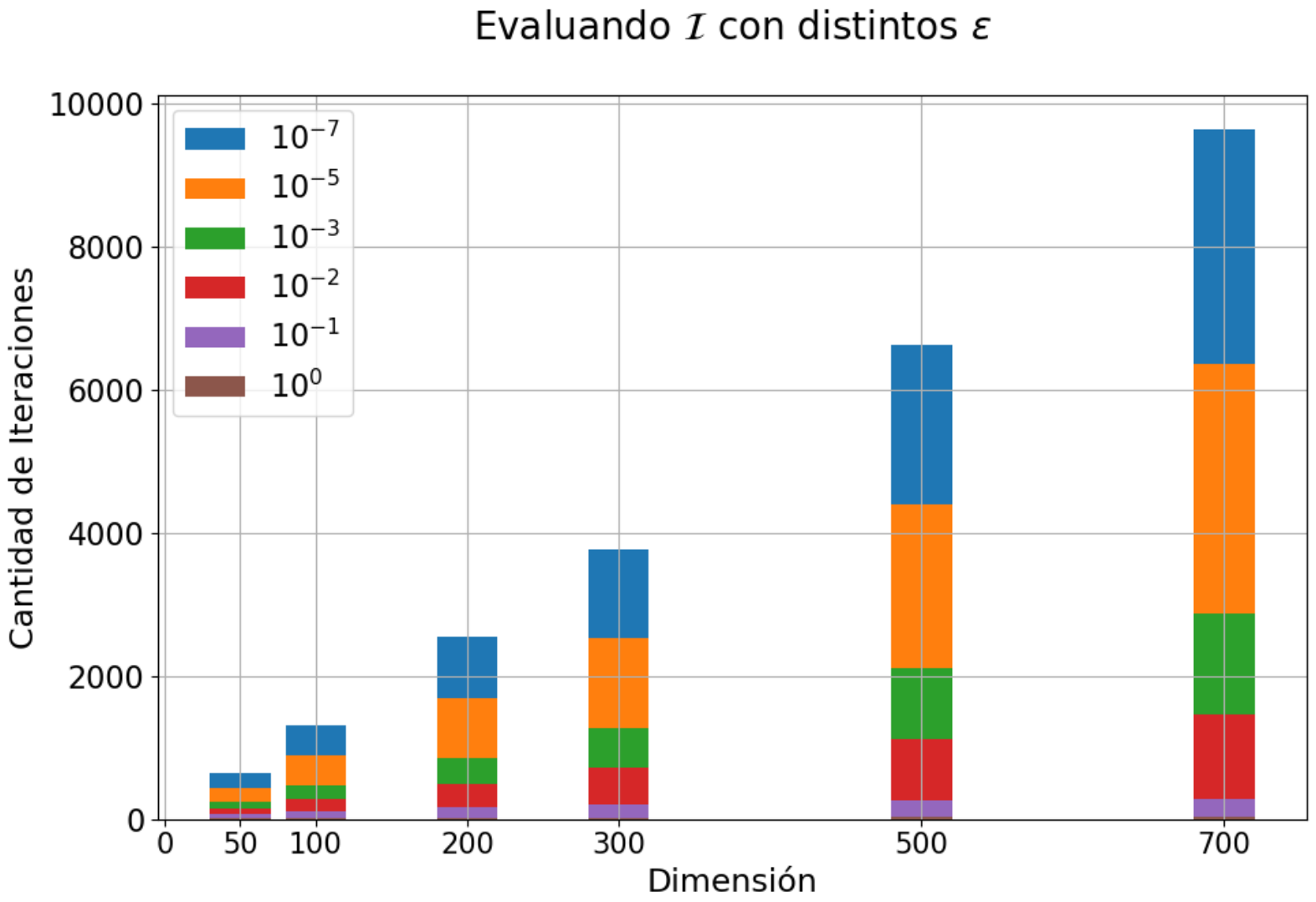
- Lo que si merece la pena estudiar es la cantidad de iteraciones que hacen falta para que el algoritmo termine.

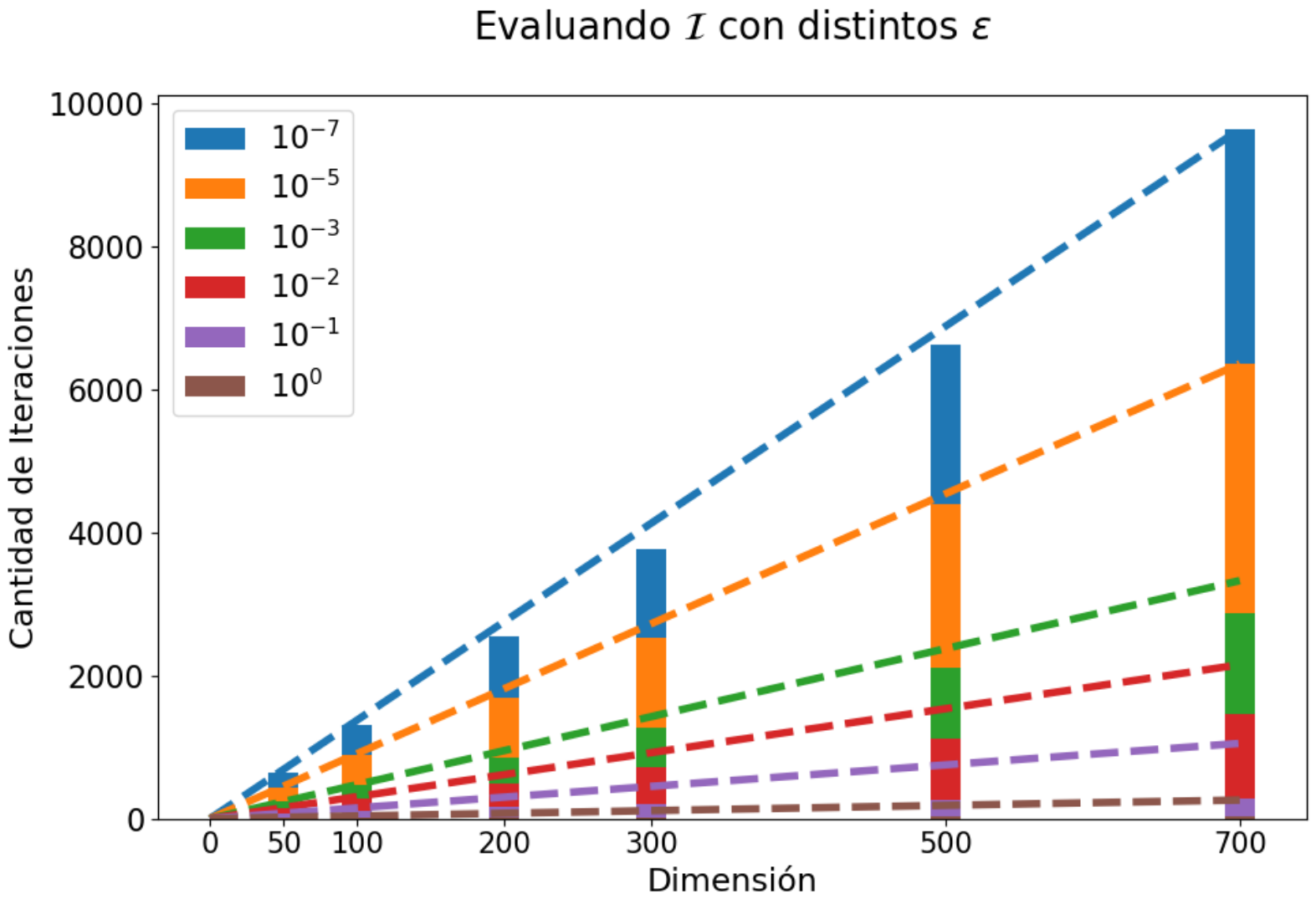
- Es decir, la cantidad de "me muevo un poco en distintas direcciones y me quedo con el mejor vector" necesarios para que el parámetro *step* sea menor que ε. Vamos a llamar a esta cantidad $\mathcal{I}$.

- En uno de los experimentos, se analizó como evoluciona $\mathcal{I}$ en función de la dimensión inicial de los datos (n).

- En uno de los experimentos, se analizó como evoluciona $\mathcal{I}$ en función de la dimensión inicial de los datos (n).

- Veamos un gráfico, resultado de la experimentación, que muestra esta evolución ante distintos n y distintos valores de parada ε.


Evaluando $\mathcal{I}$ con distintos ε
Cantidad de Iteraciones
Dimensión
10^{-7}
10^{-5}
10^{-3}
10^{-2}
10^{-1}
10^{0}
0
2000
4000
6000
8000
10000
0
50
100
200
300
500
700


Evaluando $\mathcal{I}$ con distintos ε
Cantidad de Iteraciones
Dimensión
10^{-7}
10^{-5}
10^{-3}
10^{-2}
10^{-1}
10^{0}
0
2000
4000
6000
8000
10000
0
50
100
200
300
500
700

Ventajas

Los algoritmos de los artículos anteriores tenían dos desventajas, que son a su vez puntos fuertes de nuestro algoritmo :

Ventajas

Los algoritmos de los artículos anteriores tenían dos desventajas, que son a su vez puntos fuertes de nuestro algoritmo :

- La primera y la más significativa era que la performance de métodos anteriores dependía fuertemente del vector inicial.

Ventajas

Los algoritmos de los artículos anteriores tenían dos desventajas, que son a su vez puntos fuertes de nuestro algoritmo :

- La primera y la más significativa era que la performance de métodos anteriores dependía fuertemente del vector inicial.
- Nuestro algoritmo ha demostrado en las experimentaciones no tener esta desventaja, ya que su performance es bastante uniforme respecto al vector inicial gracias al valor inicial de *step*.

- La segunda desventaja reside en que los métodos anteriores usaban propiedades propias de las funciones u_i para funcionar.

- La segunda desventaja reside en que los métodos anteriores usaban propiedades propias de las funciones u_i para funcionar.

- Mientras que nuestro algoritmo funciona bien para calcular el máximo de cualquier función en la esfera, lo cual siempre es una ventaja.

Una Aplicación

- Si bien usamos nuestro algoritmo *Evolutive EigenGame* dentro del contexto de PCA, resuelve un problema más general.

- Si bien usamos nuestro algoritmo *Evolutive EigenGame* dentro del contexto de PCA, resuelve un problema más general.

- Esto es algo útil en general, por lo que decidimos usar esta idea para resolver una ecuación diferencial.

- Si bien usamos nuestro algoritmo *Evolutive EigenGame* dentro del contexto de PCA, resuelve un problema más general.

- Esto es algo útil en general, por lo que decidimos usar esta idea para resolver una ecuación diferencial.

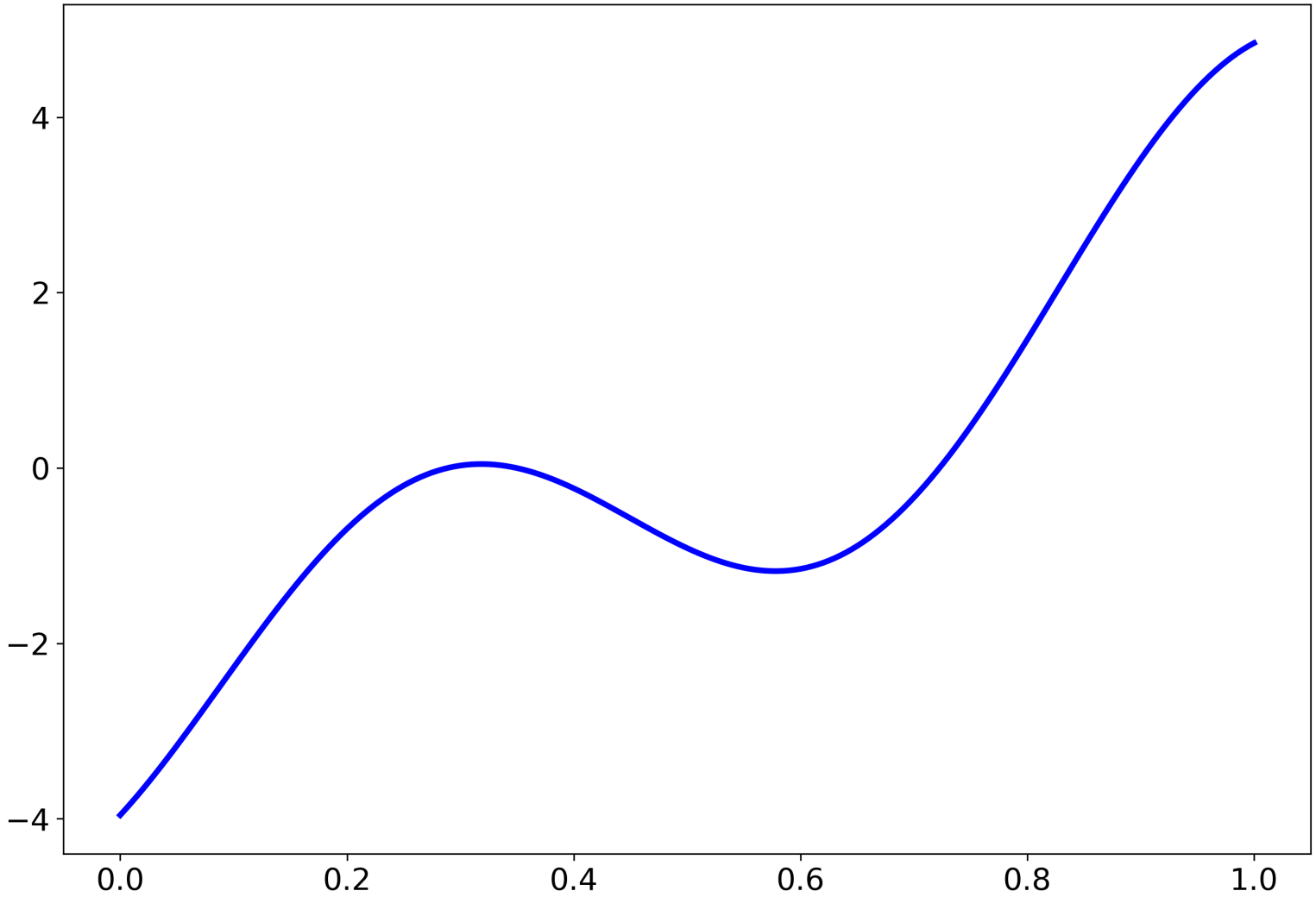
- Concretamente, la ecuación

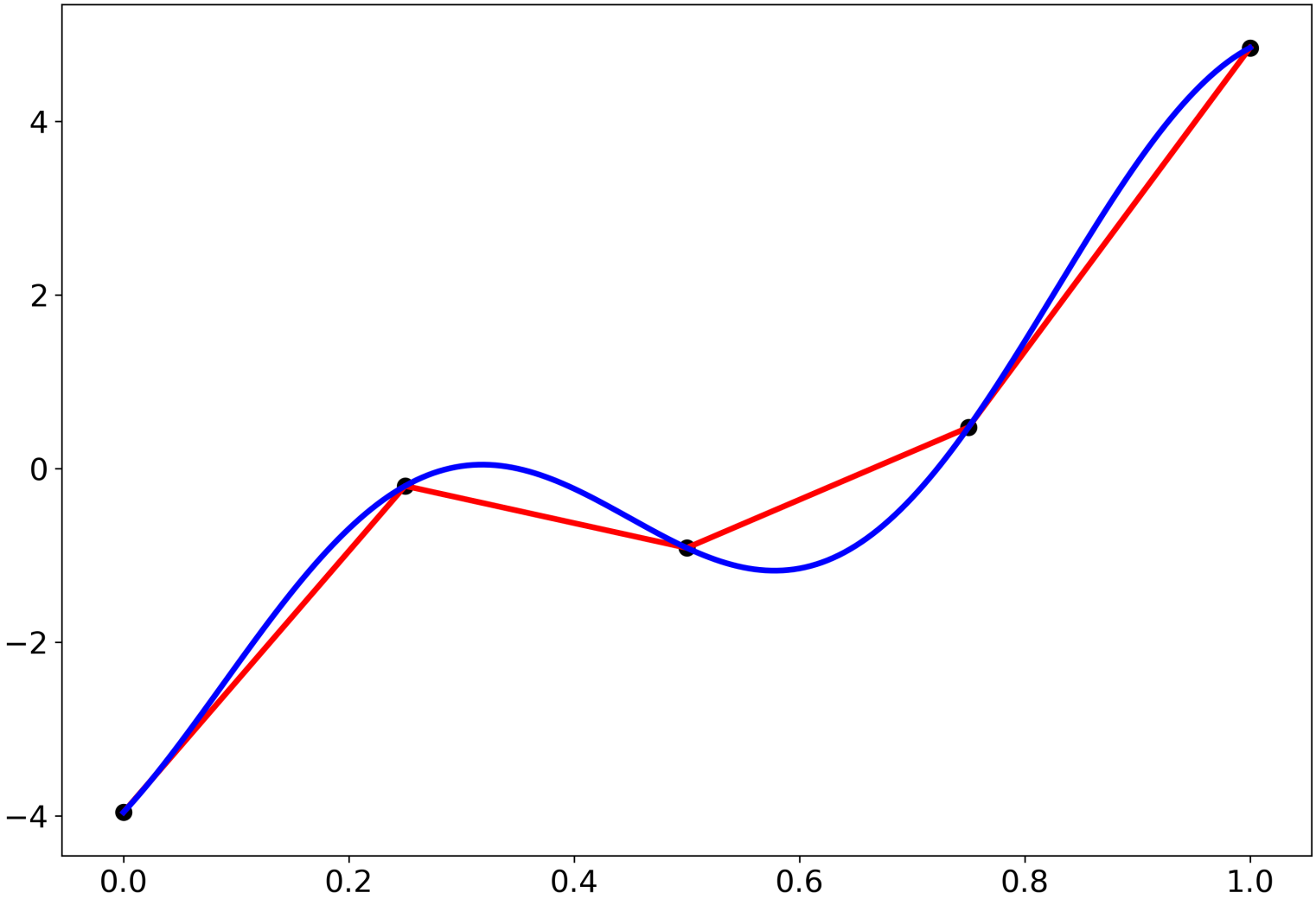
$$\begin{cases} -u''(x) = (\lambda + V(x)) \cdot u(x) \\ u(0) = u(1) = 0 \end{cases}$$

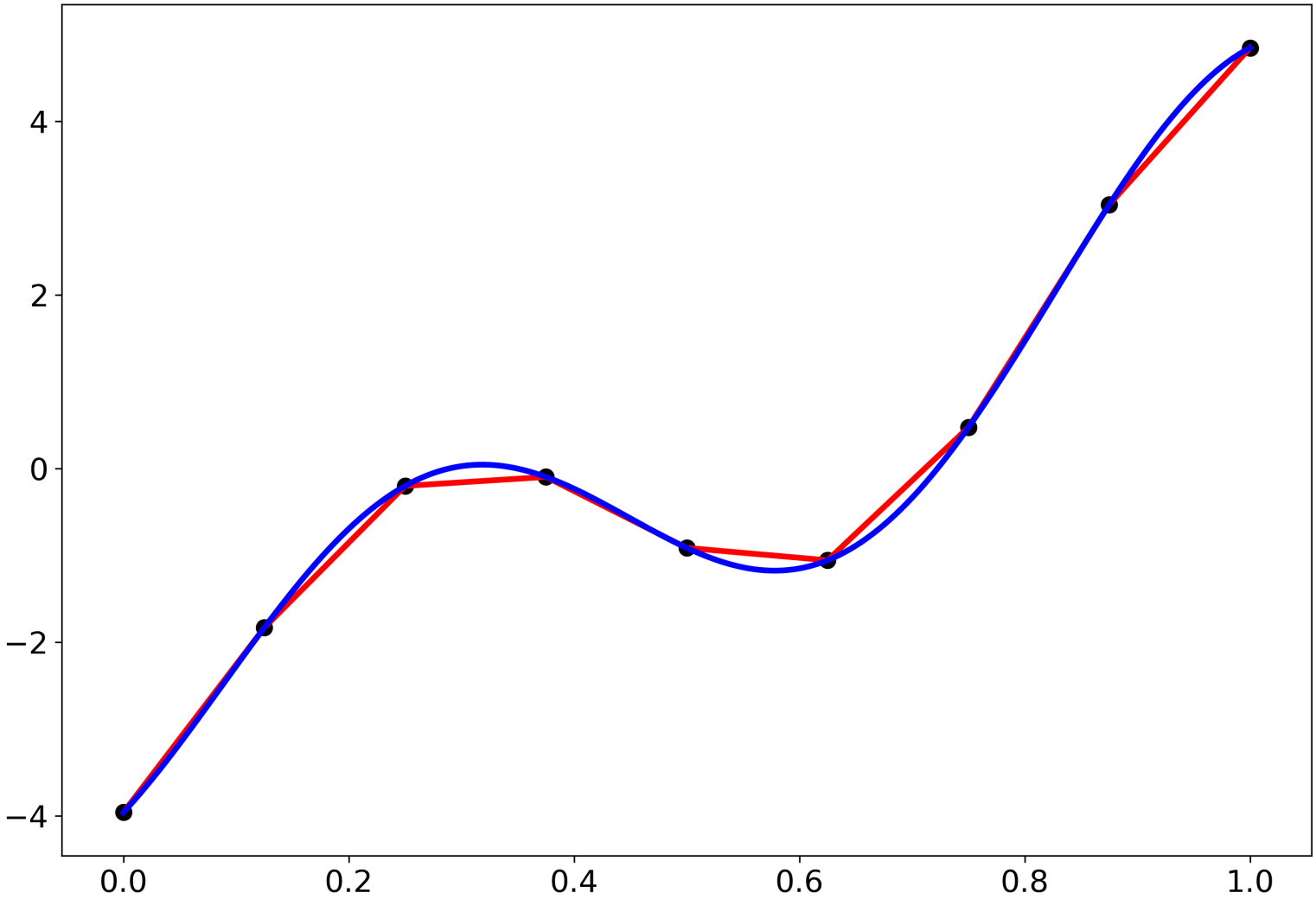
- ¿Cómo hacemos para resolver una ecuación diferencial con una computadora ?

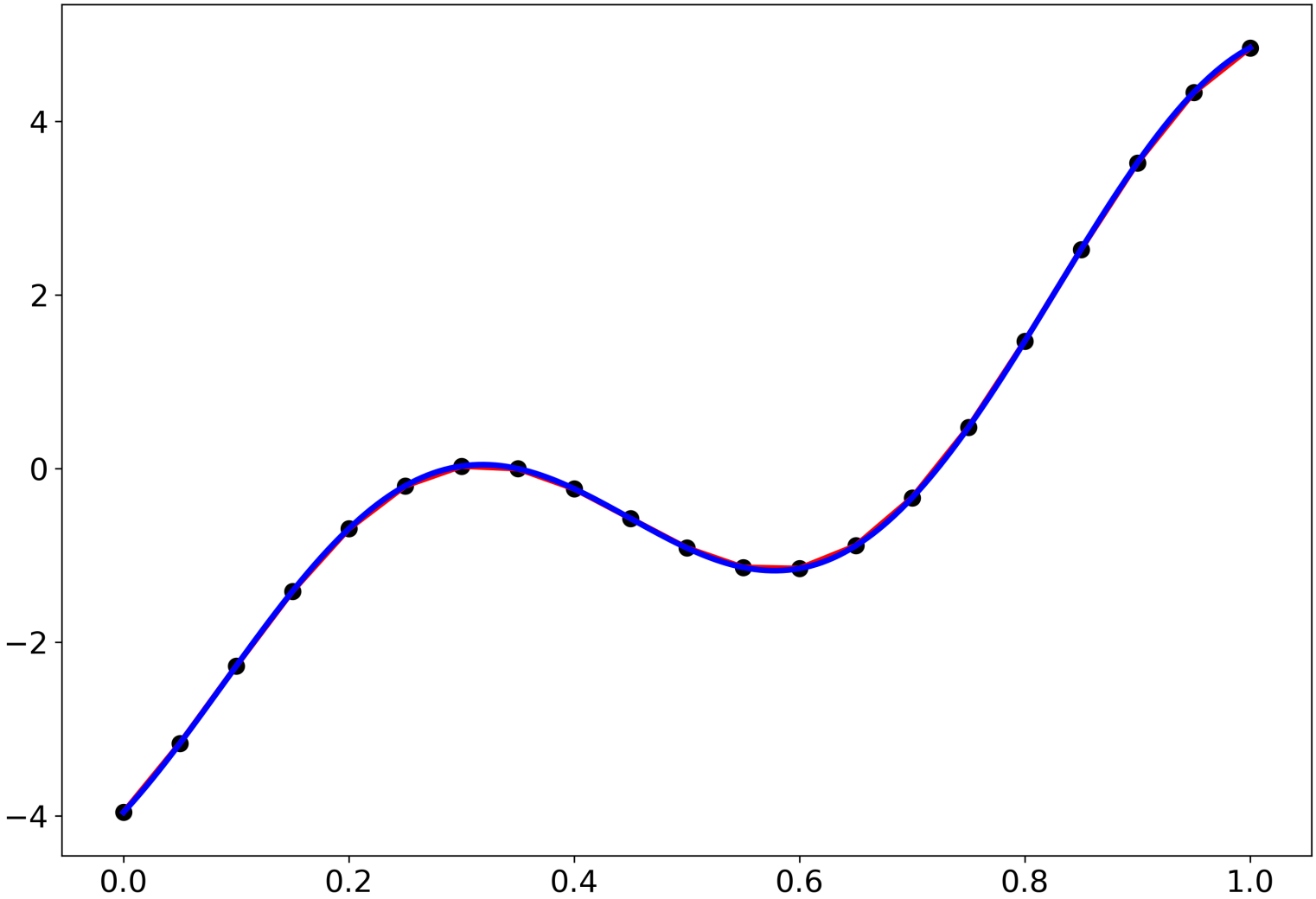
- ¿Cómo hacemos para resolver una ecuación diferencial con una computadora ?
- La idea es que podemos aproximar la derivada de una función teniendo algunos valores. Obviamente, cuantos más mejor.

- ¿Cómo hacemos para resolver una ecuación diferencial con una computadora ?
- La idea es que podemos aproximar la derivada de una función teniendo algunos valores. Obviamente, cuantos más mejor.
- Por ejemplo, $u'(x) \approx \frac{u(x+h)-u(x)}{h}$ si h es suficientemente chico.


4
2
0
−2
−4
0.0
0.2
0.4
0.6
0.8
1.0


4
2
0
−2
−4
0.0
0.2
0.4
0.6
0.8
1.0


4
2
0
−2
−4
0.0
0.2
0.4
0.6
0.8
1.0


4
2
0
−2
−4
0.0
0.2
0.4
0.6
0.8
1.0

- De la misma forma, podemos usar

$$u''(x) \approx \frac{u(x+h) + u(x-h) - 2u(x)}{h^2}.$$

- De la misma forma, podemos usar

$$u''(x) \approx \frac{u(x+h) + u(x-h) - 2u(x)}{h^2}.$$

- Entonces podemos transformar el problema en

$$M \cdot u = \lambda \cdot u,$$

que es algo que nuestro algoritmo puede resolver.

- Por ejemplo, la ecuación $-u''(x) = \lambda u(x)$ tiene solución si y solo si $\lambda = (i \cdot \pi)^2$ para $i \in \mathbb{N}$. Si resolvemos este caso con nuestro algoritmo, obtenemos los siguientes valores para λ :

- Por ejemplo, la ecuación $-u''(x) = \lambda u(x)$ tiene solución si y solo si $\lambda = (i \cdot \pi)^2$ para $i \in \mathbb{N}$. Si resolvemos este caso con nuestro algoritmo, obtenemos los siguientes valores para λ :

Componente (i)	λ_i	$(i \cdot \pi)^2$	$\frac{\lambda_i}{(i \cdot \pi)^2}$
1	9.483	9.869	0.96
2	37.897	39.478	0.956
3	85.134	88.826	0.958
4	151.015	157.913	0.956
5	235.289	246.740	0.953
6	337.638	355.305	0.95
7	457.673	483.610	0.946
8	594.939	631.654	0.941

¡Muchas gracias !

¿Preguntas?