



# Aprendizaje No Supervisado

## Sistemas de Inteligencia Artificial

---

- Banfi, Malena
- Fleischer, Lucas
- Perez Rivera, Mateo
- Szejer, Ian

Grupo 3



# Tabla de contenido

01

---

Red de  
Kohonen

02

---

Regla de Oja

03

---

Modelo de  
Hopfield

# Ejercicio 1

---

El conjunto de datos europe.csv corresponde a características económicas, sociales y geográficas de 28 países de Europa. Las variables son:

- Country : Nombre del país.
- Area: area.
- GDP: producto bruto interno.
- Inflation: inflación anual.
- Life.expect: expectativa de vida media en anos.
- Military: presupuesto militar.
- Pop.growth: tasa de crecimiento poblacional.
- Unemployment: tasa de desempleo.





# 01

---

## Red de Kohonen

# Ejercicio 1.1

---

Implementar la red de Kohonen y aplicarla para resolver los siguientes problemas:

- Asociar países que posean las mismas características geopolíticas, económicas y sociales.
- Realizar al menos un gráfico que muestre los resultados.
- Realizar un gráfico que muestre las distancias promedio entre neuronas vecinas.
- Analizar la cantidad de elementos que fueron asociados a cada neurona.



# Como podemos solucionar esto con kohonen

- Kohonen nos permite reducir la dimensionalidad del problema a uno de  $K \times K$  dimensiones
- Utiliza el aprendizaje competitivo para agrupar en neuronas los países que comparten características sociopolíticas similares
- Las neuronas además tienen un vecindario, cuyo radio inicia en un valor  $X$  y disminuye hasta un mínimo  $Y$
- Al ingresar un nuevo dato a nuestra red, se evalúa su distancia euclidiana a todos los pesos, dando una neurona ganadora, cuyo vector de pesos es más similar a la entrada

**Distancia Euclídea:**

$$W_k = \arg \min_{1 \leq j \leq N} \{ \| X^p - W_j \| \}$$



Luego la neurona ganadora y su vecindario, según la regla de kohonen se actualizan los pesos de la neurona y su vecindario:

$$N_R(i) = \{n / \|n - n_R\| < R(i)\}$$

*Regla de kohonen*

$$\text{Si } j \in N_R(i) \rightarrow W_j^{i+1} = W_j^i + \eta(i) * (X^p - W_j^i)$$

$$\text{Si } j \notin N_R(i) \rightarrow W_j^{i+1} = W_j^i$$

*Actualización de pesos*

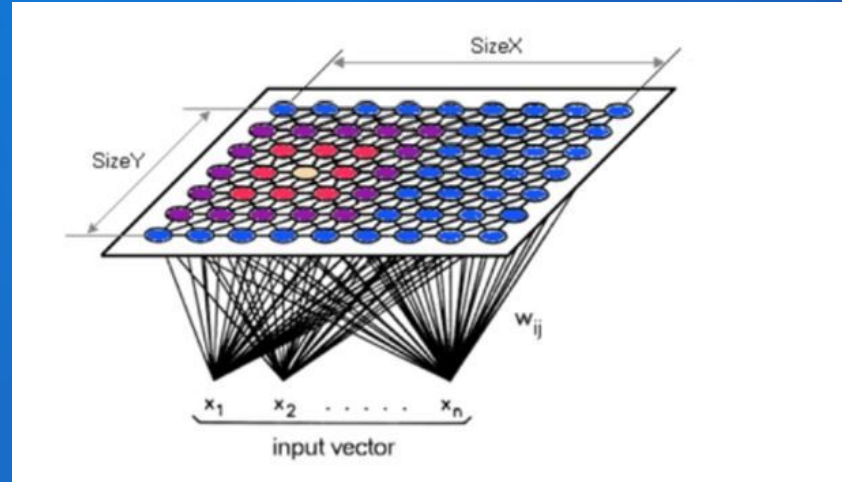
# Datos: Países de europa

Country	Area	GDP	Inflation	Life.expect	Military	Pop.growth	Unemployment
Austria	83871	41600	3.5	79.91	0.8	0.03	4.2
Belgium	30528	37800	3.5	79.65	1.3	0.06	7.2
Bulgaria	110879	13800	4.2	73.84	2.6	-0.8	9.6
Croatia	56594	18000	2.3	75.99	2.39	-0.09	17.7
Czech Republic	78867	27100	1.9	77.38	1.15	-0.13	8.5
Denmark	43094	37000	2.8	78.78	1.3	0.24	6.1
Estonia	45228	20400	5	73.58	2	-0.65	12.5
Finland	338145	38000	3.3	79.41	2	0.07	7.8
Germany	357022	38100	2.5	80.19	1.5	-0.2	6
Greece	131957	26300	3.3	80.05	4.3	0.06	17.4
Hungary	93028	19600	3.9	75.02	1.75	-0.18	10.9
Iceland	103000	38100	4	81	0	0.67	7.4
Ireland	70273	40800	2.6	80.32	0.9	1.11	14.4
Italy	301340	30500	2.9	81.86	1.8	0.38	8.4
Latvia	64589	16800	4.4	72.93	1.1	-0.6	12.8
Lithuania	65300	19100	4.1	75.55	0.9	-0.28	15.4
Luxembourg	2588	80600	3.4	79.75	0.9	1.14	5.7
Netherlands	41543	42000	2.3	80.91	1.6	0.45	4.4
Norway	323802	53400	1.3	80.32	1.9	0.33	3.3
Poland	312685	20200	4.2	76.25	1.9	-0.08	12.4
Portugal	92090	23400	3.7	78.7	2.3	0.18	12.7
Slovakia	49035	23300	3.9	76.03	01.08	0.1	13.2
Slovenia	20273	28800	1.8	77.48	1.7	-0.19	11.8
Spain	505370	30500	3.1	81.27	1.2	0.65	21.7
Sweden	450295	40700	3	81.18	1.5	0.17	7.5
Switzerland	41277	44500	0.2	81.17	1	0.92	2.8
Ukraine	603550	7200	8	68.74	1.4	-0.63	7.9
United Kingdom	243810	36500	4.5	80.17	2.7	0.55	8.1

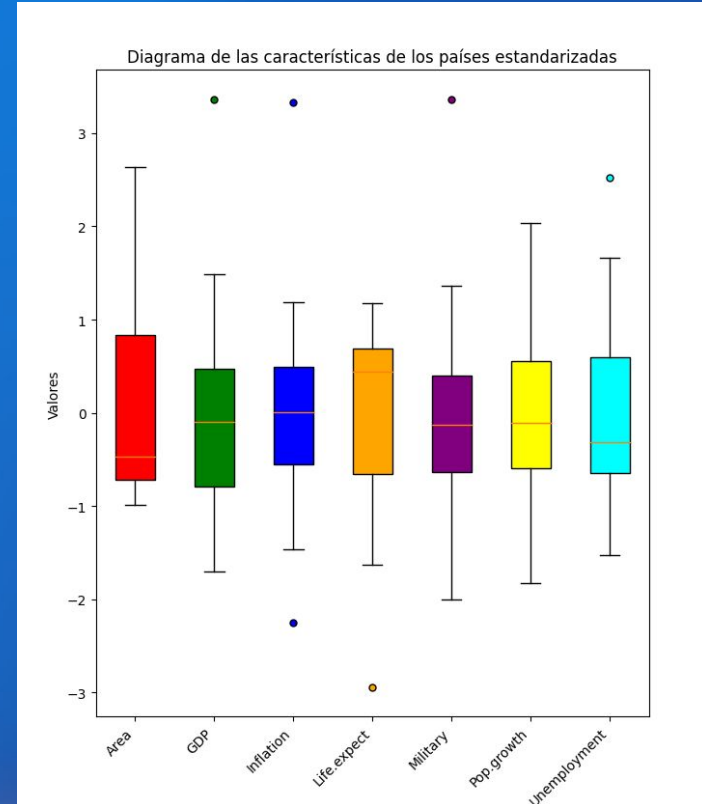
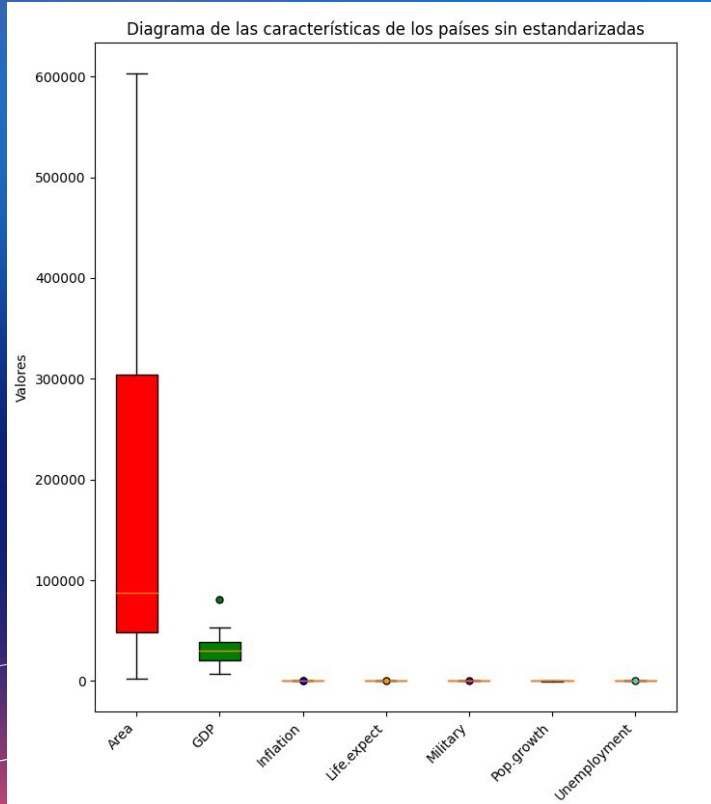


# Hiperparametros utilizados

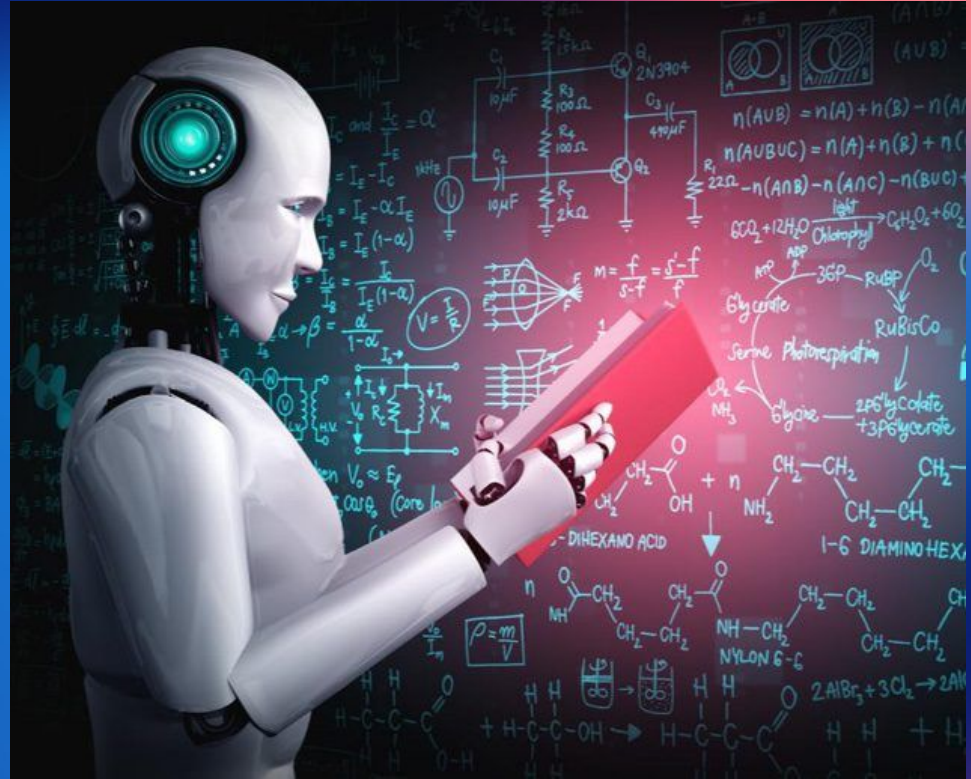
- $n$  -> Taza de aprendizaje
- $K$  -> Dimensiones de la red ( $K \times K$ )
- $R_{inicial}$  -> Radio inicial del vecindario
- $R_{final}$  -> Radio final del vecindario



# Estandarización de datos



# Resultados Obtenidos

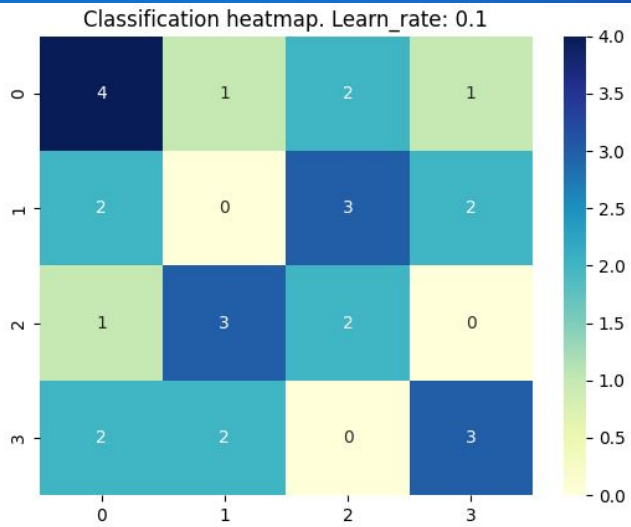
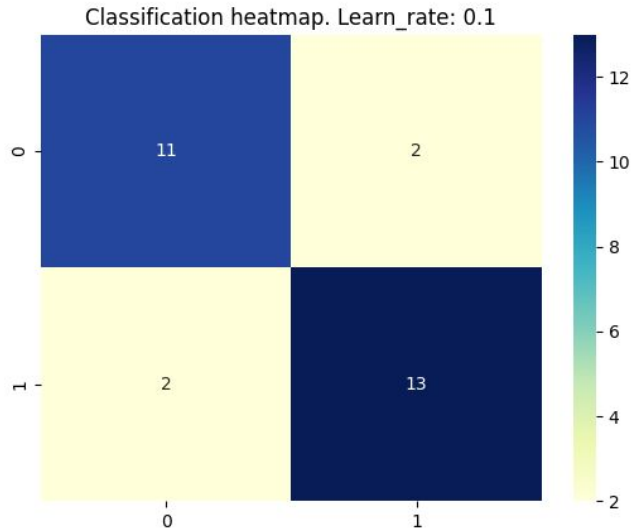


# Resultados con otras configuraciones

Podemos ver que tomando  $k = 2$ , la agrupación se divide en solo 2 grupos, reduciendo de 28 países a 2 grupos. Algo que no nos parece óptimo ya que la pérdida de información es demasiada.

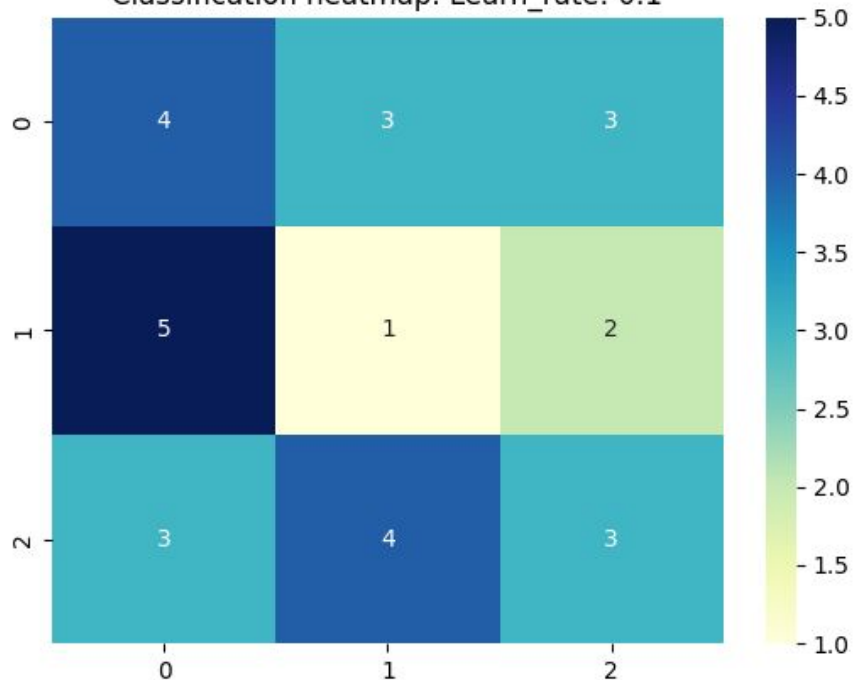
Mientras que al utilizar  $k = 4$ , se divide de forma bastante pareja pero formando 13 grupos, dividiendo los países en la mitad de la cantidad original, no nos parece suficiente disminución de dimensionalidad,

Por lo que se decidió utilizar  $k = 3$ . A continuación pasamos a ver los resultados con esta red.



# Resultados obtenidos

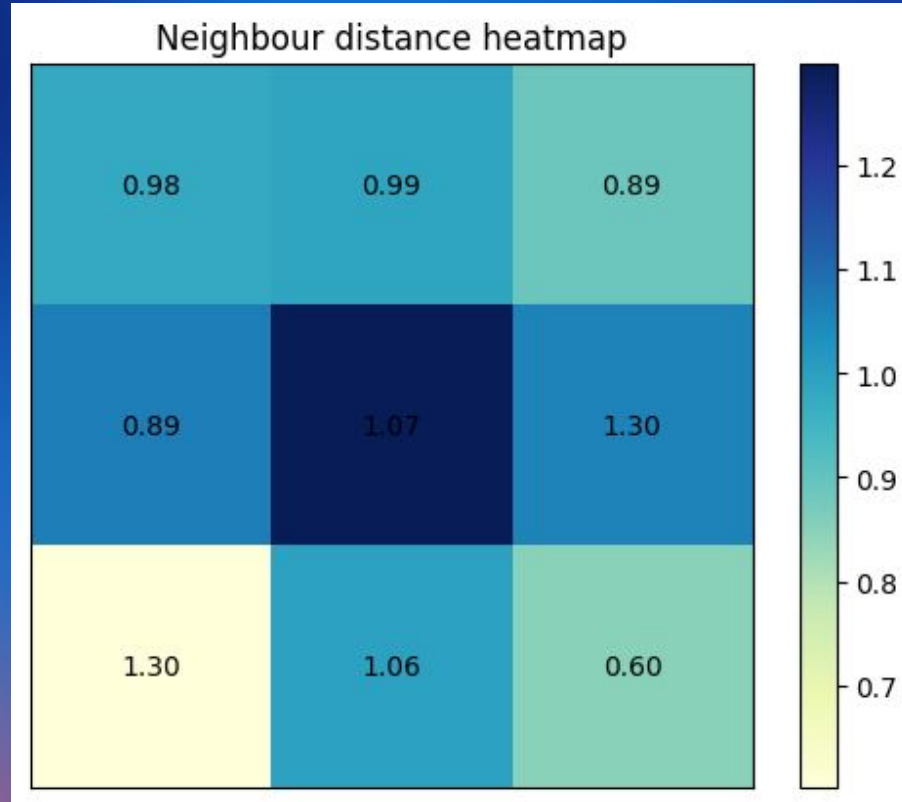
Classification heatmap. Learn\_rate: 0.1



Classification heatmap. Learn\_rate: 0.1



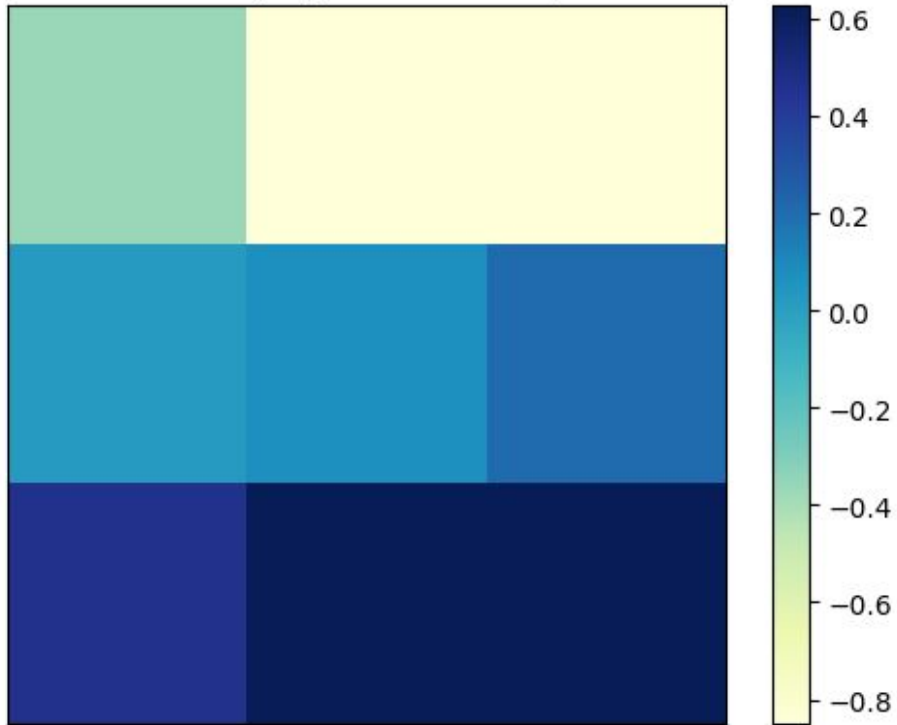
# Distancia promedio entre vecinos



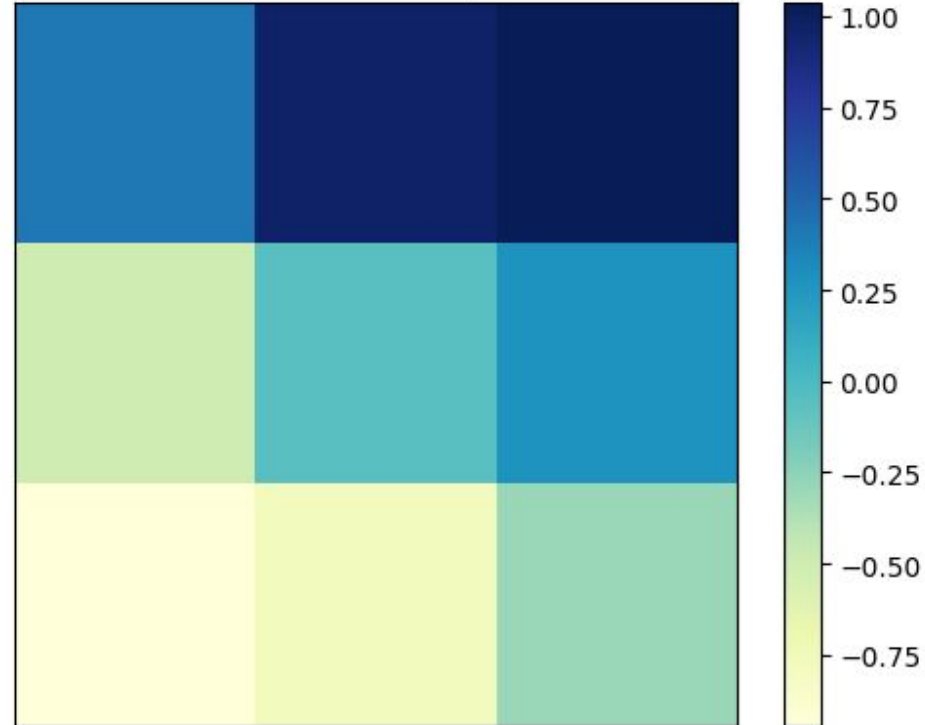


# Relación entre variables: Unemployment vs GDP

Unemployment heatmap

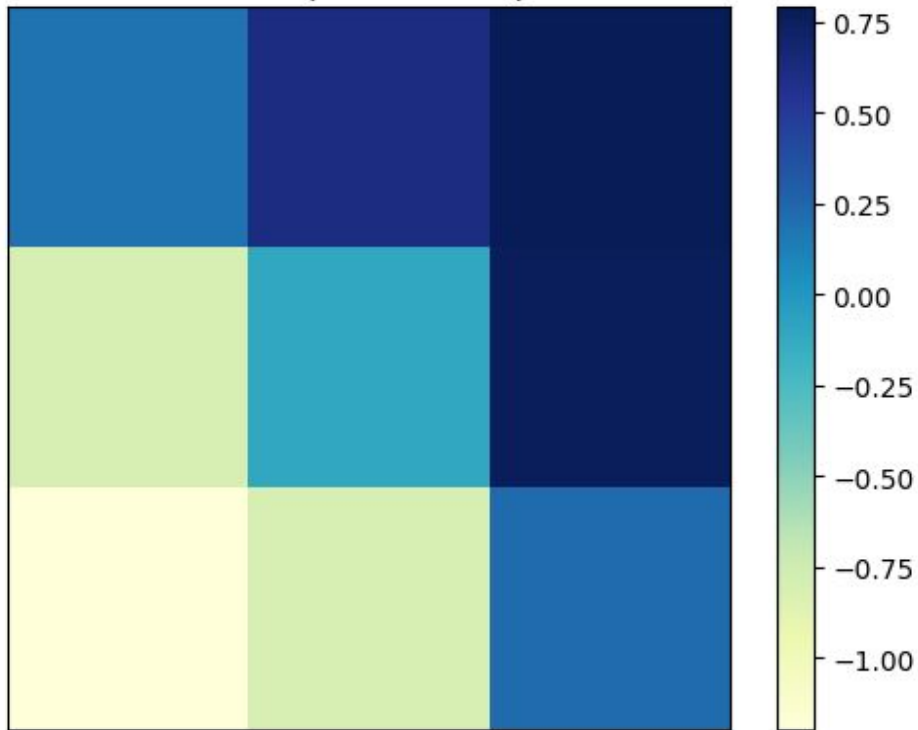


GDP heatmap

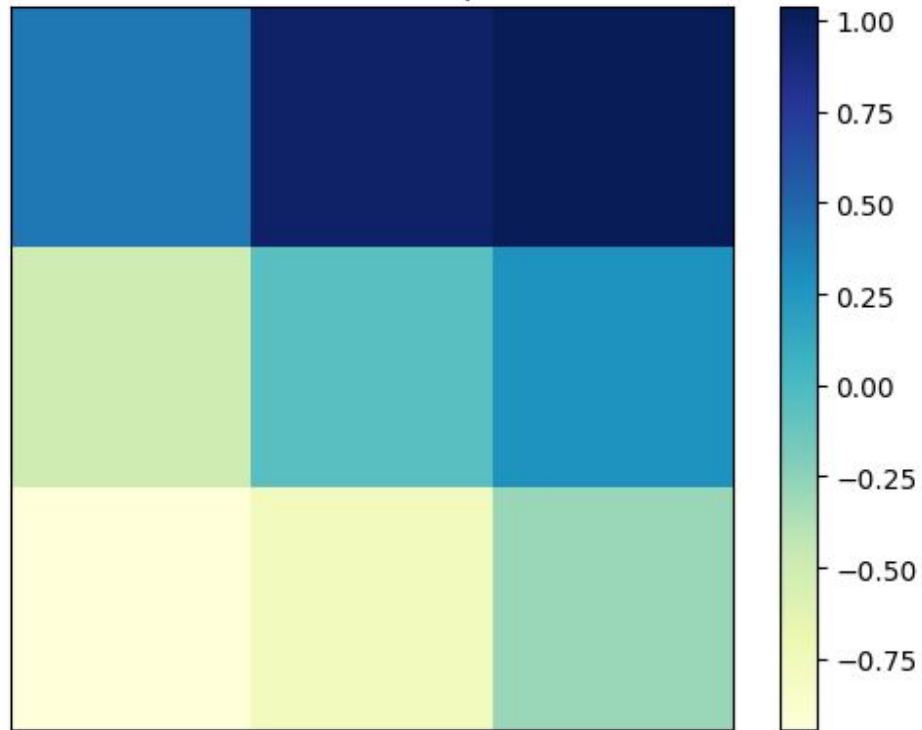


# Relación entre variables: Life expectancy vs GDP

Life.expect heatmap



GDP heatmap



# Conclusiones

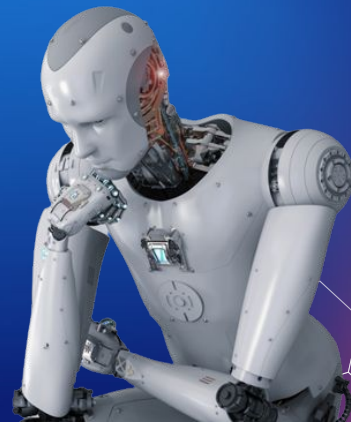
- En primera instancia pudimos obtener con el uso de la red de Kohonen una disminución de la dimensionalidad, pero sin pérdidas significativas de la topología de los datos originales.
- Si comparamos los resultados obtenidos con un análisis superficial de los datos podemos ver lo siguiente:

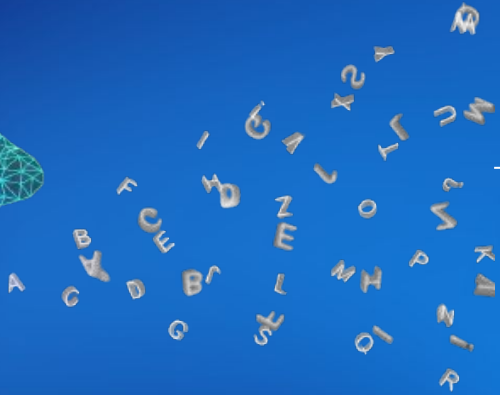
Por ejemplo el grupo formado por Luxemburgo, Netherlands, Norway y Switzerland

Luxembourg	2588	80600	3.4	79.75	0.9	1.14	5.7
Netherlands	41543	42000	2.3	80.91	1.6	0.45	4.4
Norway	323802	53400	1.3	80.32	1.9	0.33	3.3
Switzerland	41277	44500	0.2	81.17	1	0.92	2.8

Estos 4 países agrupados por Kohonen, comparten altamente las características: Life. expect, military y Unemployment. Por lo que agruparlos juntos parece una buena opción

- También la red nos proporcionó información de la correlación entre algunas
- Características de los países, como por ejemplo:
  - Neuronas con alta unemployment suelen tener además bajo GDP
  - Neuronas con alto life.expect tienen también alto GDP





02

---

# Regla de Oja



# Componentes Principales

## Eliminar redundancia

---

Busca eliminar la redundancia de información de un set de datos

## Mayor variabilidad de datos


---

Se busca obtener una cantidad 'q' de variables, que sean combinación lineal de las originales pero que recojan la mayor variabilidad de los datos posibles

## Variables no correlacionadas

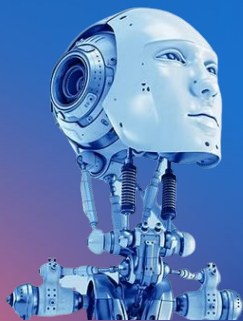
---

La idea es encontrar una cantidad 'q' de variables menor a la inicial y que no estén correlacionadas



# Regla de Oja

- Es un perceptrón simple lineal (no supervisado) con una sola capa de salida en la que se van actualizando los pesos según la regla de Oja.
- Luego de varias iteraciones el método converge al autovector correspondiente al mayor autovalor de la matriz de correlaciones de los datos de entrada. Con este vector w final se construye la primera componente principal.
- Garantiza convergencia al autovector correspondiente al mayor autovalor de la matriz de correlaciones de los datos de entrada.
- Las componentes principales permiten extraer características importantes de un conjunto de datos reduciendo su dimensionalidad, permitiendo realizar un análisis de forma más simple y fácil de comprender.





# Ejercicio 1.2

---

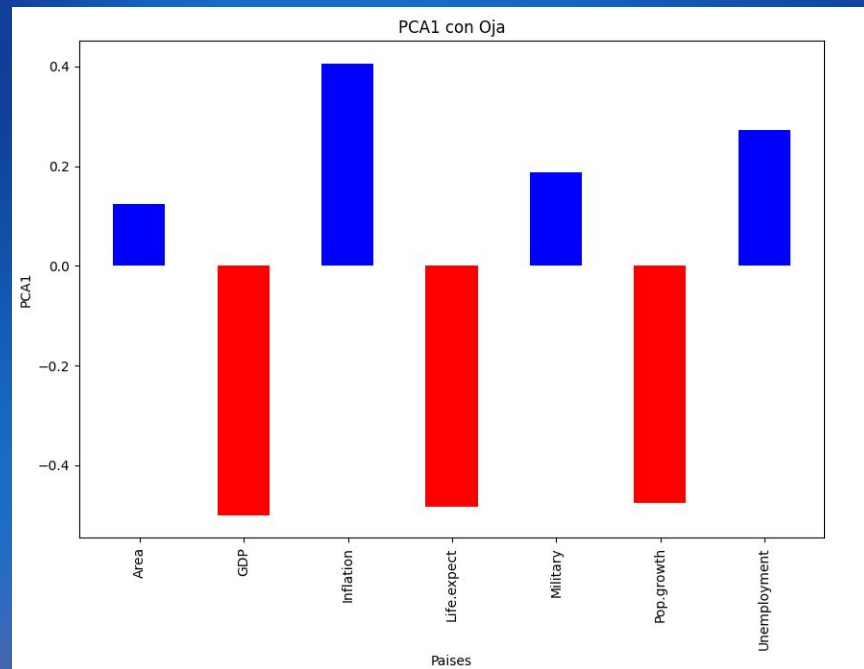
Implementar una red neuronal utilizando la regla de Oja para resolver los siguientes problemas:

- Calcular la primer componente principal para este conjunto de datos.
- Interpretar el resultado del primer componente.
- Comparar el resultado del ejercicio de Oja con el resultado de calcular la primer componente principal con una librería.





# Componente Principal y Autovector

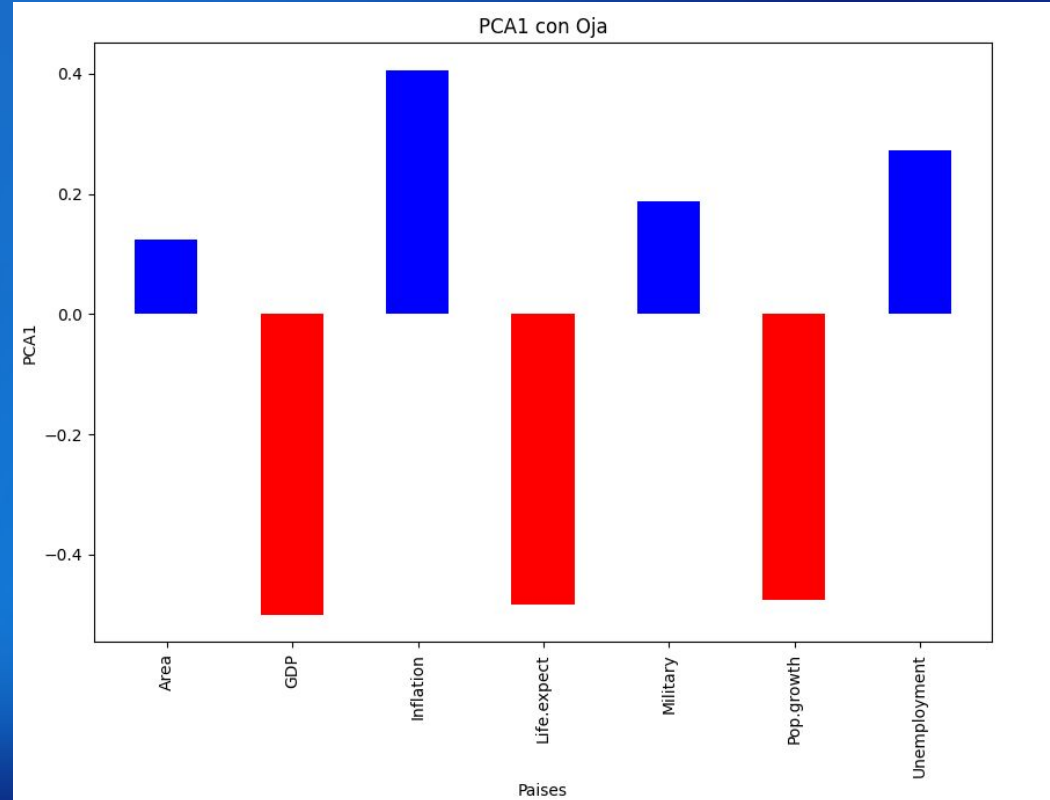


Autovector = [0.12494283 -0.50049984 0.40658603 -0.48288751 0.18805402 -0.475689 0.27162229]

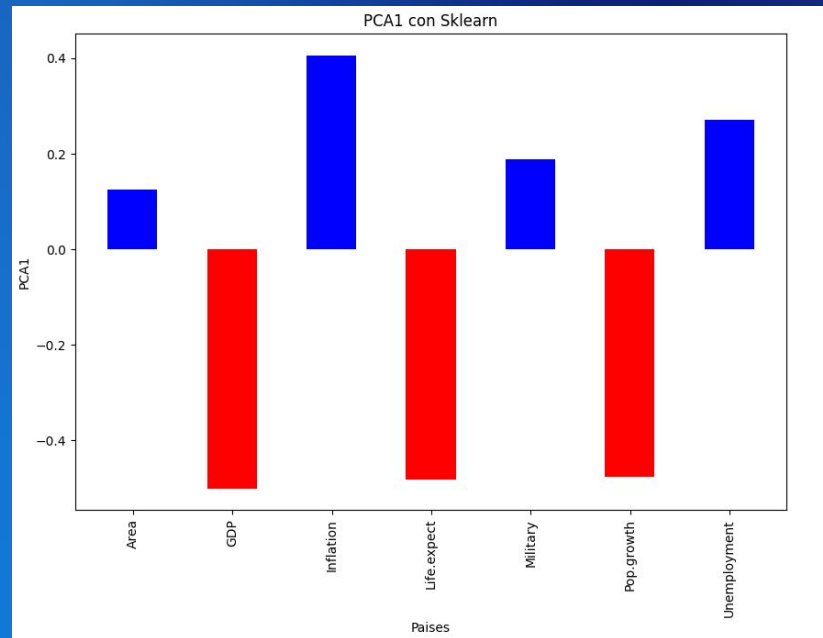
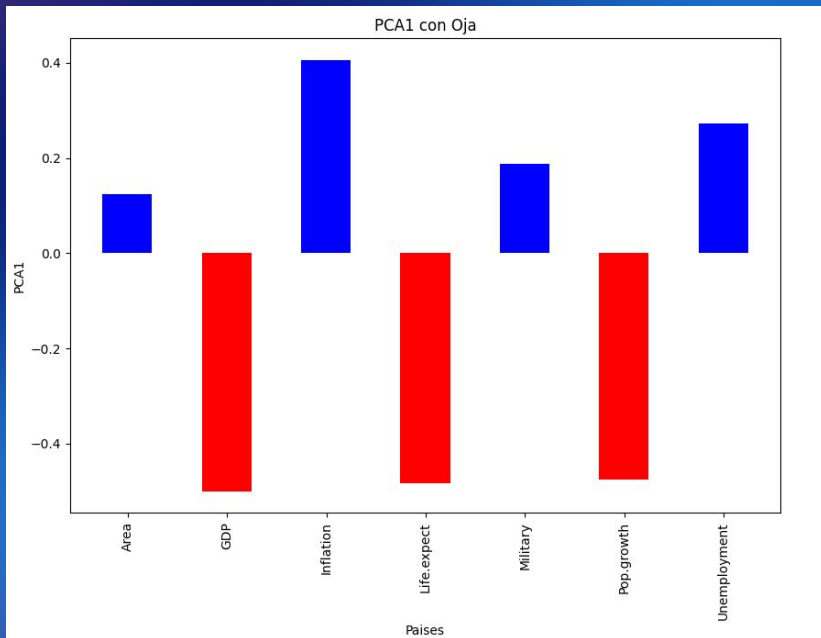
# Interpretación de la Primer Componente

La primer componente nos permite transformar las 7 variables en un solo número que se caracteriza de la siguiente manera:

- Se ve influenciado positivamente por:
  - Area
  - Inflation
  - Military
  - Unemployment
- Se ve influenciado negativamente por:
  - GDP
  - Life Expect
  - Pop. Growth



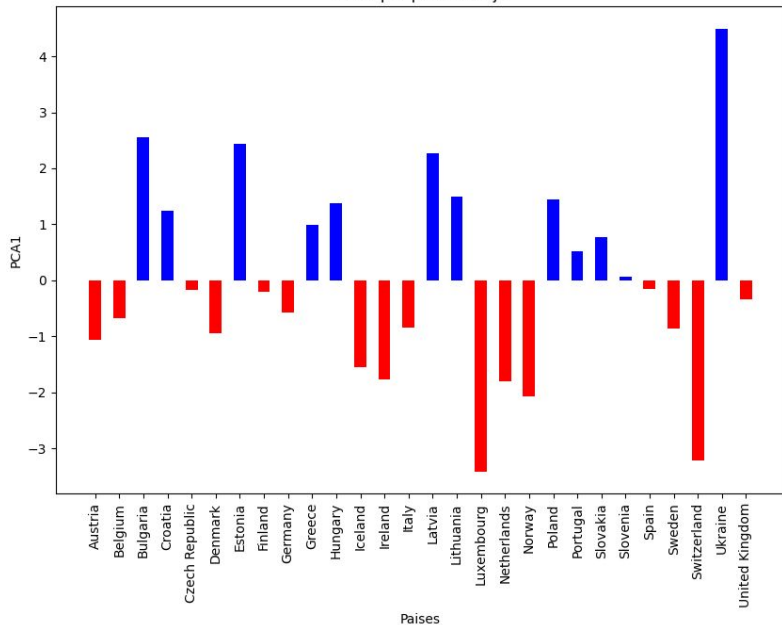
# Comparación con la librería Sklearn



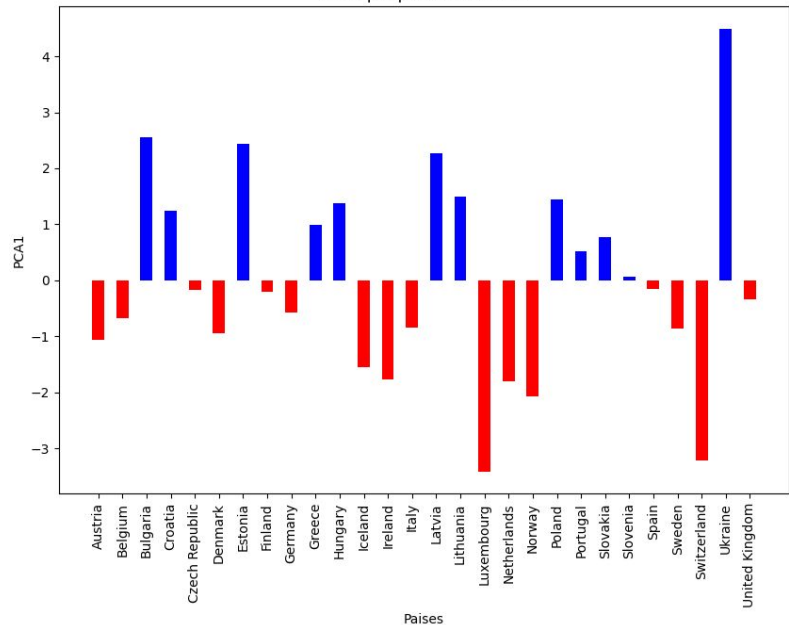
Autovector Oja = [0.12494283 -0.50049984 0.40658603 -0.48288751 0.18805402 -0.475689 0.27162229]  
Autovector Sklearn = [0.1248739 -0.50050586 0.40651815 -0.48287333 0.18811162 -0.47570355 0.27165582]  
Error = [0.00006893 0.00000602 0.00006788 0.00001418 0.0000576 0.00001455 0.00003353]

# Comparación con la librería Sklearn

PCA1 por pais con Oja



PCA1 por pais con Sklearn



Error: [ 0.00006588 0.00013289 0.00006584 0.00021067 0.00006933 0.00002553 0.00002391 0.00005092  
0.0000584 0.00027606 0.00002078 0.00014422 0.00003464 0.00002099 0.00002072 0.00000826 0.0000582  
0.00001207 0.00000016 0.00006138 0.00008779 0.00002242 0.00016312 0.0000749 0.00011814 0.00011801  
0.00004473 0.00026119]



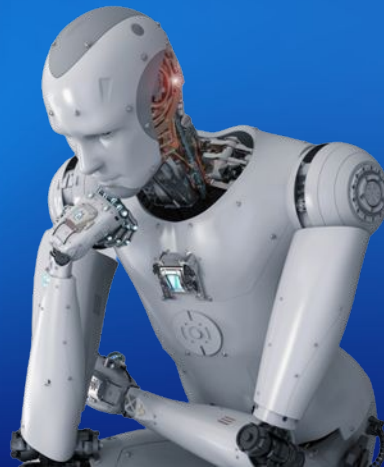
# Elección del learning rate

- Para elegir el mejor learning rate, corrimos el algoritmo varias veces con distintos valores:
  - Learning rate = 0.001
    - Norma de diferencia de la primer componente = 2.0013
    - Norma de diferencia de autovectores de países = 18.6813
  - Learning rate = 0.0001
    - Norma de diferencia de la primer componente = 2.000131935759368
    - Norma de diferencia de autovectores de países = 18.670291874250783
  - Learning rate = 0.00001
    - Norma de diferencia de la primer componente = 0.000119
    - Norma de diferencia de autovectores de países = 0.00066

Por lo tanto, el learning rate elegido fue de 0.00001

# Conclusiones

- La diferencia (error) entre la implementación de Oja y la librería Sklearn es:
  - Norma de diferencia de la Primer Componente: 0.000119
  - Norma de diferencia de Autovectores de países = 0.000663
- Es muy importante estandarizar los datos ya que se puede observar más con detalle cada variable y se ve la variabilidad que tiene cada una de las variables.
- La tasa de aprendizaje debe ser baja para asegurar la convergencia





03

---

# Modelo de Hopfield



# Ejercicio 2

---

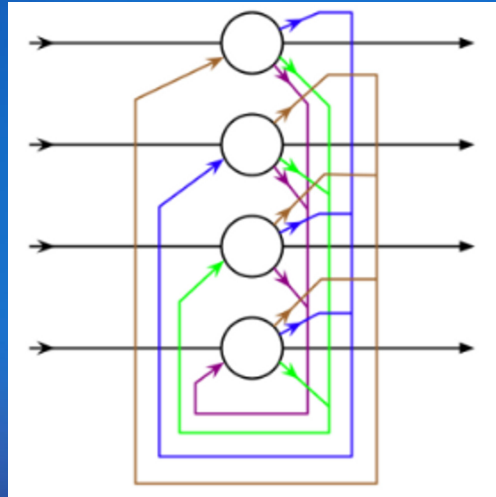
Construir patrones de letras del abecedario utilizando 1 y  $-1$  y matrices de  $5 \times 5$ .

- A. Almacenar 4 patrones de letras. Implementar el modelo de Hopfield para asociar matrices ruidosas de  $5 \times 5$  con los patrones de las letras almacenadas. Los patrones de consulta deben ser alteraciones aleatorias de los patrones originales. Mostrar los resultados que se obtienen en cada paso hasta llegar al resultado final.
- B. Ingresar un patrón muy ruidoso e identificar un estado espurio.



# Modelo de Hopfield

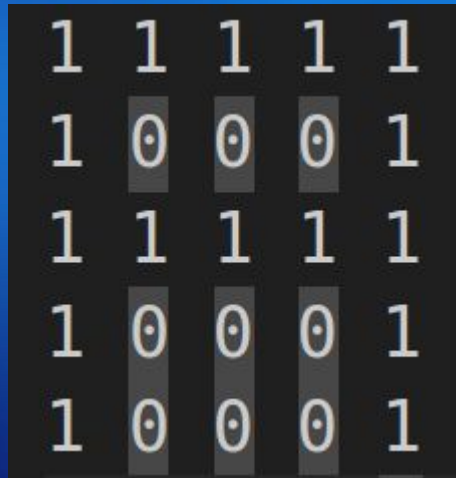
- Las redes de Hopfield son un sistema de neuronas conectadas todas entre sí.
- Ninguna neurona está conectada con si misma.
- Todas las neuronas están en una sola capa de entrada y salida.
- Podemos entrenarlas con un conjunto de patrones que pueden ser almacenados en la red y luego reconocidos, con un nivel de eficiencia que depende que se cumplan ciertos requisitos



# Patrones numericos

Tenemos patrones de números 5x5, representados con 1 y 0. Los ceros serán almacenados como -1.

La idea es almacenar 4 patrones de letras y luego poder asociar matrices ruidosas con los patrones almacenados.



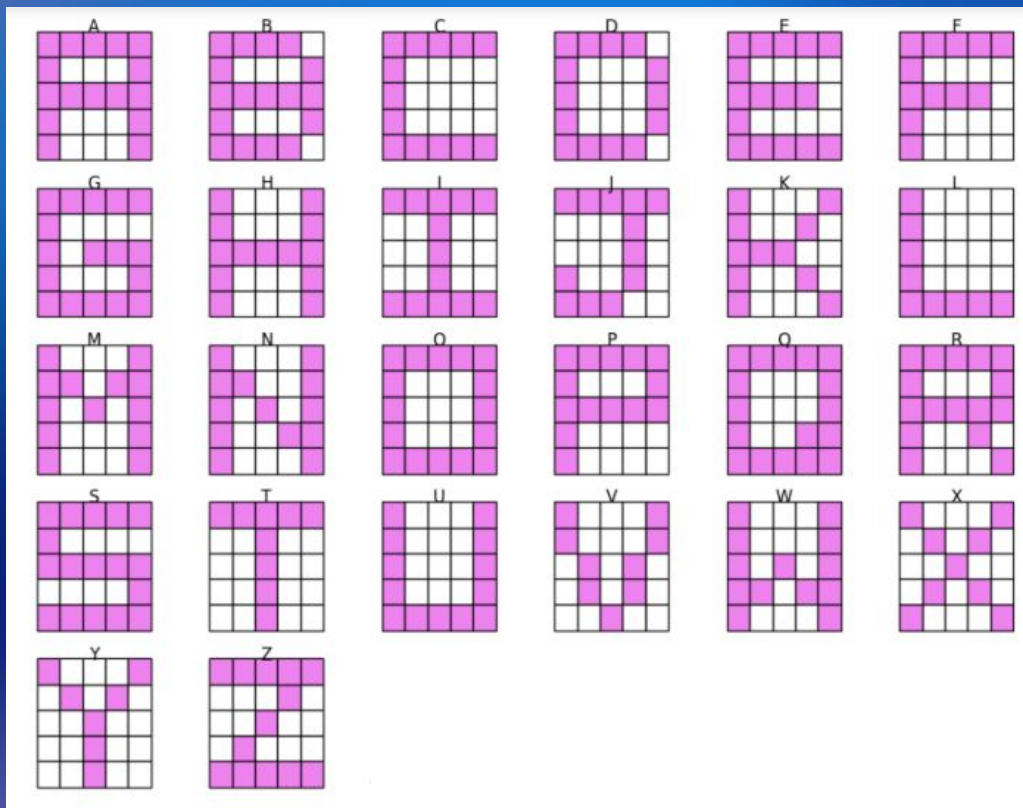
1	1	1	1	1
1	0	0	0	1
1	1	1	1	1
1	0	0	0	1
1	0	0	0	1



*	*	*	*	*
*				*
*	*	*	*	*
*				*
*				*



# Abecedario de letras



# Características

- Hopfield demostró que su red está asociada a una función de energía y que los mínimos locales de esta función son los patrones almacenados.
- Existen mínimos locales que no son los patrones almacenados, se llaman estados espurios.
- Los estados espurios son estados estables o puntos fijos que no corresponden a ninguno de los patrones almacenados en la red.
- Cuando se almacenan patrones en una red de Hopfield, se espera que, al presentar una versión parcial o ruidosa de uno de esos patrones, la red se recupere y converja al patrón original. Sin embargo, en algunos casos, la dinámica de la red puede llevar a estados estables diferentes de los patrones almacenados, a los que se les llama estados espurios.

# Ortogonalidad de los patrones

$ \langle, \rangle $ medio	grupo
1.33	('A', 'L', 'T', 'V')
1.33	('E', 'T', 'V', 'W')
1.33	('L', 'R', 'T', 'X')
1.33	('R', 'T', 'V', 'W')
1.67	('A', 'J', 'L', 'V')
1.67	('A', 'T', 'V', 'W')

Gran diferencia entre letras

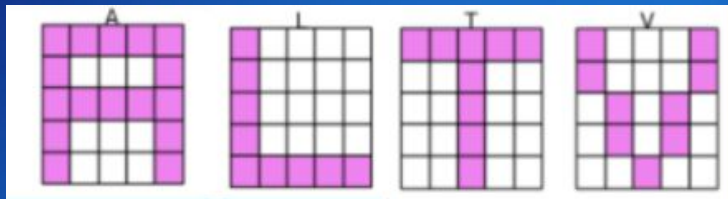
.

.

$ \langle, \rangle $ medio	grupo
18.67	('C', 'G', 'O', 'Q')
19.00	('C', 'D', 'O', 'Q')
19.00	('D', 'G', 'O', 'Q')
19.00	('D', 'O', 'Q', 'U')
19.67	('A', 'F', 'P', 'R')

Mucha coincidencia entre letras

# Entrenamiento con el patrón ortogonal



1.33

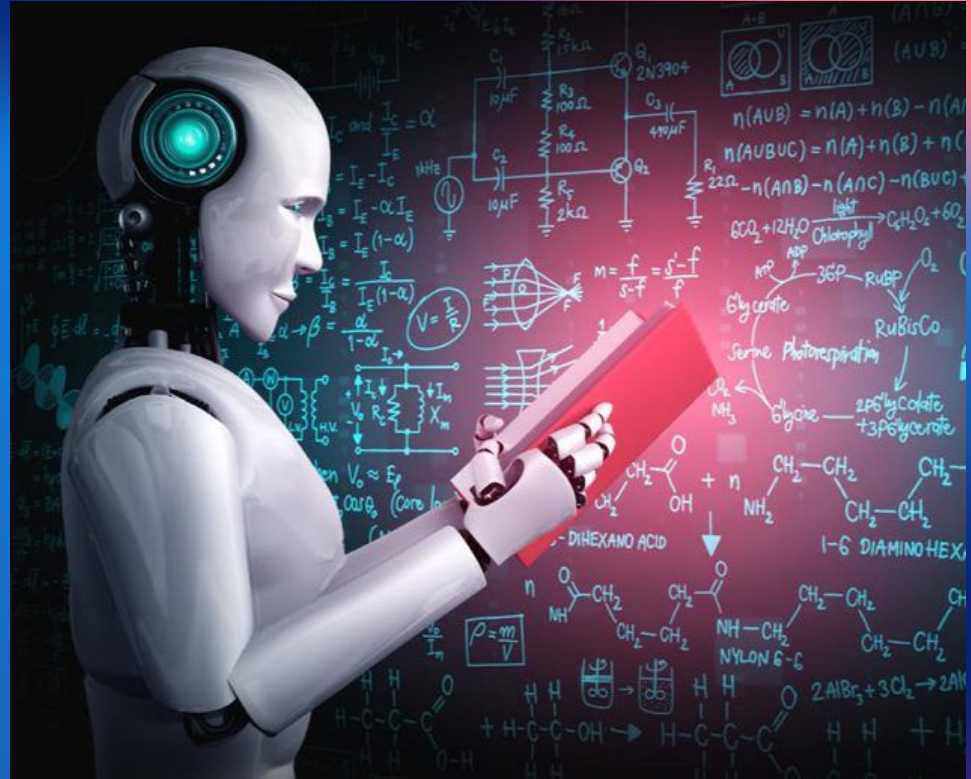
## Mutación de patrones

```
noise = 6  
letter_with_noise = mutate_pattern(letters[0], noise)
```

Letter to predict with noise: 6



# Resultados Obtenidos





# Resultados sin aplicar mutaciones

Letter to predict:

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

Iteration: 1

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

Letter predicted in 1 iteration/s

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

Letter to predict:

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

Iteration: 1

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

Letter predicted in 1 iteration/s

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

Letter to predict:

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

Iteration: 1

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

Letter predicted in 1 iteration/s

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

Letter to predict:

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

Iteration: 1

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

Letter predicted in 1 iteration/s

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

Se puede observar que sin ninguna mutación aplicada a los 4 patrones, el algoritmo es capaz de identificar de manera correcta.



# Resultados letra "A" con ruido 8 aplicado

Letter to predict with noise: 8

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

Iteration: 1

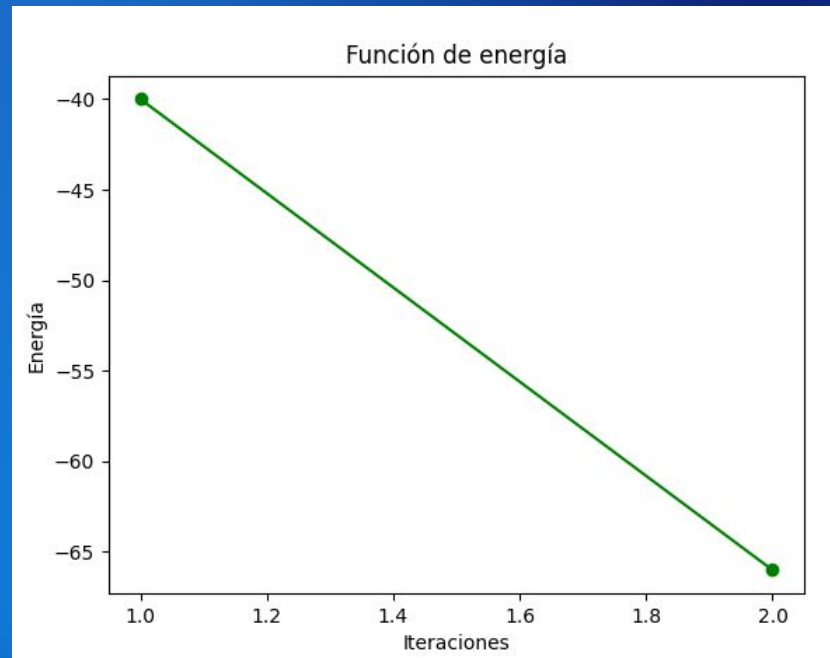
```
* * * * *  
*          *  
* * * * *  
*          *  
*          *
```

Iteration: 2

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

Letter predicted in 2 iteration/s

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



# Resultados letra "L" con ruido 8 aplicado

Letter to predict with noise: 8

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

Iteration: 1

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

Iteration: 2

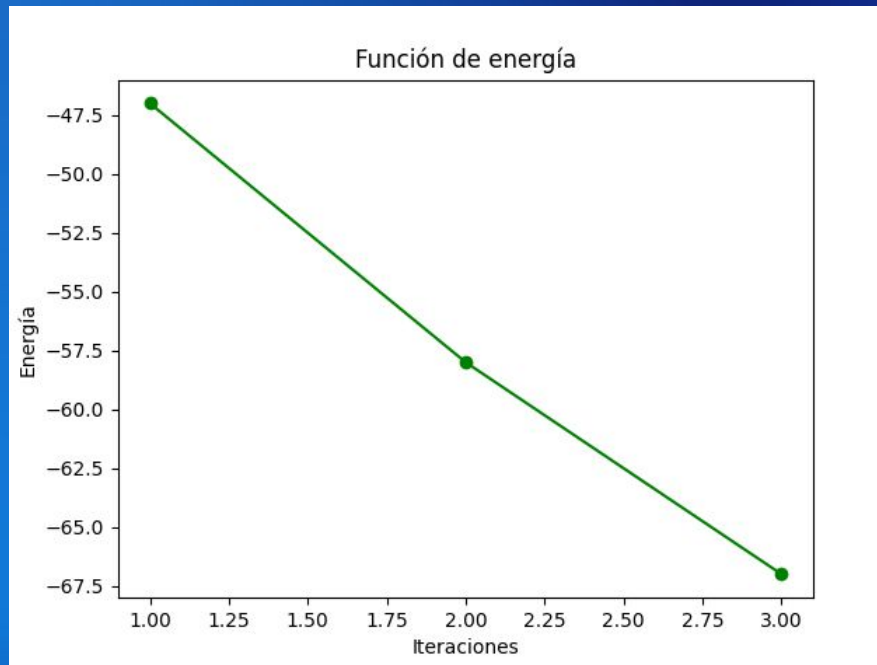
```
*  
*  
*      *  
*  
* * * * *
```

Iteration: 3

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

Letter predicted in 3 iteration/s

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



# Resultados letra "T" con ruido 8 aplicado

Letter to predict with noise: 8

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

Iteration: 1

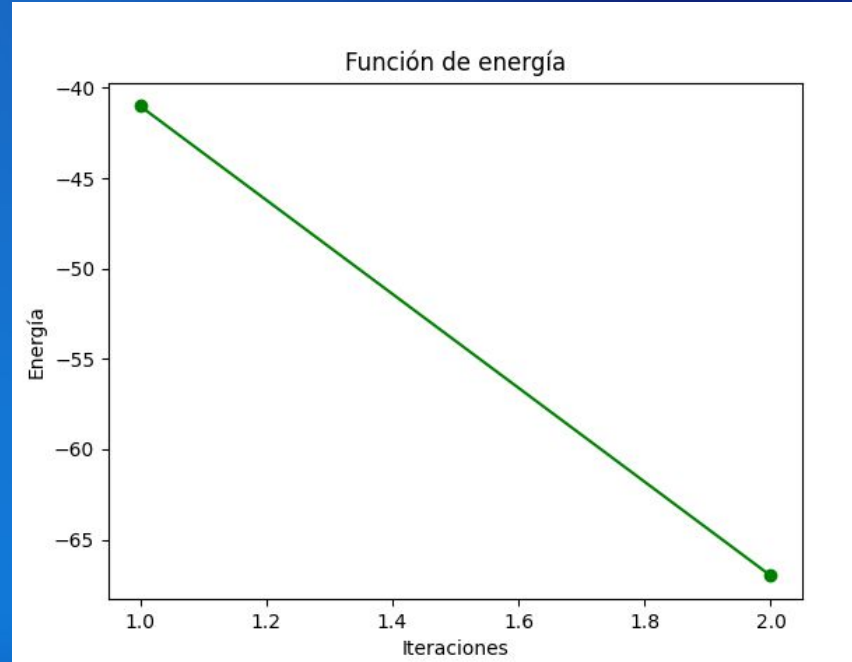
```
* * * *  
  
  *  
  *
```

Iteration: 2

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

Letter predicted in 2 iteration/s

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



# Resultados letra "V" con ruido 8 aplicado

```
Letter to predict with noise: 8
*   *
* * * *
*   * *
   * *
  * *

Iteration: 1

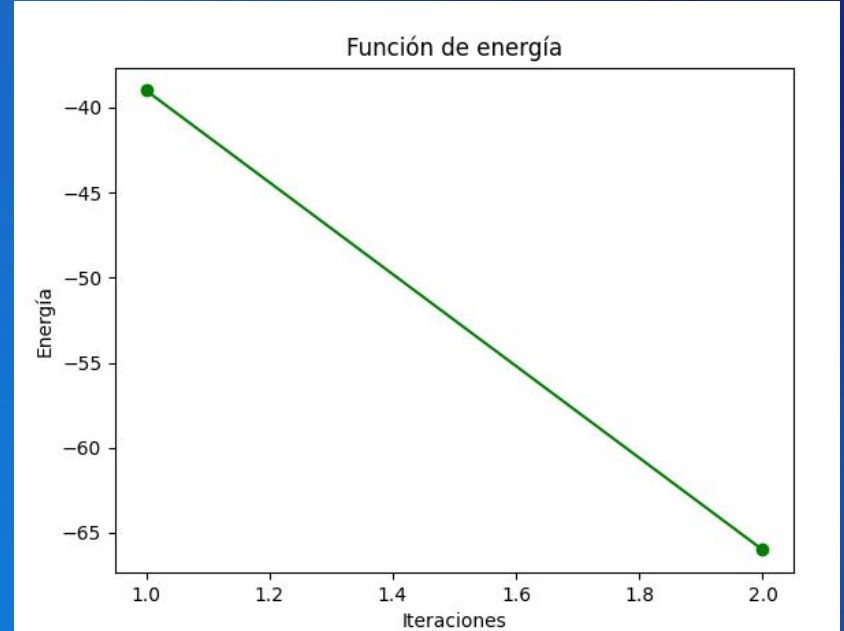
   *
  * *
 * *
  *

Iteration: 2

*   *
*   *
*   *
 * *
 * *
  *

Letter predicted in 2 iteration/s

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



Se puede observar que con ruido 8 aplicado a los 4 patrones, el algoritmo es capaz de identificar de manera correcta.

# Resultados letra "A" con ruido 10 aplicado

Letter to predict with noise: 10

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

Iteration: 1

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

Iteration: 2

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

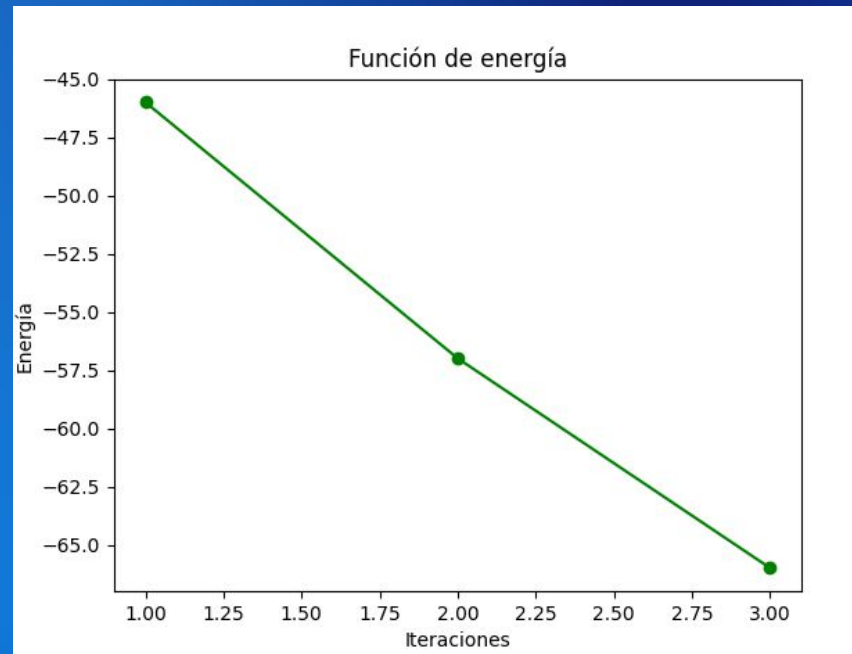
Iteration: 3

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

Letter predicted in 3 iteration/s

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

Estado espurio



# Resultados letra "T" con ruido 10 aplicado

Letter to predict with noise: 10

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

Iteration: 1

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

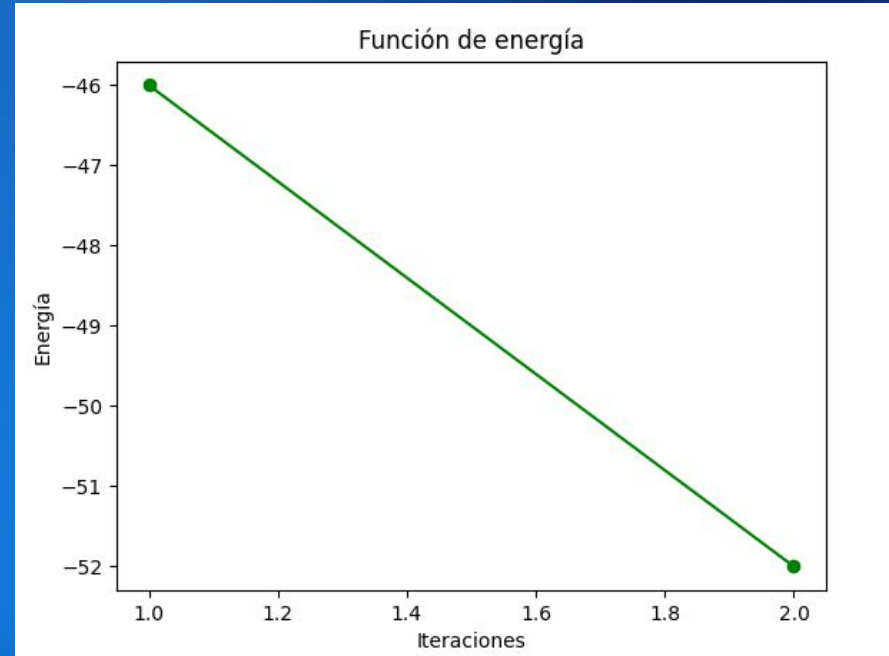
Iteration: 2

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

Letter predicted in 2 iteration/s

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

Estado espurio





# Letras fuera del patrón de entrenamiento

Letra “B” con ruido 0

```
Letter to predict:
* * * *
*      *
* * * *
* * * *
*      *
* * * *

Iteration: 1

* * * *
*      *
* * * *
*      *
*      *

Letter predicted in 1 iteration/s

* * * *
*      *
* * * *
*      *
*      *

Estado
incorrecto
```

Letra “M” con ruido 0

```
Letter to predict:
*      *
* *   *
* *   *
* *   *
*      *

Iteration: 1

* * * *
*      *
* *   *
*      *
* *   *

Iteration: 2

* * * *
*      *
* * * *
*      *
*      *

Iteration: 3

* * * *
*      *
* * * *
*      *
*      *

Letter predicted in 3 iteration/s

* * * *
*      *
* * * *
*      *
*      *

Estado
incorrecto
```

Letra “X” con ruido 0

```
Letter to predict:
*      *
*      *
*      *
*      *
*      *

Iteration: 1

*      *
*      *
*      *
*      *

Iteration: 2

*      *
*      *
*      *

Iteration: 3

*      *
*      *
*      *

Iteration: 4

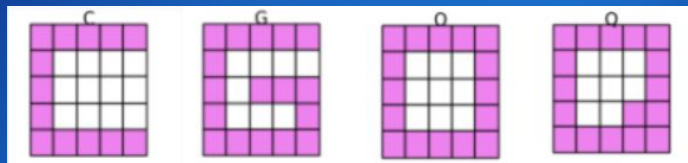
*      *
*      *
*      *

Letter predicted in 4 iteration/s

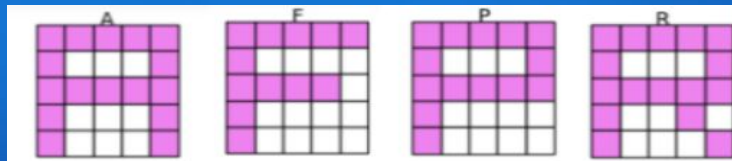
*      *
*      *
*      *
*      *

Estado
espurio
```

# Entrenamiento con los patrones no ortogonales



18.67



19.67

<,>  medio	grupo
------------	-------

18.67	('C', 'G', 'O', 'Q')
-------	----------------------

19.00	('C', 'D', 'O', 'Q')
-------	----------------------

19.00	('D', 'G', 'O', 'Q')
-------	----------------------

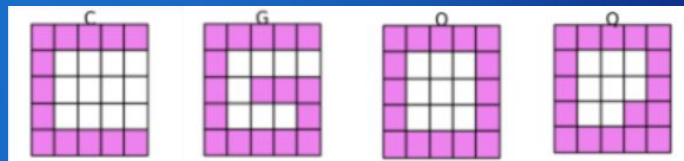
19.00	('D', 'O', 'Q', 'U')
-------	----------------------

19.67	('A', 'F', 'P', 'R')
-------	----------------------

# Analizando las “peores combinaciones”

$ \langle, \rangle $ medio	grupo
18.67	('C', 'G', 'O', 'Q')
19.00	('C', 'D', 'O', 'Q')
19.00	('D', 'G', 'O', 'Q')
19.00	('D', 'O', 'Q', 'U')
19.67	('A', 'F', 'P', 'R')

Patrón almacenado



```
Letter to predict:
* * * * *
*
*
*
*
* * * * *

Iteration: 1

* * * * *
*
*
*
*
* * * * *

Iteration: 2

* * * * *
*
*
*
*
* * * * *

Letter predicted in 2 iteration/s

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

Estado incorrecto

```
Letter to predict:
* * * * *
*
*
*
*
* * * * *

Iteration: 1

* * * * *
*
*
*
*
* * * * *

Iteration: 2

* * * * *
*
*
*
*
* * * * *

Letter predicted in 2 iteration/s

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

Estado incorrecto

```
Letter to predict:
* * * * *
*
*
*
*
* * * * *

Iteration: 1

* * * * *
*
*
*
*
* * * * *

Letter predicted in 1 iteration/s

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

```
Letter to predict:
* * * * *
*
*
*
*
* * * * *

Iteration: 1

* * * * *
*
*
*
*
* * * * *

Letter predicted in 1 iteration/s

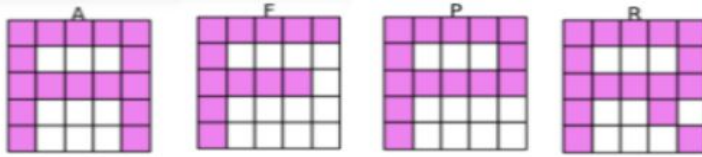
* * * * *
*
*
*
*
* * * * *
```

Estado incorrecto

|<>| medio grupo

18.67	('C', 'G', 'O', 'Q')
19.00	('C', 'D', 'O', 'Q')
19.00	('D', 'G', 'O', 'Q')
19.00	('D', 'O', 'Q', 'U')
19.67	('A', 'F', 'P', 'R')

## Patrón almacenado



Letter to predict:

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

Iteration: 1

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

Iteration: 2

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

Letter predicted in 2 iteration/s

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

Letter to predict:

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

Iteration: 1

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

Letter predicted in 1 iteration/s

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

Estado incorrecto

Letter to predict:

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

Iteration: 1

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

Letter predicted in 1 iteration/s

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

Letter to predict:

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

Iteration: 1

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

Iteration: 2

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

Letter predicted in 2 iteration/s

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

Estado incorrecto

Estado incorrecto

# ¿Cuál será mejor?

max: 3   count: 3	2.00	('R', 'U', 'V', 'Z')
max: 3   count: 4	2.33	('B', 'M', 'V', 'Z')
max: 3   count: 6	3.00	('F', 'M', 'V', 'Z')
max: 5   count: 1	2.33	('A', 'J', 'L', 'X')
max: 5   count: 1	2.00	('A', 'L', 'T', 'X')

Tenemos dos casos:

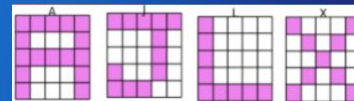
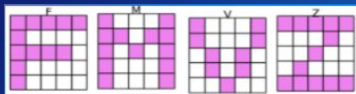
- Máximo 3 que aparece 6 veces con un promedio 3.0
- Máximo 5 que aparece una vez con promedio más bajo, 2.33

- Sin aplicar mutaciones a los patrones almacenados ("F", "M", "V", "Z" y "A", "J", "L", "X"), el algoritmo clasifica correctamente.

Veamos qué sucede cuando se aplican mutaciones...

max: 3   count: 6	3.00	('F', 'M', 'V', 'Z')
max: 5   count: 1	2.33	('A', 'J', 'L', 'X')

- Al aplicar ruido 8 podemos ver que el patrón almacenado “F”, “M”, “V”, “Z” da mejores resultados:



Letter to predict with noise: 8

```

* * *
* * *
* * *
*

```

Letter predicted in 3 iteration/s

```

*      *
*      *
* * *
*      *

```

Estado espurio

Letter to predict with noise: 8

```

*      *
*
* * *
*      *
*      *

```

Letter predicted in 2 iteration/s

```

*      *
* * * *
* * * *
* * * *
*      *

```

Letter to predict with noise: 8

```

* *
*
* * * *
*      *
*      *

```

Letter predicted in 3 iteration/s

```

* * *
* * * *
* * * *
* * * *
*      *

```

Estado espurio

Letter to predict with noise: 8

```

* * * *
*      *
*      *
* * * *
*

```

Letter predicted in 1 iteration/s

```

* * * *
*      *
*      *
*      *
*      *

```

Estado espurio

Letter to predict with noise: 8

```

*
*      *
*      *
* * *
* * *

```

Letter predicted in 2 iteration/s

```

*      *
*      *
*      *
*      *
*      *

```

Letter to predict with noise: 8

```

* * * *
* * * *
*      *
* * * *
* * * *

```

Letter predicted in 1 iteration/s

```

* * * *
*      *
*      *
*      *
* * * *

```

Letter to predict with noise: 8

```

* *
* *
* *
* *
*      *

```

Letter predicted in 2 iteration/s

```

*
*
*
*
* * * *

```

Letter to predict with noise: 8

```

*      *
*      *
*      *
* * *
* * *

```

Letter predicted in 4 iteration/s

```

*      *
*      *
*      *
*      *
*      *

```



# Entonces...

max: 3   count: 3	2.00	('R', 'U', 'V', 'Z')
max: 3   count: 4	2.33	('B', 'M', 'V', 'Z')
max: 3   count: 6	3.00	('F', 'M', 'V', 'Z')
max: 5   count: 1	2.33	('A', 'J', 'L', 'X')
max: 5   count: 1	2.00	('A', 'L', 'T', 'X')

- Como conclusión podemos ver que NO es conveniente tener números menores a 3 teniendo una más alta (5), a tener 6 tres.
- No hay que quedarnos solo con el promedio, porque en este caso uno hubiese elegido el menor promedio y no era lo mejor. Hay que tener en cuenta la distribución de los datos con los que se está trabajando.
- Hay que tener en cuenta que la media no es un estimador robusto.

# Conclusiones

- Cuando los patrones son más ortogonales, mejor será la asociación de los patrones.
- Para los patrones ortogonales, los mismos se resuelven bien hasta ruido = 8.
- Cuando usamos letras poco ortogonales no obtenemos buenos resultados.
- Al evaluar letras fuera del conjunto de patrones almacenados, se resuelve el patrón más parecido.
- Si el patrón a asociar tiene mucho ruido, es muy posible que se llegue a un estado espurio,

Fin

# Resultados letra "L" con ruido 10 aplicado

Letter to predict with noise: 10

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

```
* *  
* *  *
```

Iteration: 1

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

Iteration: 2

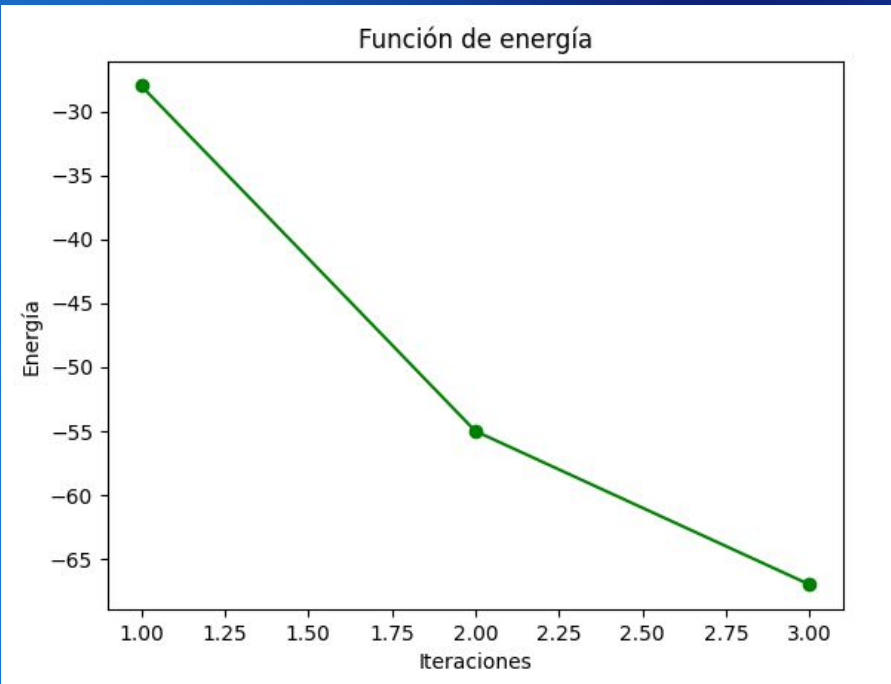
```
*  
  
*  
*  
* * * * *
```

Iteration: 3

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

Letter predicted in 3 iteration/s

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



# Resultados letra "V" con ruido 10 aplicado

Letter to predict with noise: 10

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

Iteration: 1

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

Iteration: 2

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

Iteration: 3

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

Letter predicted in 3 iteration/s

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

