

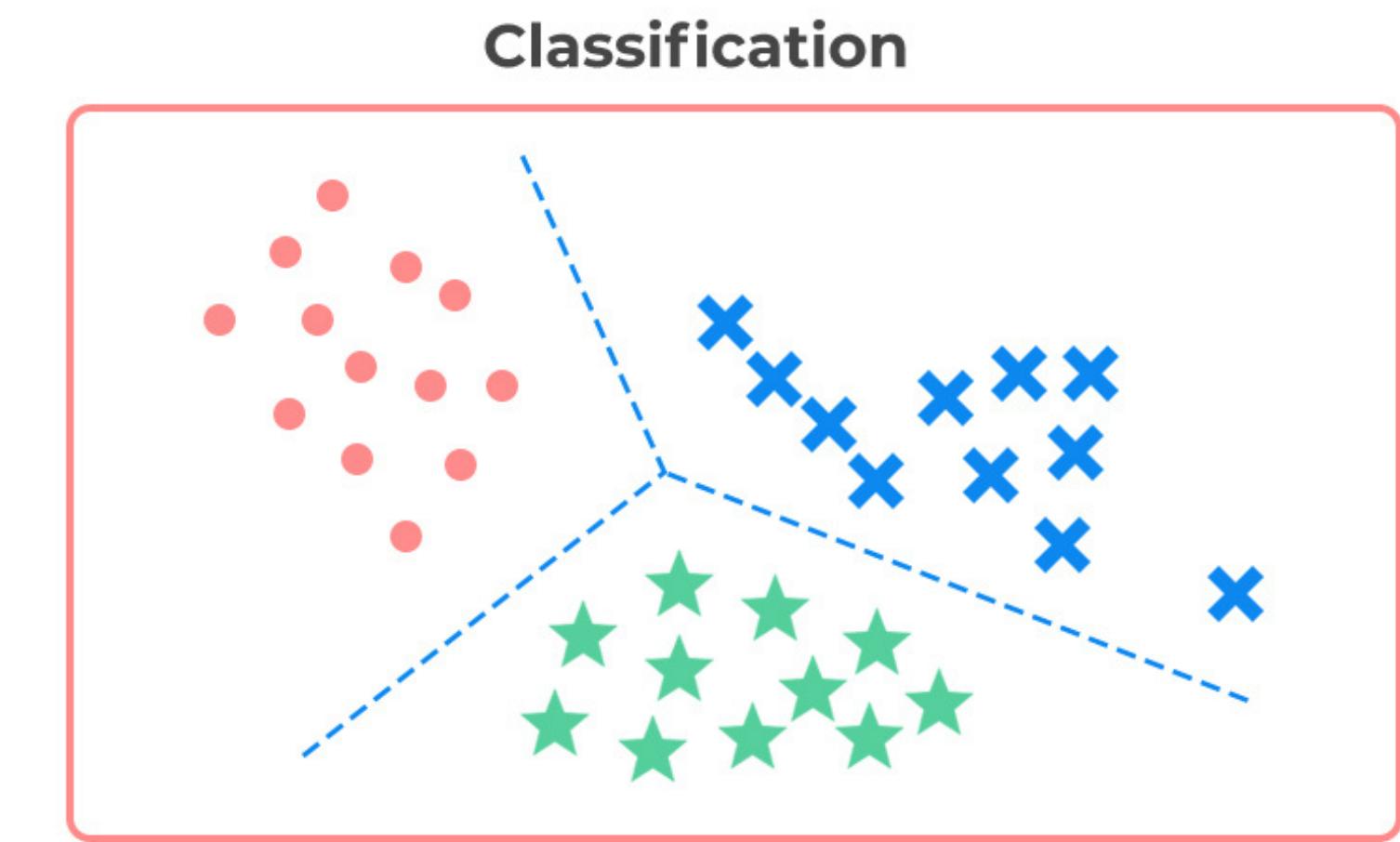
# TP4: APRENDIZAJE NO SUPERVISADO

Grupo 3:  
Agustín Spitzner  
Ana Cruz  
Camila Borinsky

# INTRODUCCIÓN

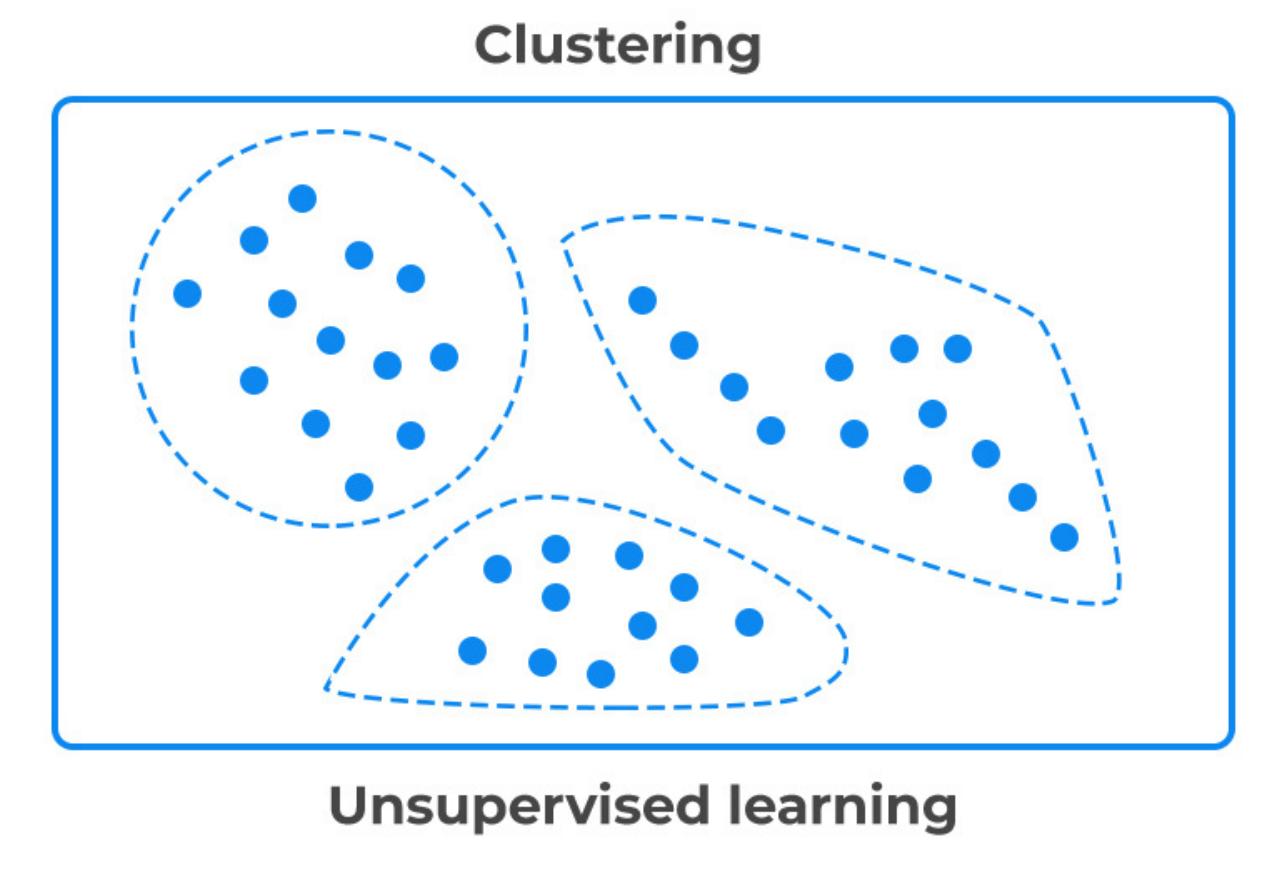
# APRENDIZAJE SUPERVISADO

Se le presenta al sistema ejemplos de input y su salida deseada. El objetivo es que pueda aprender una regla general que dado un input produzca el output correspondiente.



# APRENDIZAJE NO SUPERVISADO

Utilizado cuando no se conoce la salida deseada de los ejemplos de input. El objetivo es que el sistema pueda encontrar patrones y asociaciones entre los ejemplos para agruparlos o ordenarlos.



# DATASET

Conjunto de datos *europe.csv*

Características sociales, económicas y geográficas de 28 países de europa:

- Country: Nombre del país
- Area: Área
- GDP: Producto bruto interno
- Inflation: Inflación anual
- Life.expect: Expectativa de vida en años
- Military: Ejército
- Pop.growth: Tasa de crecimiento poblacional
- Unemployment: Tasa de desempleo

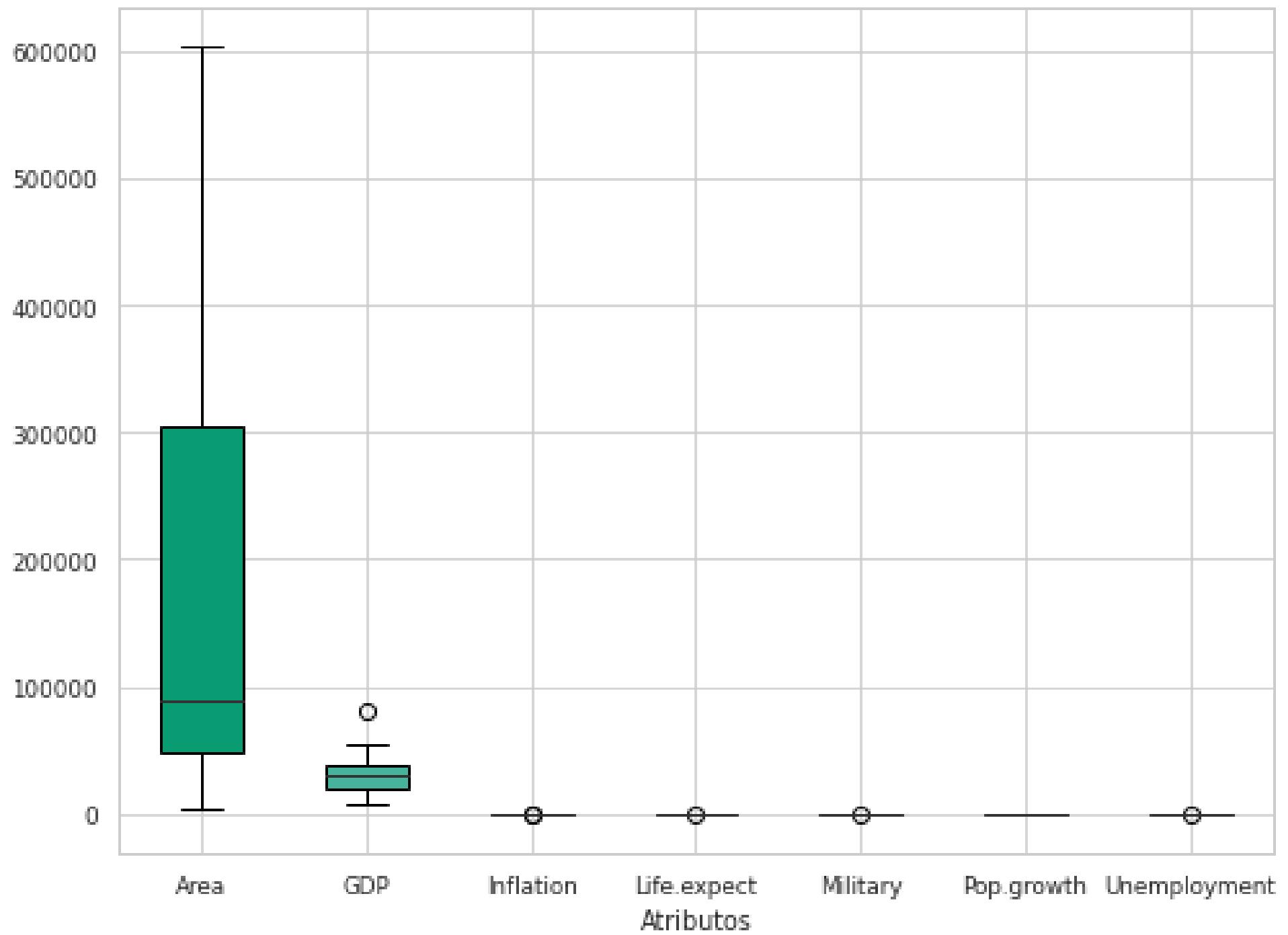
Pre-procesamiento:  
estandarización de variables

$$\tilde{X}_i = \frac{X_i - \bar{X}_i}{s_i}$$

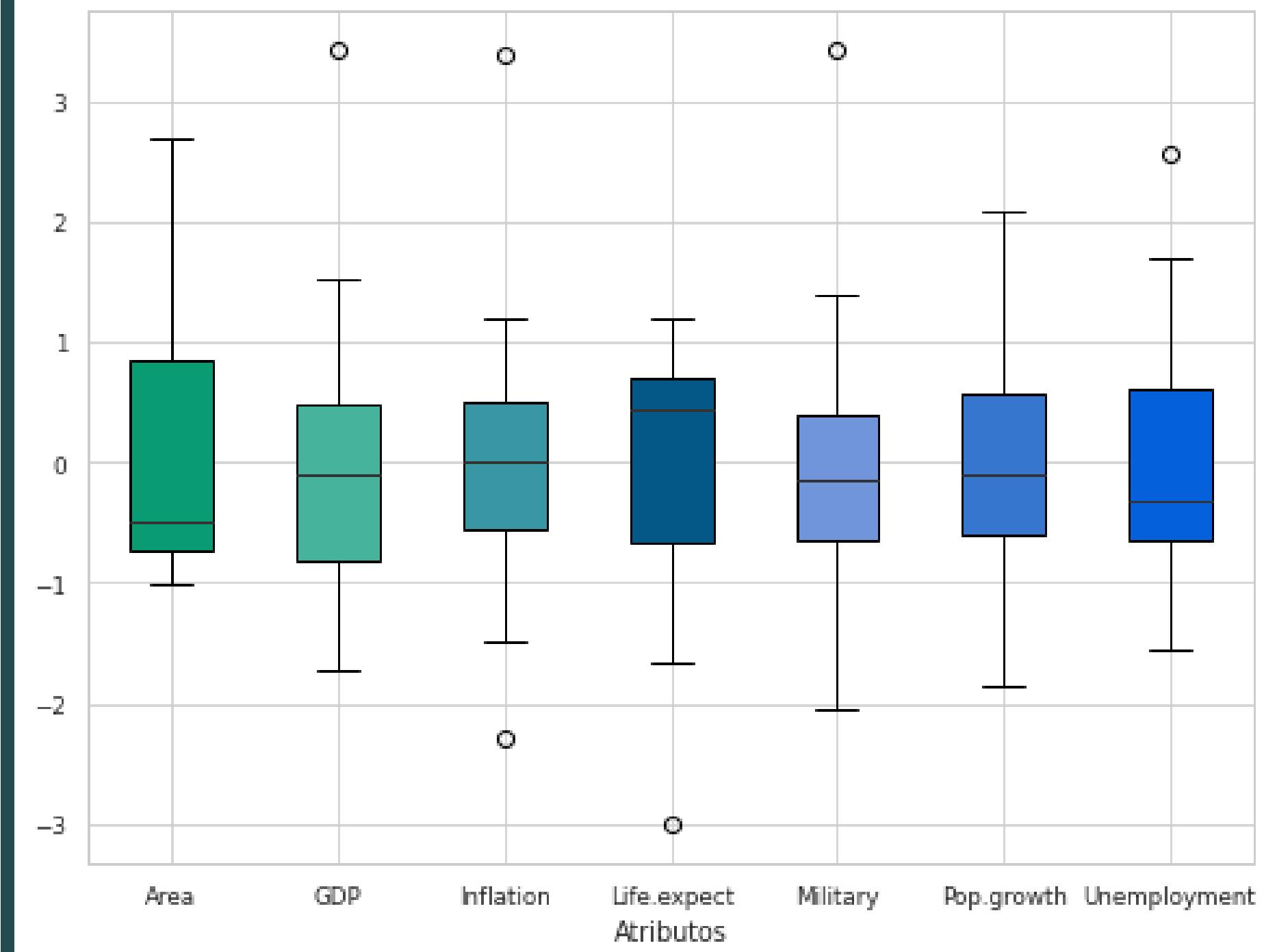
$\bar{X}_i$  : media de variable  $X_i$

$s_i$  : desvío estándar de  $X_i$

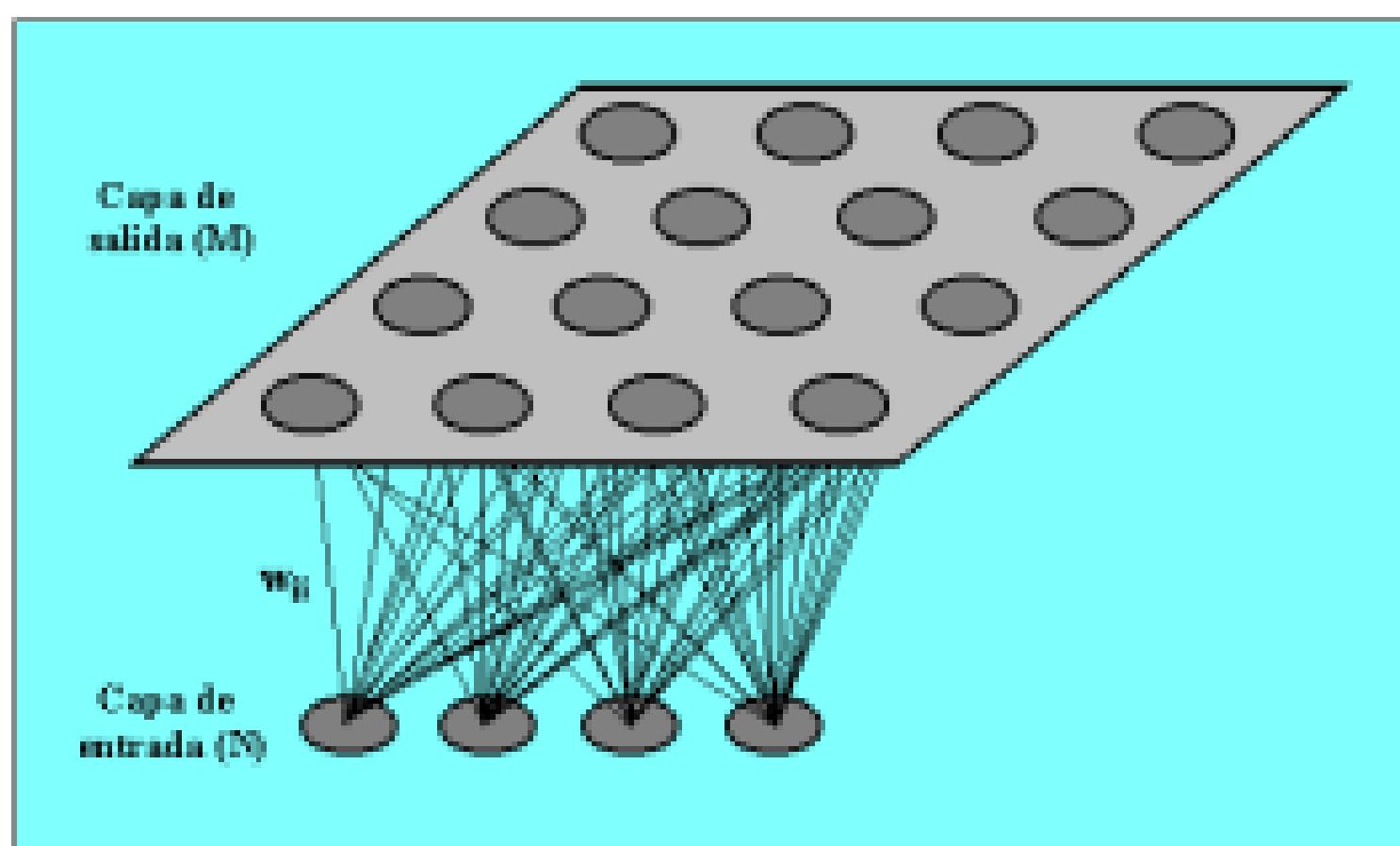
Datos no estandarizados



Datos estandarizados



# KOHONEN



$$d(x, w) = \|x(t) - w_c(t)\|$$

- Reducir la dimensión de un conjunto de datos agrupandolos en una grilla bidimensional.
- Red con 1 capa de entrada y 1 capa de salida. Cada nodo de entrada está conectado con todos los nodos de salida.
- Utiliza aprendizaje competitivo para encontrar la neurona con pesos más similares al valor de input.

# PARÁMETROS

## FIJOS

- Neuronas en capa de entrada: 7  
(dimensión de cada registro)
- Tasa de aprendizaje inicial: 1
- Radio inicial: mitad del lado de la grilla  
de salida

# PARÁMETROS VARIABLES

- Dimensiones capa de salida: 2x2, 3x3, 4x4, 5x5, 6x6
- Actualización de radio:
  - lineal
  - exponencial
  - constante
- Actualización de radio:
  - lineal
  - exponencial
  - constante
- Inicialización de pesos: aleatorio, del conjunto de entrada

$$r(t) = \frac{1 - r_i}{n \times e} t + r_i$$

$$r(t) = (r_i - 1) \times e^{\frac{-5t}{n \times e}} + 1$$

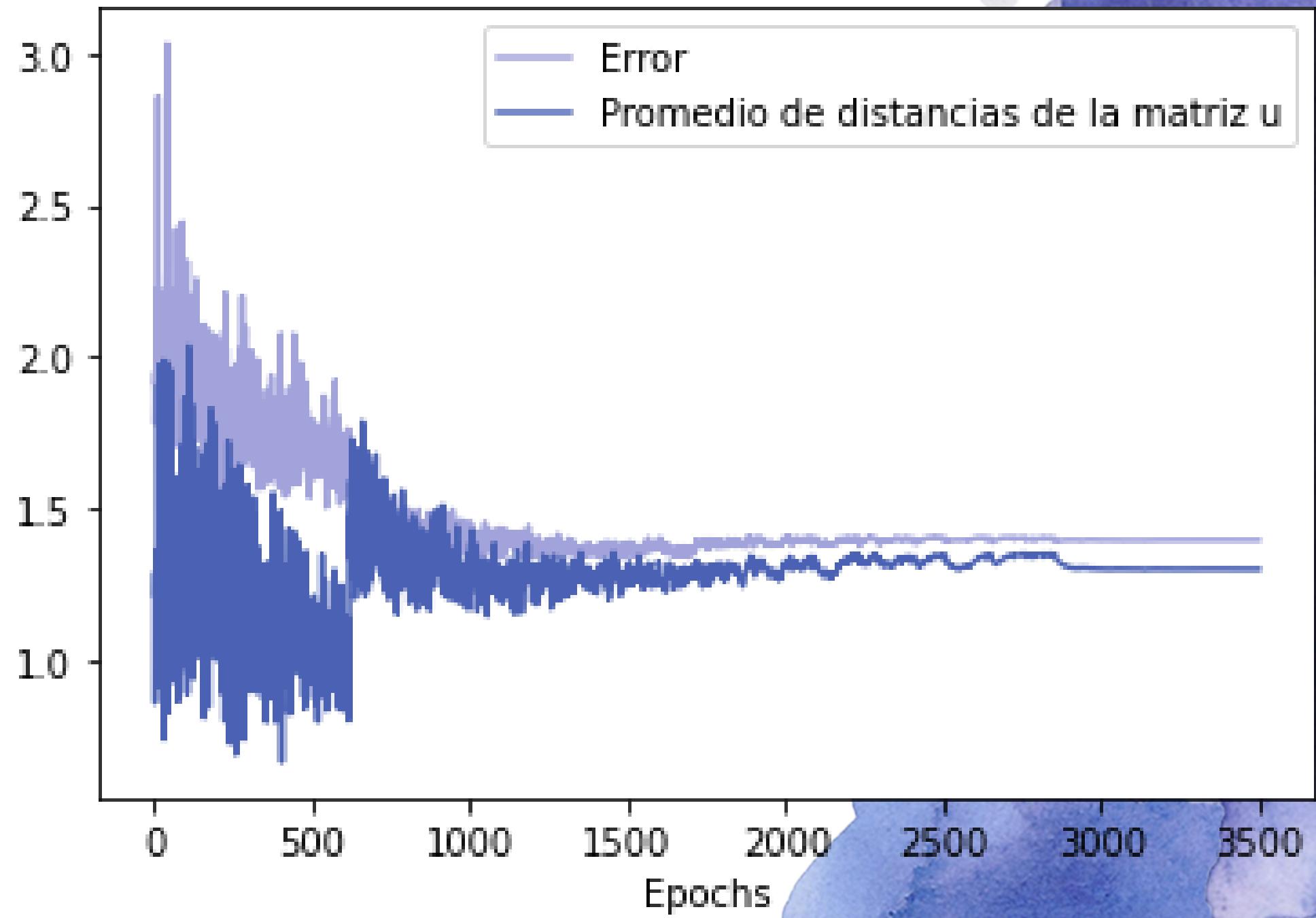
$$\eta(t) = -\frac{\eta_i}{n \times e} t + \eta_i$$

$$\eta(t) = \eta_i \times e^{\frac{-5t}{n \times e}}$$

# RESULTADOS

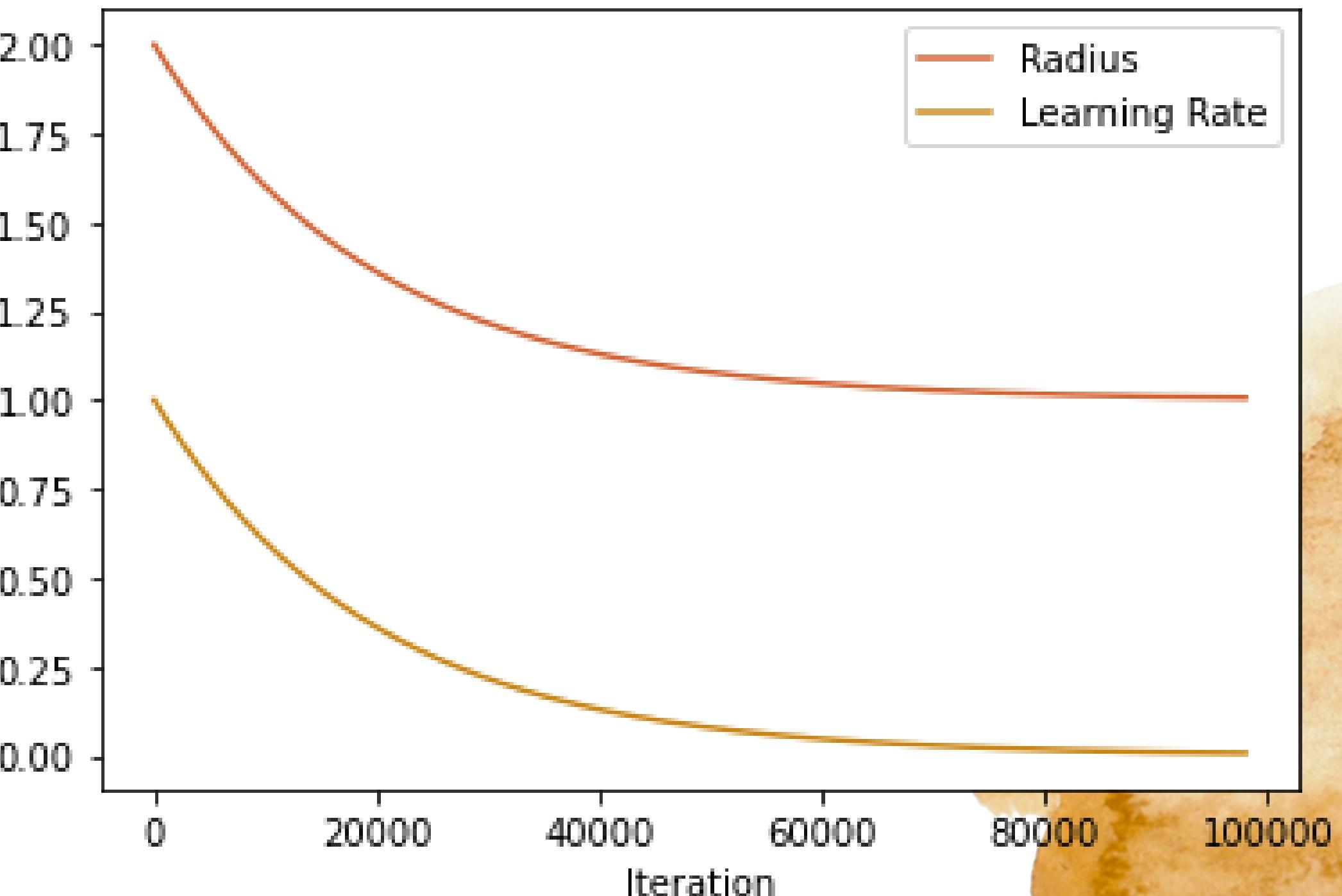
- Dimensión salida: 4x4
- Límite de épocas: 3500
- Radio inicial: 2
- Actualización radio: exponencial
- Tasa de aprendizaje inicial: 1
- Actualización tasa de aprendizaje: exponencial
- Inicialización de pesos: valores de entrada

Asociar países que posean las mismas características geopolíticas, económicas y sociales



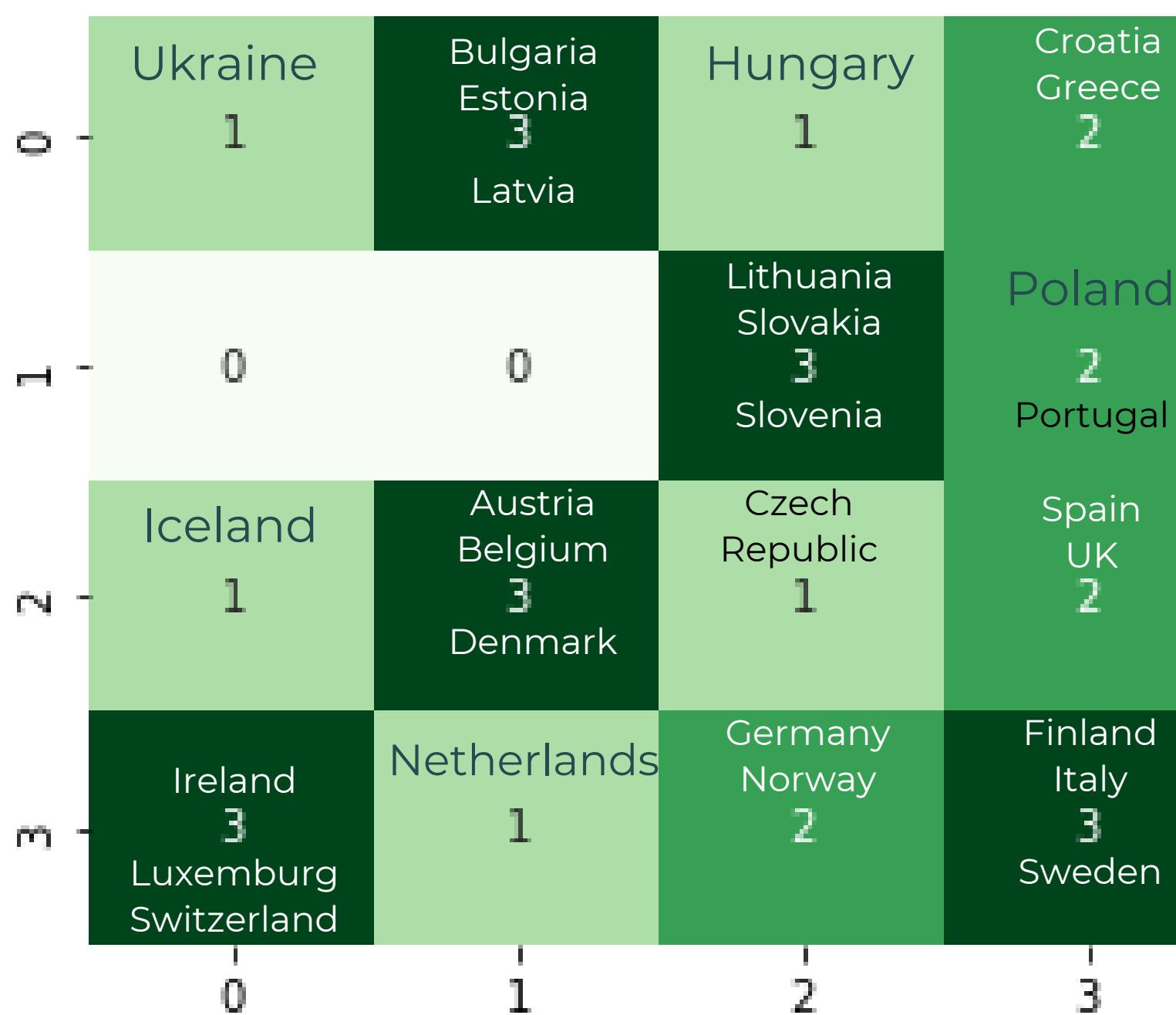
# ACTUALIZACIÓN DE RADIO Y TASA DE APRENDIZAJE

- Dimensión salida: 4x4
- Límite de épocas: 3500
- Radio inicial: 2
- Actualización radio: exponencial
- Tasa de aprendizaje inicial: 1
- Actualización tasa de aprendizaje: exponencial
- Inicialización de pesos: valores de entrada



# MAPA DE CALOR

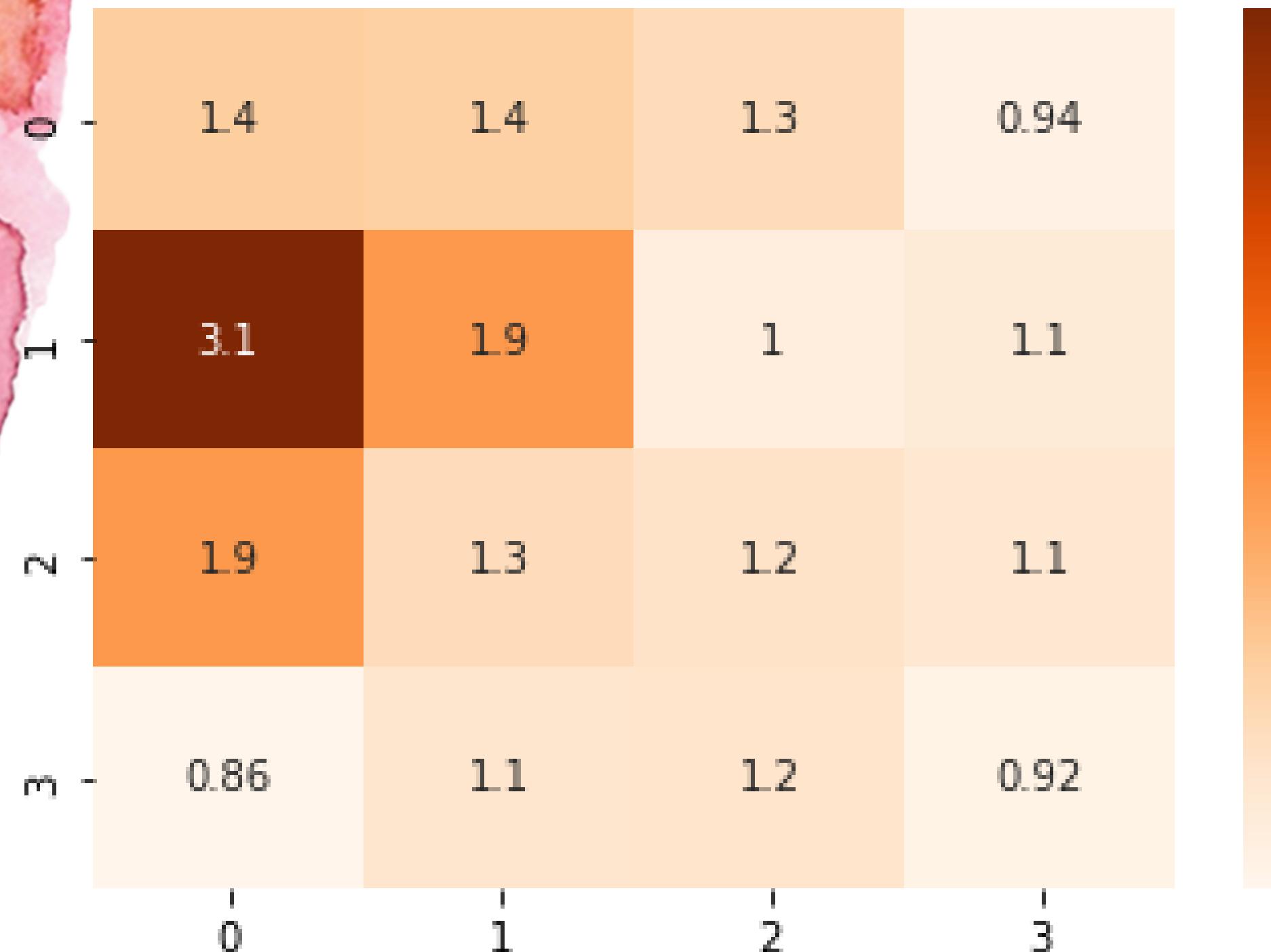
Cantidad de elementos asociados a cada neurona



- Dimensión salida: 4x4
- Límite de épocas: 3500
- Radio inicial: 2
- Actualización radio: exponencial
- Tasa de aprendizaje inicial: 1
- Actualización tasa de aprendizaje: exponencial
- Inicialización de pesos: valores de entrada

# MATRIZ-U

Distancias promedio entre  
neuronas vecinas



- Dimensión salida: 4x4
- Límite de épocas: 3500
- Radio inicial: 2
- Actualización radio: exponencial
- Tasa de aprendizaje inicial: 1
- Actualización tasa de aprendizaje: exponencial
- Inicialización de pesos: valores de entrada

# CONCLUSIONES KOHONEN

- Es necesario estandarizar los datos de entrada
- Difícil elegir la cantidad de neuronas de salida
- Útiles cuando se tiene un dataset de muchas dimensiones (aunque de mayor costo computacional)
- El Quantization Error que usamos para evaluar sirve pero decrece para mayor dimensión de salida y mayor cantidad de épocas.
- No encontramos mayores diferencias en inicializar los pesos aleatoriamente.
- Quedó fuera de scope probar con topología hexagonal y probar otras métricas de evaluación (error topológico, convergencia, confiabilidad)

# OJA

- Las componentes principales son utilizadas para extraer características destacadas o importantes de un conjunto de datos bajando la dimensión del mismo
- La regla de oja permite calcular la primera componente principal iterativamente (ventajas computacionales)
- Oja propone una modificación al aprendizaje Hebbiano asegurando la convergencia.



# PRIMERA COMPONENTE PRINCIPAL

	Oja	Librería PCA
Area	0.12455953	0.124874
GDP	-0.49946862	-0.500506
Inflation	0.40921789	0.406518
Life.expect	-0.48552654	-0.482873
Military	0.18307143	0.188112
Pop.growth	-0.47705044	-0.475704
Unemployment	0.26646134	0.271656

Tasa de aprendizaje: 0.0001  
Iteraciones: 2000  
Ejecuciones: 10

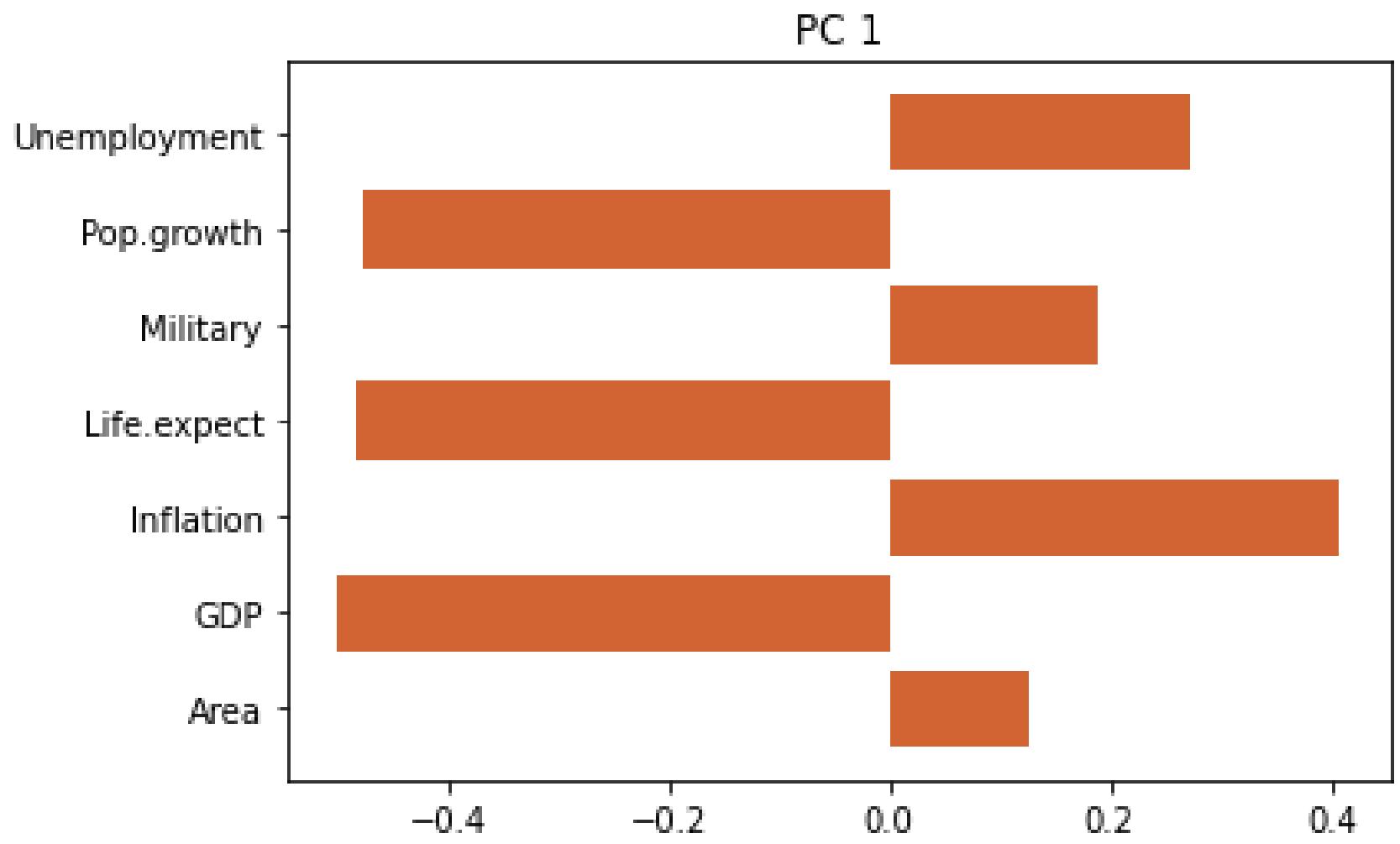


# ANÁLISIS DE PRIMERA COMPONENTE PRINCIPAL

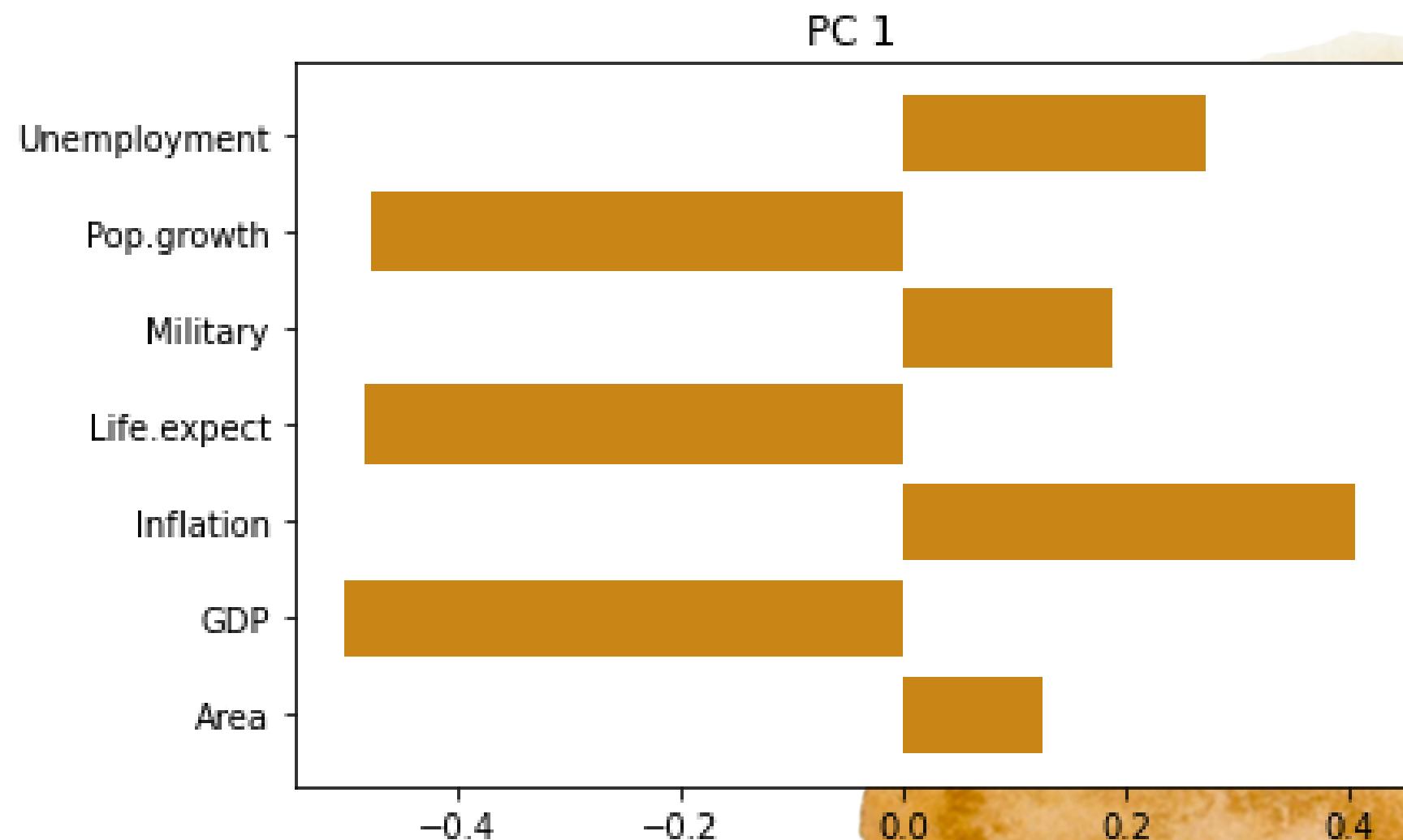
Tasa de aprendizaje: 0.0001  
Iteraciones: 2000  
Ejecuciones: 10

# LOADINGS DE PC1

OJA

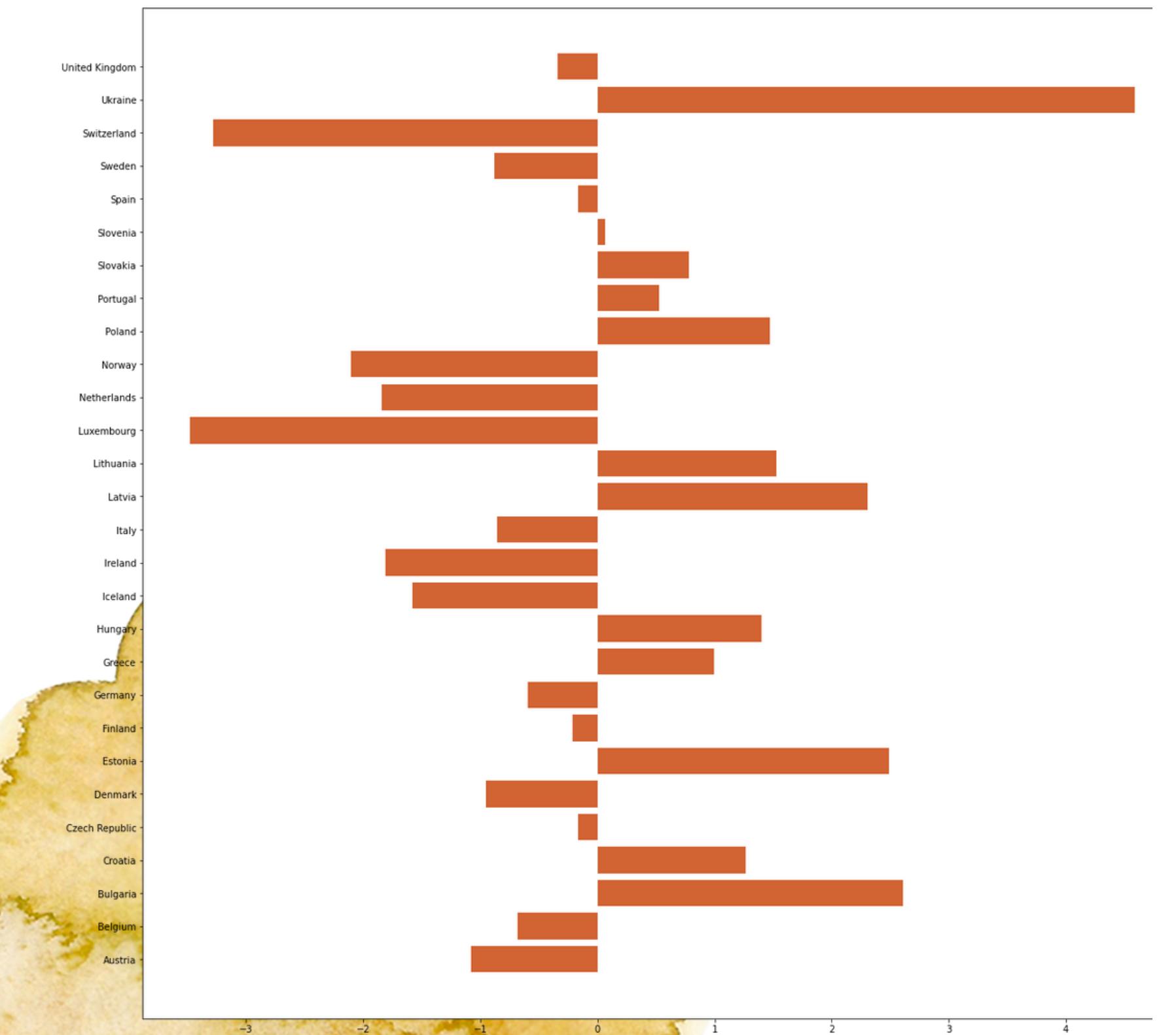


LIBRERÍA PCA

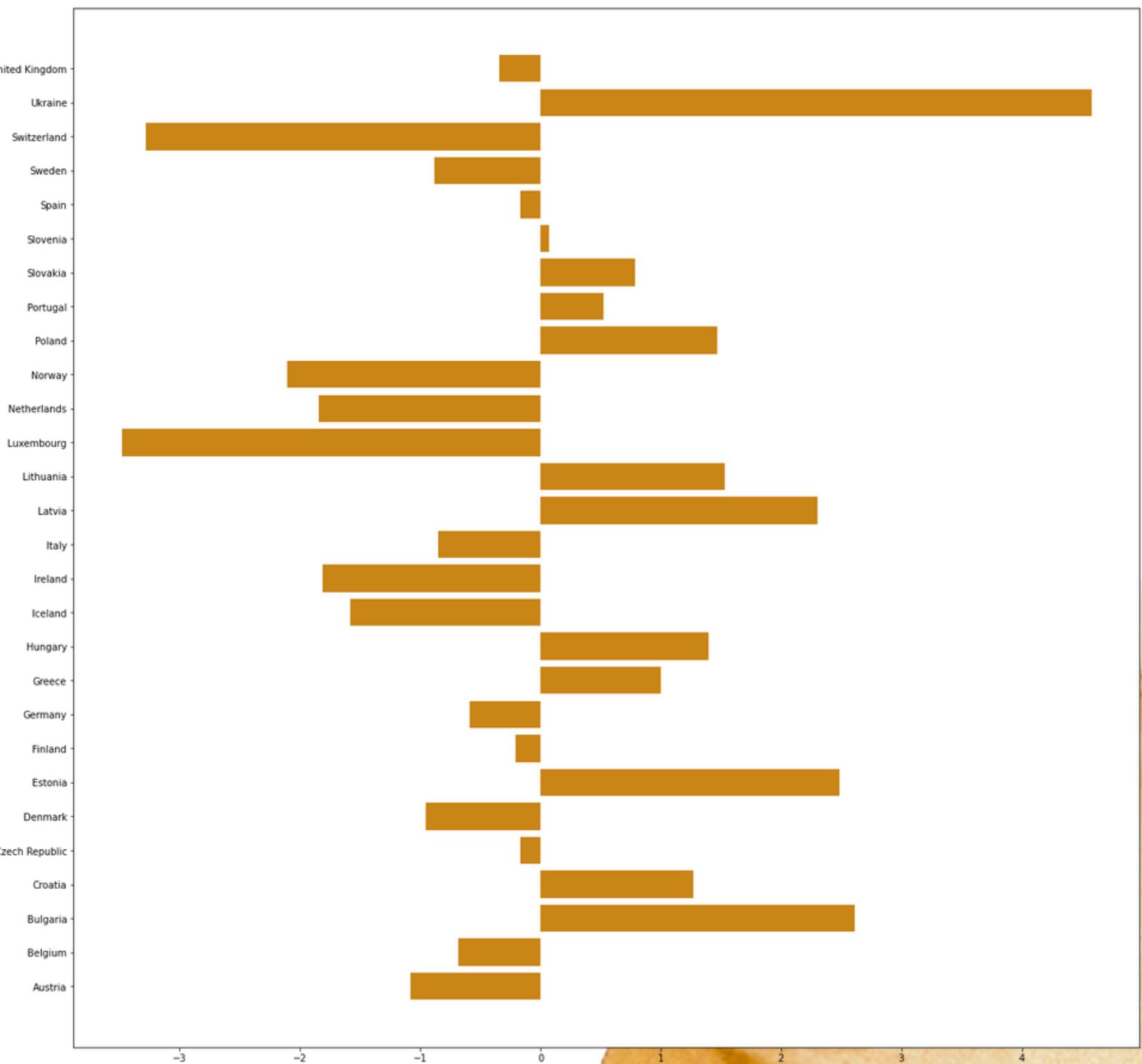


Tasa de aprendizaje: 0.0001  
Iteraciones: 2000  
Ejecuciones: 10

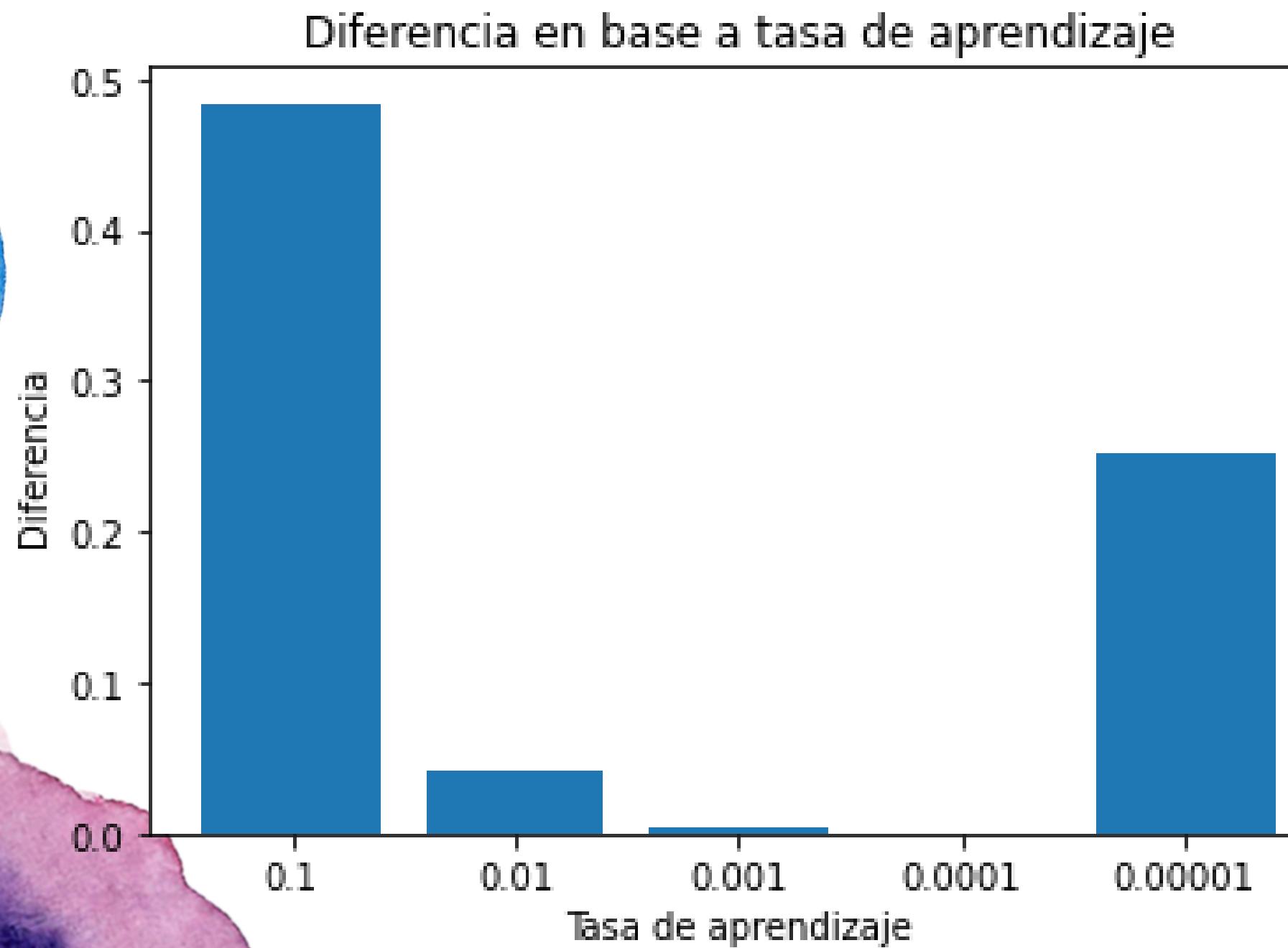
# OJA



# LIBRERÍA PCA



# DIFERENCIA CON LIBRERÍA EN BASE A TASA DE APRENDIZAJE



Iteraciones: 1000

Ejecuciones: 10

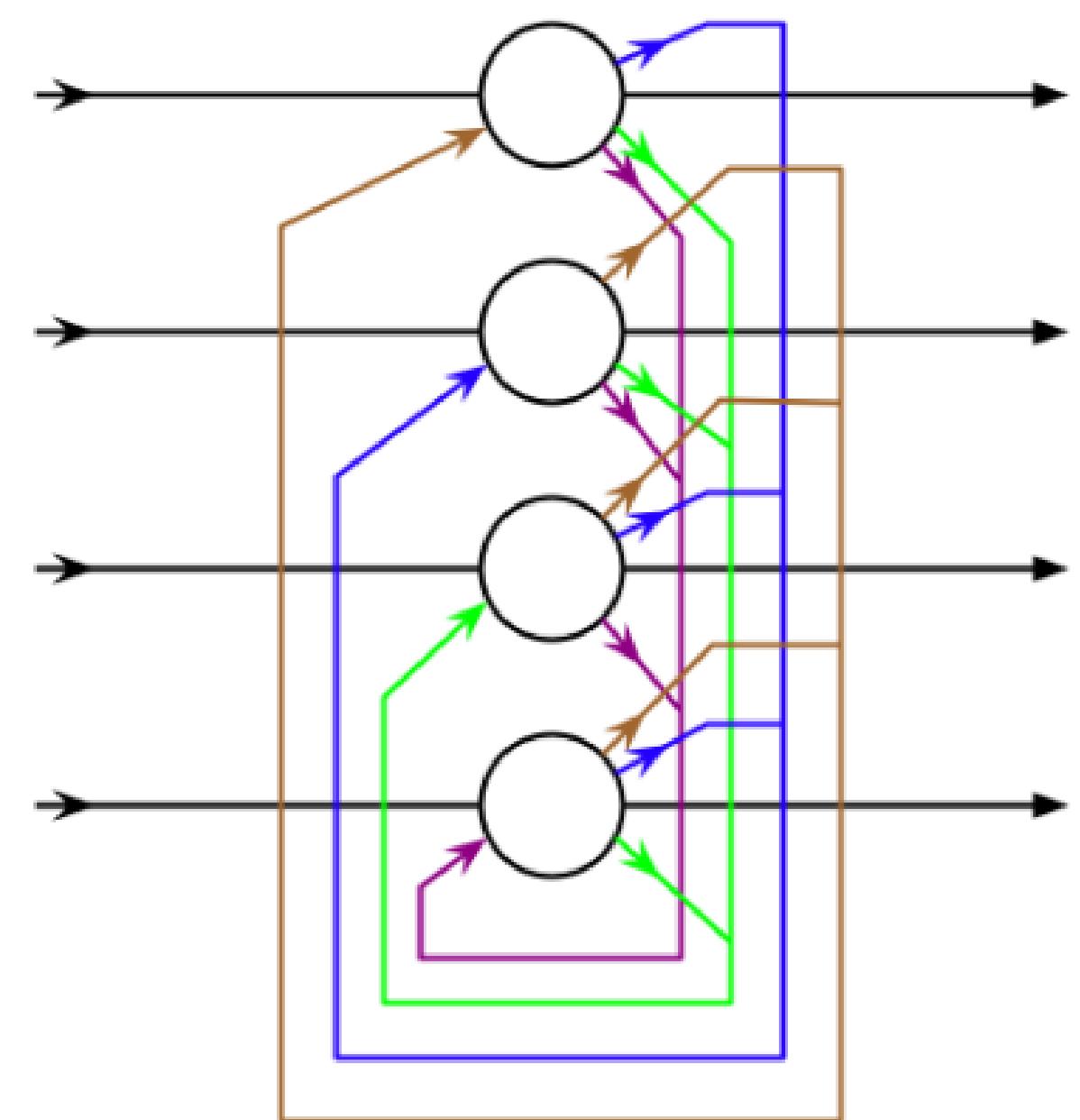
$$Diferencia = \sum_{i=1}^7 \frac{|library_i - O_i|}{7}$$

# CONCLUSIONES OJA

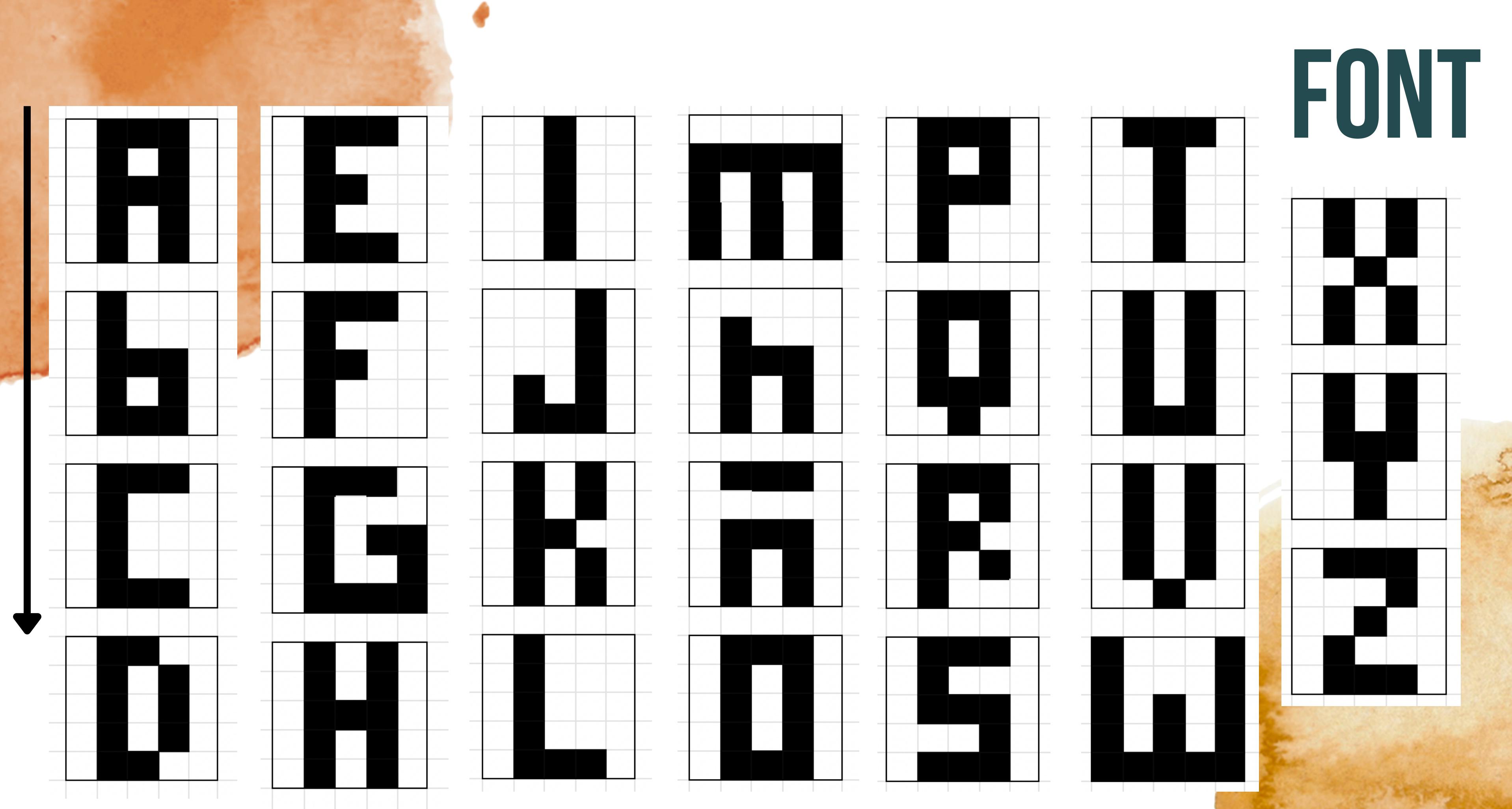
- Tanto la librería como el método de Oja convergen hacia el mismo autovector (o al opuesto)
- Disminuir la tasa de aprendizaje no siempre mejora la aproximación hacia la primera componente
- Presenta ventajas computacionales
- Para datasets con dimensionalidad muy alta puede ser más útil Kohonen para agrupar registros y bajar la dimensionalidad.

# HOPFIELD

- Memoria asociativa: en base a patrones almacenados, dado un nuevo patrón responde el que más se parece.
- Redes recurrentes: Todas las neuronas se conectan entre sí, pero no consigo mismas
- Sirve para asignar un patrón de consulta binario con alguno almacenado.



# FONT



# PATRONES

Patrón más ortogonal: {G,I,M,X} (Avg\_dot: 5.0)

Patrón menos ortogonal: {A,H,K,X} (Avg\_dot: 21.67)

**UTILIZAMOS {G,I,M,X}**

# EFECTO DE AUMENTO EN PROBABILIDAD DE RUIDO

Patterns: ('G', 'I', 'M', 'X'), Avg\_dot: 5.0  
Letter: X, noise probability: 0%

```
*  *
*  *
*
*  *
*  *
```

Found pattern in 1 iterations

```
*  *
*  *
*
*  *
*  *
```

Patterns: ('G', 'I', 'M', 'X'), Avg\_dot: 5.0  
Letter: X, noise probability: 1%

```
*  *
*  *
*
*  *
*  *
```

Found pattern in 1 iterations

```
*  *
*  *
*
*  *
*  *
```

Patterns: ('G', 'I', 'M', 'X'), Avg\_dot: 5.0  
Letter: X, noise probability: 5%

```
*  *
*  *
*
*  *
*  *
```

Found pattern in 1 iterations

```
*  *
*  *
*
*  *
*  *
```

# EFECTO DE AUMENTO EN PROBABILIDAD DE RUIDO

Patterns: ('G', 'I', 'M', 'X'), Avg\_dot: 5.0  
Letter: X, noise probability: 10%

```
*  *
*  *
*
*
*  *
```

Found pattern in 2 iterations

```
*  *
*  *
*
*
*  *
```

Patterns: ('G', 'I', 'M', 'X'), Avg\_dot: 5.0  
Letter: X, noise probability: 20%

```
*  *
*  *  *
*  *
*  *  *  *
*  *
```

Found pattern in 2 iterations

```
*  *
*  *
*
*
*  *
```

# EFECTO DE AUMENTO EN PROBABILIDAD DE RUIDO

Patterns: ('G', 'I', 'M', 'X'), Avg\_dot: 5.0  
Letter: X, noise probability: 30%

```
* * * *
* * * *
*
* * * *
```

Found pattern in 2 iterations

```
* *
* *
*
* *
*
```

Patterns: ('G', 'I', 'M', 'X'), Avg\_dot: 5.0  
Letter: X, noise probability: 40%

```
* * *
*
*
*
*
* *
```

Found pattern in 3 iterations

```
*
```

# ESTADO ESPÚREO

Patterns: ('G', 'I', 'M', 'X'), Avg\_dot: 5.0

Letter: X, noise probability: 30%

```
*      *
      * * *
*
*   *
```

Found pattern in 4 iterations

```
*      *
* * *
*
* * *
* *
```

Encuentra un patrón  
pero este es inválido

# CASO DE ERROR

```
Patterns: ('G', 'I', 'M', 'X'), Avg_dot: 5.0
Letter: X, noise probability: 10%
* *
* * *
* * * *
* *

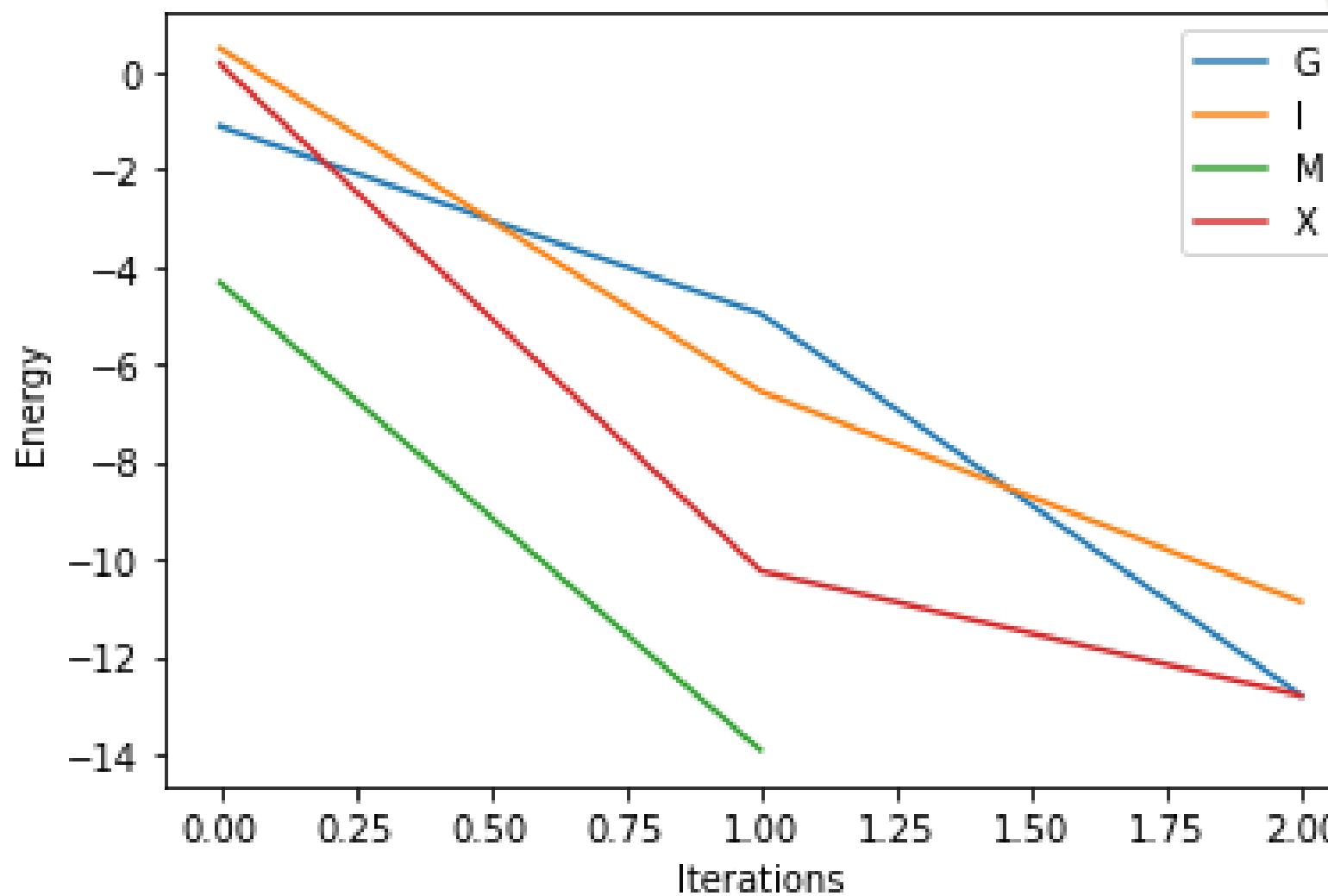
Found pattern in 4 iterations
* *
* *
* *
* * *
* * *
* * *
* *
* *
* *
* *
* *
* *
```

```
Patterns: ('G', 'I', 'M', 'X'), Avg_dot: 5.0
Letter: X, noise probability: 25%
* *
* * *
* * * *
* *
* * * *
* *

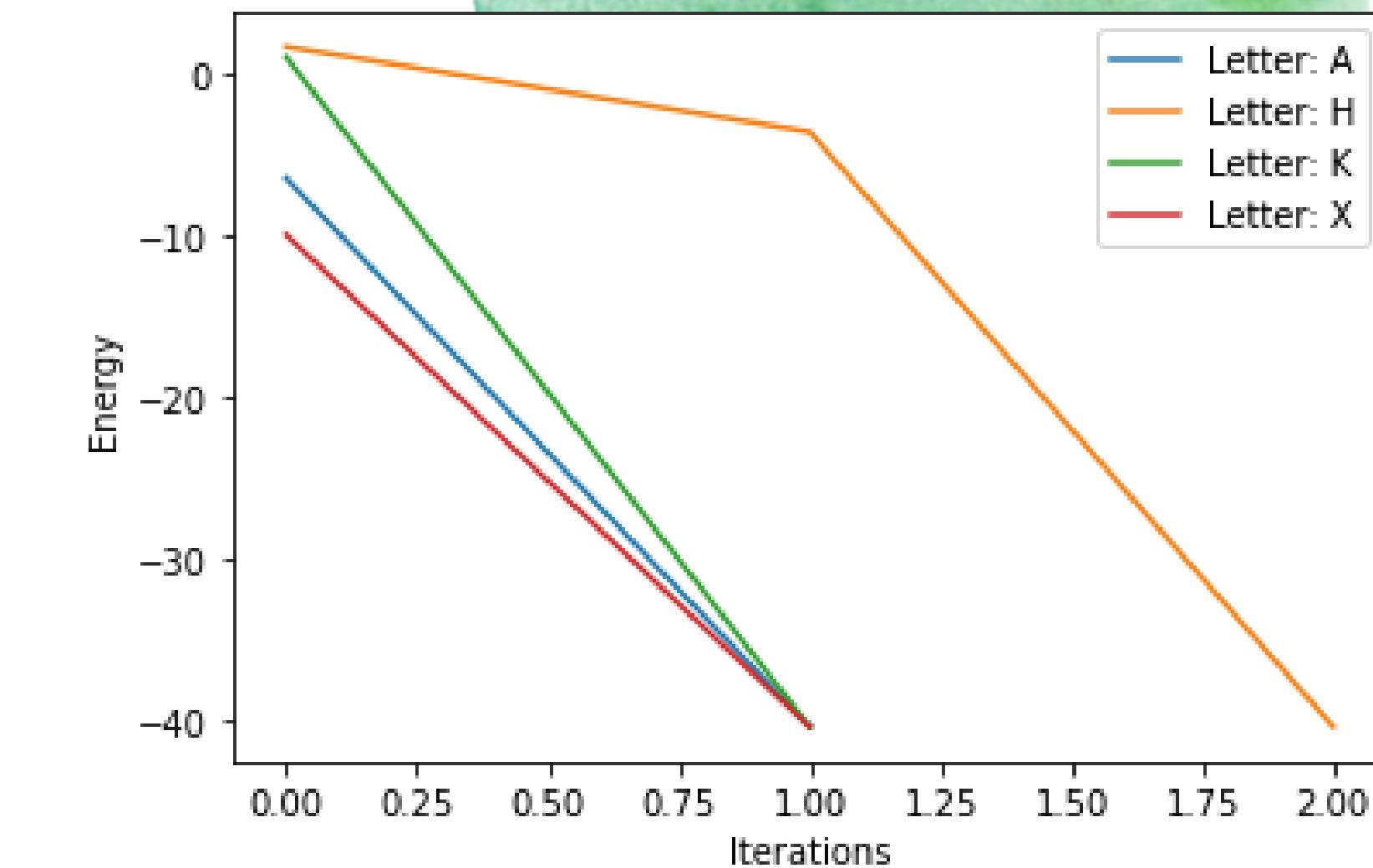
Found pattern in 4 iterations
* *
* *
* *
* * *
* * *
* * *
* *
* *
* *
* *
* *
* *
```

# CASO DE ACIERTO

# ENERGÍA



Letras muy ortogonales  
porcentaje de ruido 30%

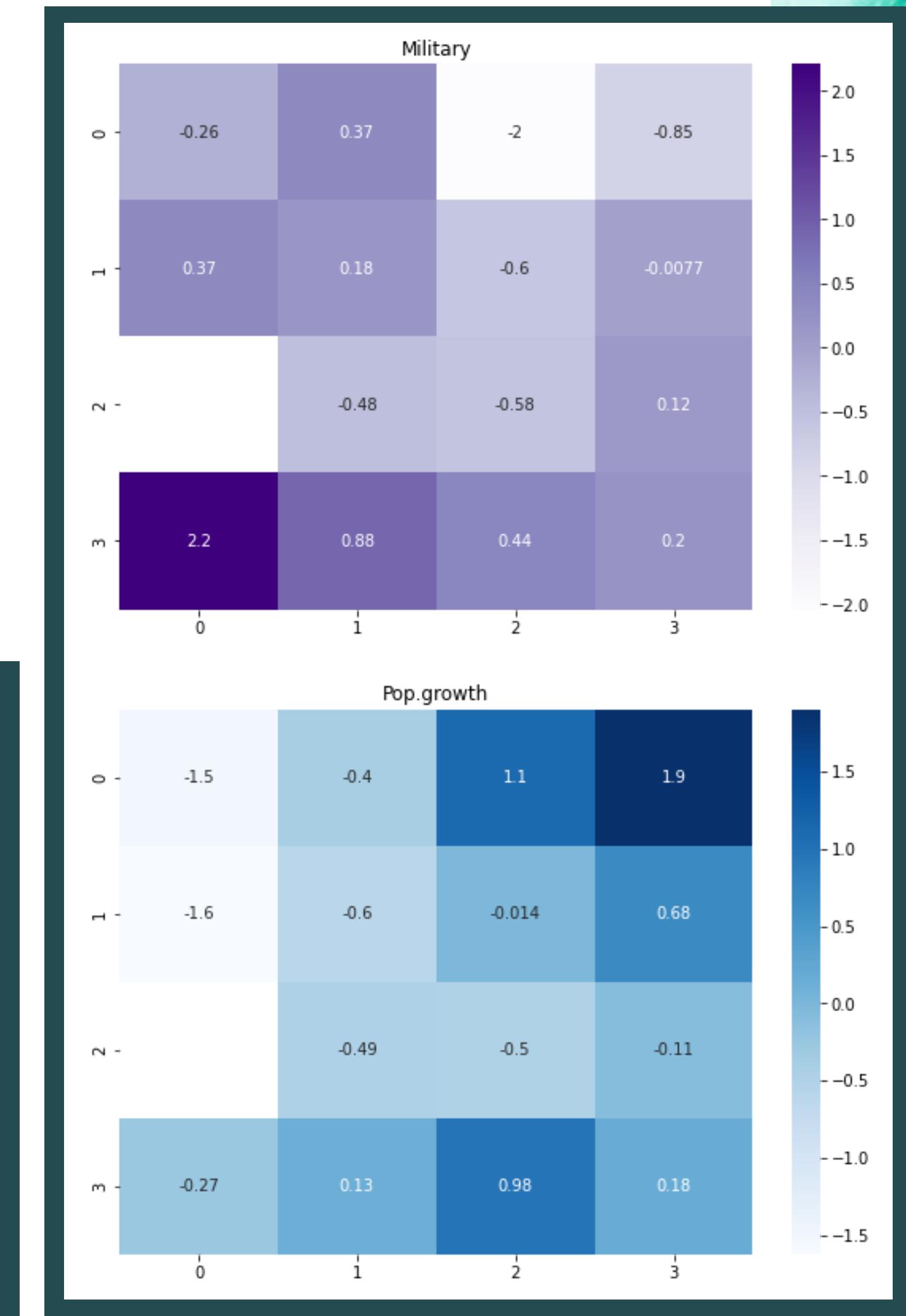
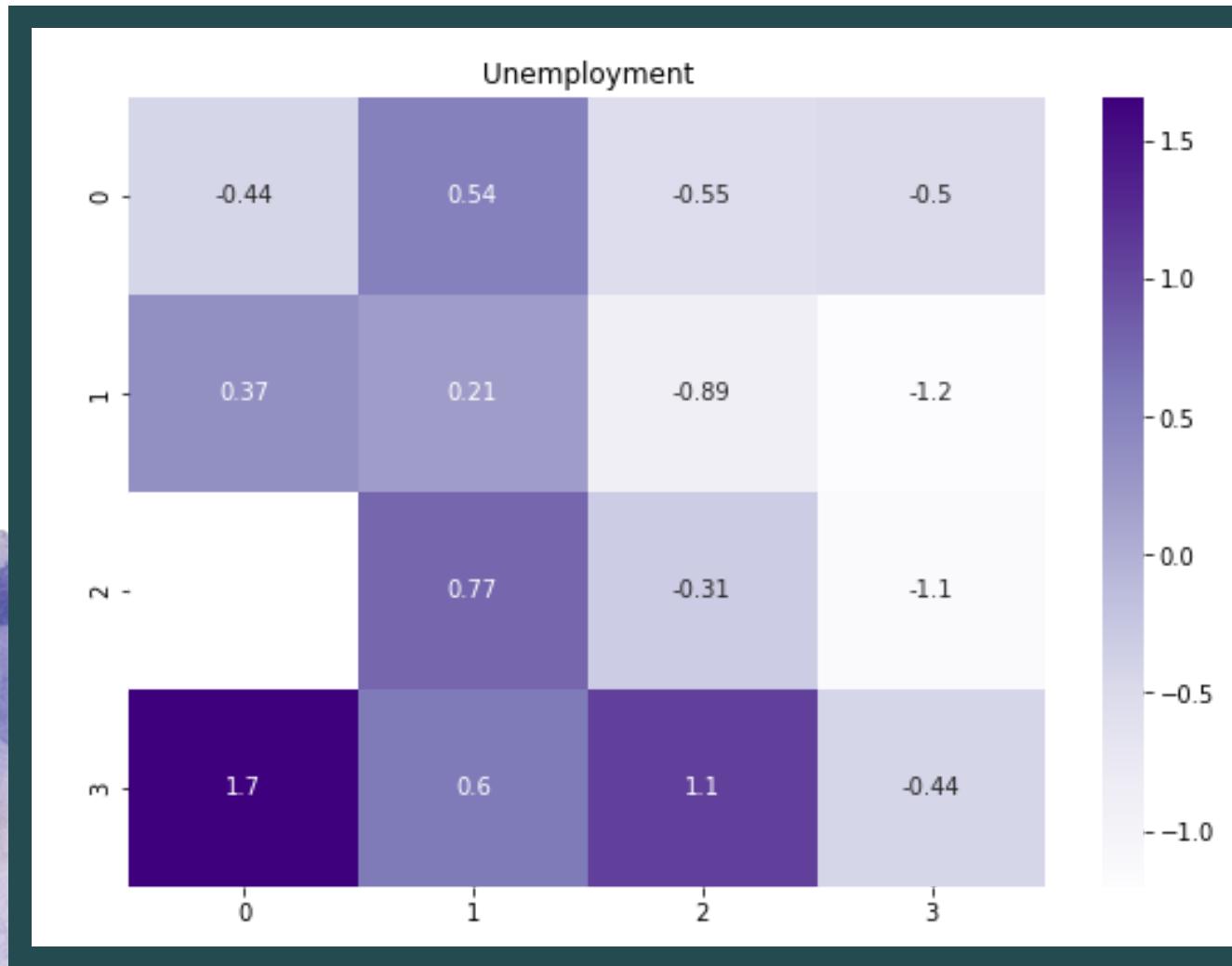


Letras menos ortogonales  
porcentaje de ruido 30%

# CONCLUSIONES HOPFIELD

- A mayor ruido, la cantidad de iteraciones necesarias para converger aumenta
- Existen patrones más ortogonales que otros
- Se pueden almacenar únicamente 4 patrones (15% dimension del patrón), los cuales conviene que sean lo más ortogonales posibles
- Como hay solo 25 bits (1 o -1) para representar cada letra, entonces estas no pueden ser realmente ortogonales

# VARIABLES AISLADAS





# SENSIBILIDAD DE PARÁMETROS

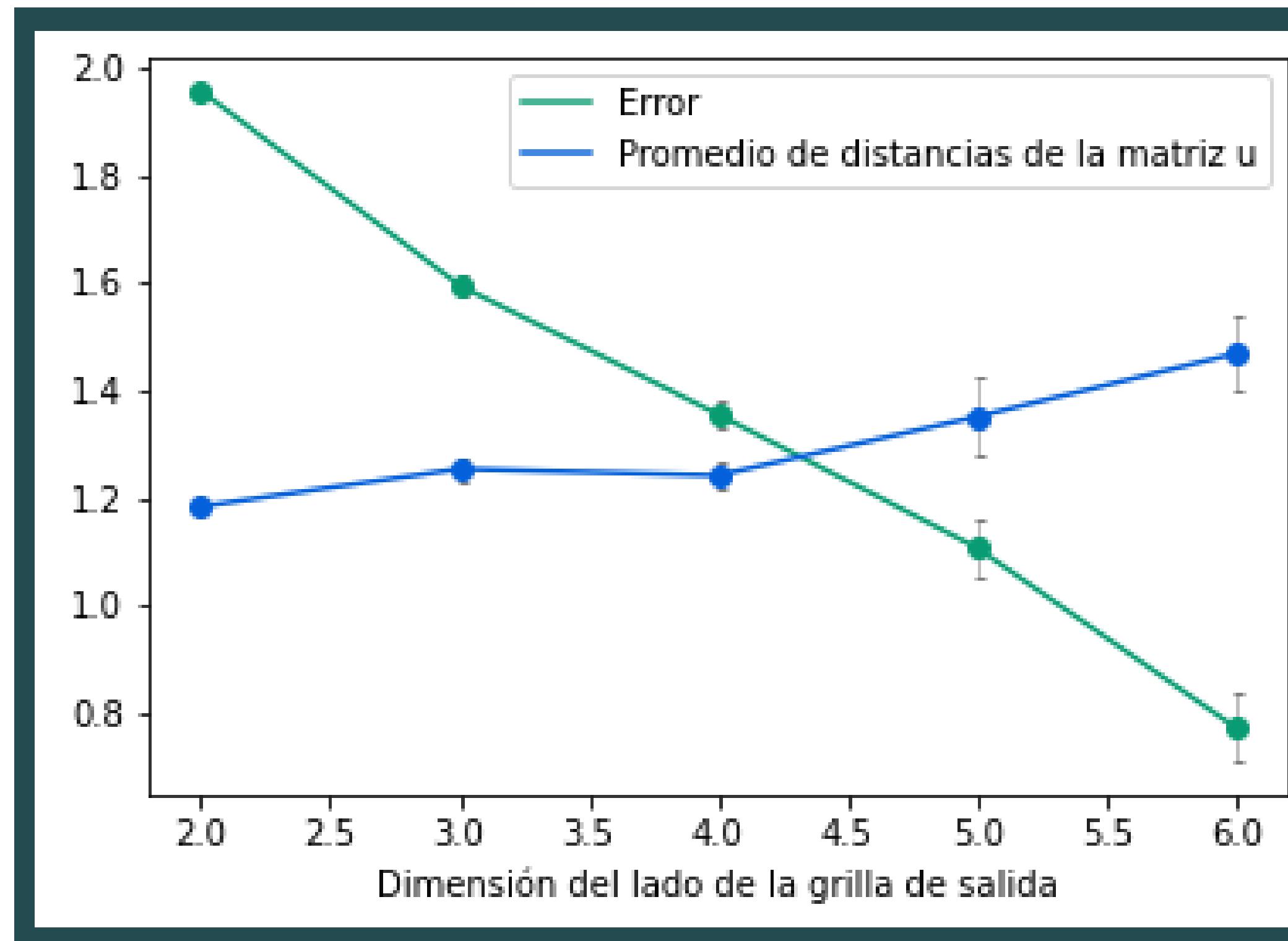
Vamos a analizar cómo afectan:

- La dimensión de la grilla de salida
- Inicialización de pesos

## Métricas:

- Quantization error (promedio de distancias entre cada input y los pesos de la neurona ganadora)
- Promedio de distancias entre los pesos de una neurona con sus vecinos

# DIMENSIÓN GRILLA DE SALIDA

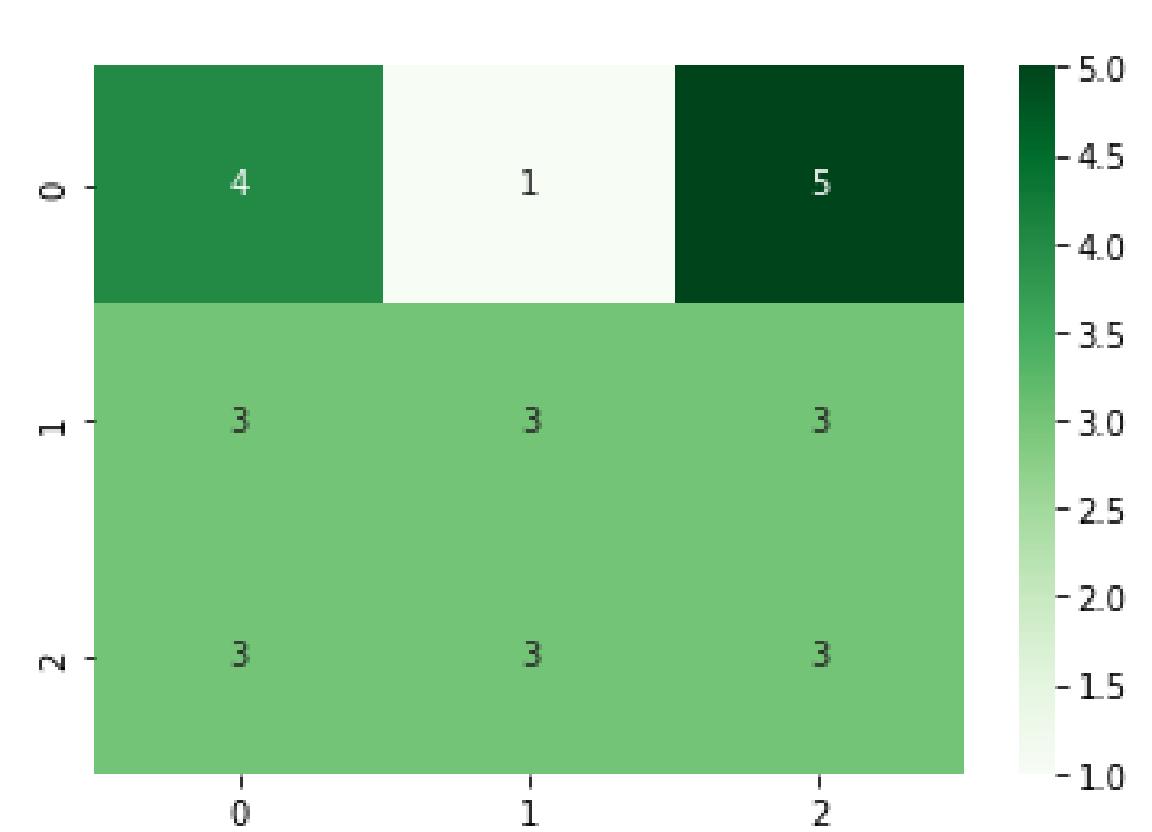
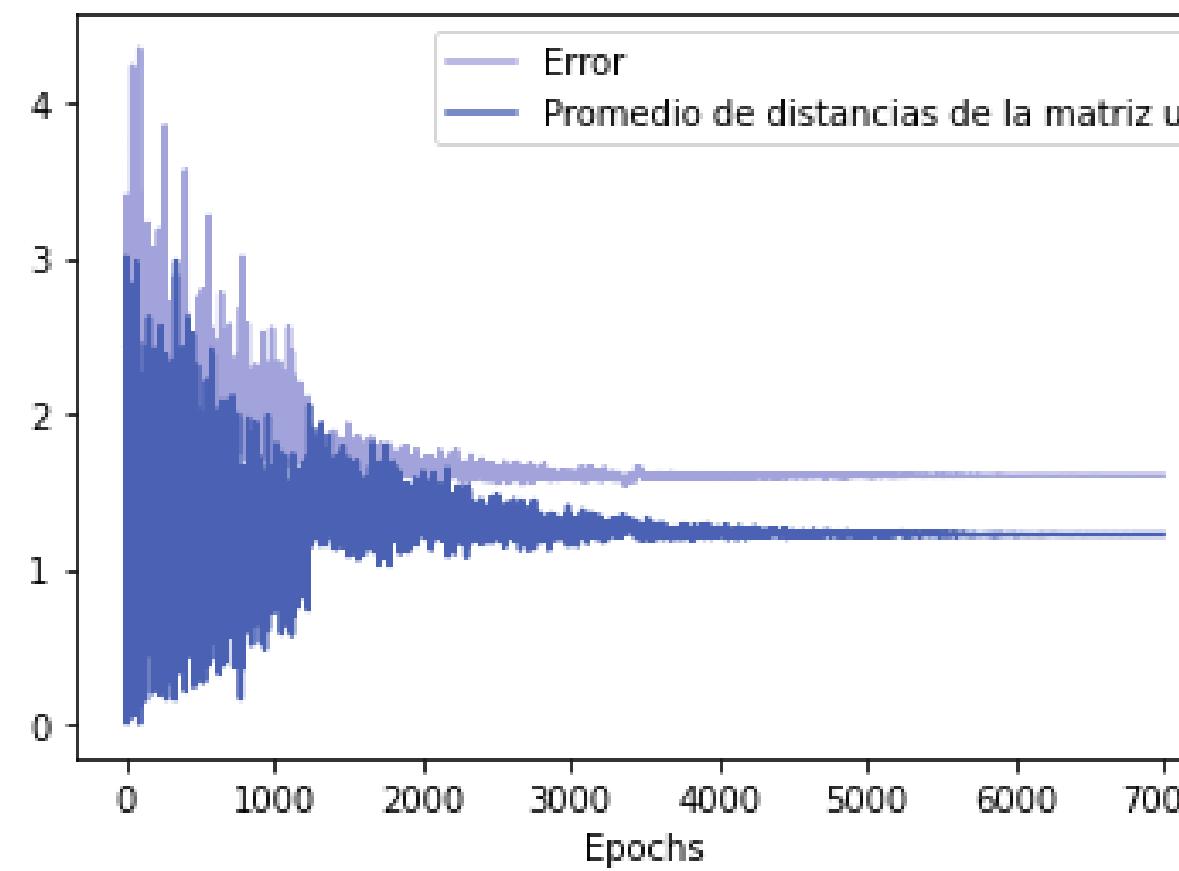


- Límite de épocas: 2100
- Radio inicial: mitad del lado de la grilla de salida
- Actualización radio: exponencial
- Tasa de aprendizaje inicial: 1
- Actualización tasa de aprendizaje: exponencial
- Inicialización de pesos: valores de entrada

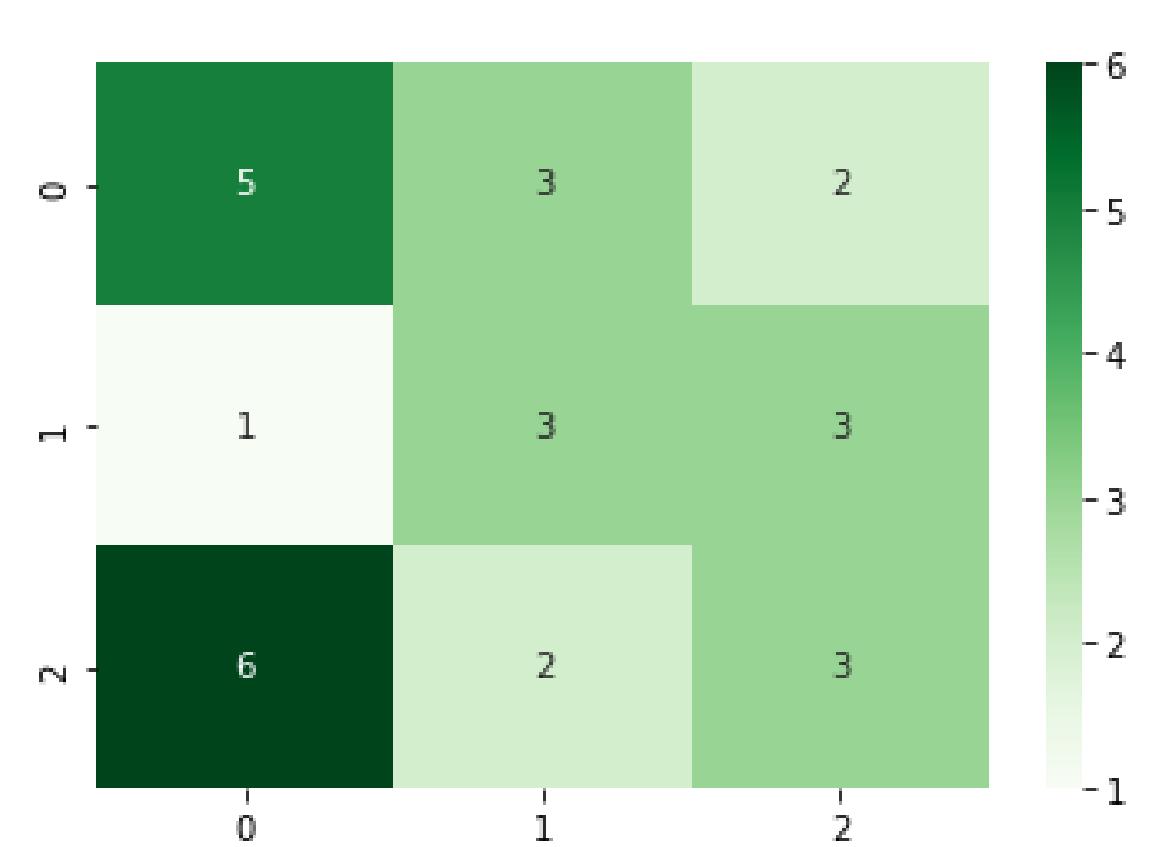
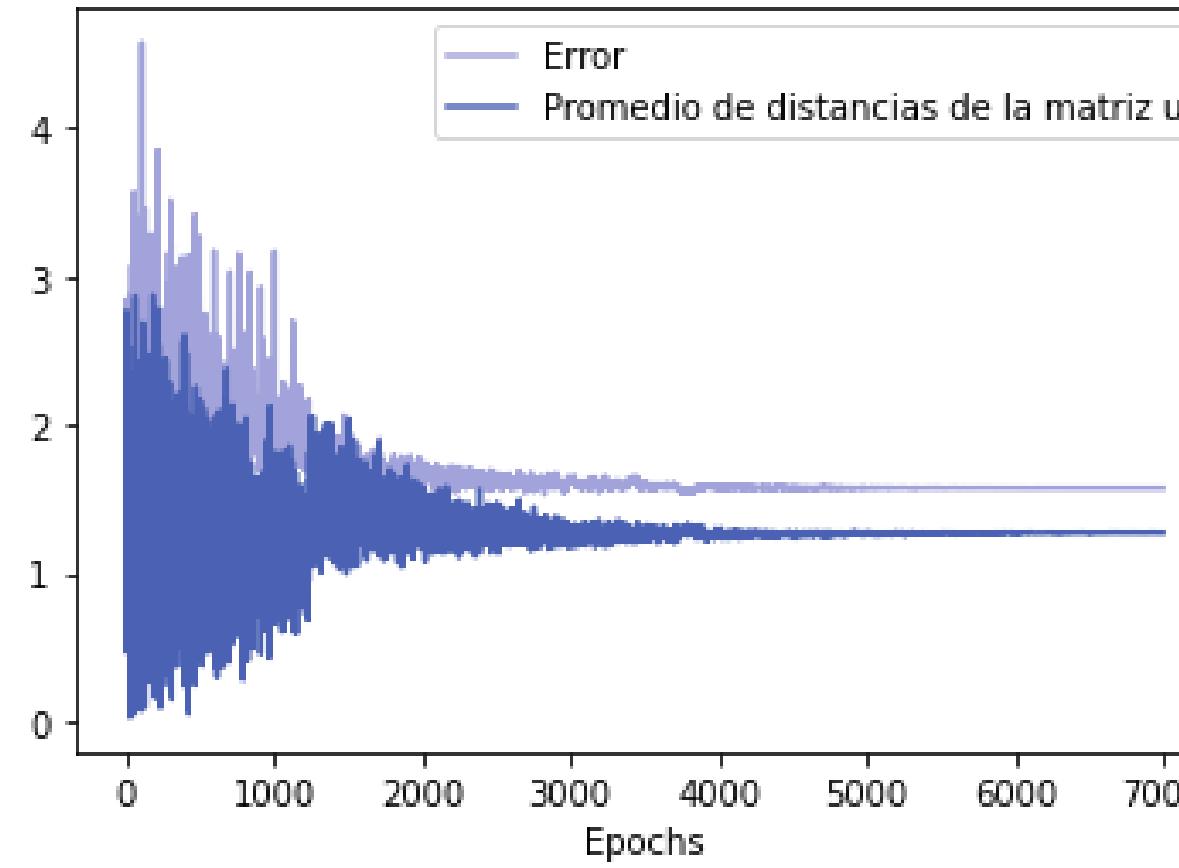
# INICIALIZACIÓN DE PESOS

- Límite de épocas: 2100
- Radio inicial: mitad del lado de la grilla de salida
- Actualización radio: exponencial
- Tasa de aprendizaje inicial: 1
- Actualización tasa de aprendizaje: exponencial
- Dimensión grilla de salida: 3x3

VALORES DE ENTRADA



RANDOM





**FIN**