# TP4

# MÉTODOS DE APRENDIZAJE NO SUPERVISADO

Juan Pablo Oriana - 60621
Tomás Cerdeira - 60051
Santiago Garcia Montagner - 60352

# 01

# Paises de Europa

# Paises de Europa

Se ofrece un csv con datos de 28 países de Europa y sus respectivas variables:

- Country
- Area
- GDP
- Inflation
- Life.expect
- Military
- Pop.growth
- Unemployment

# Conjunto de datos - CSV

```
"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
```

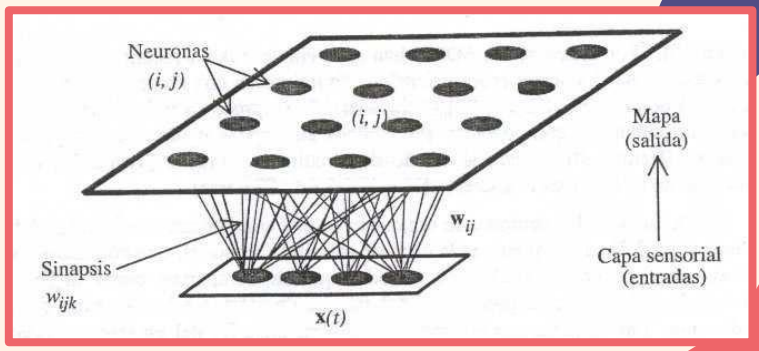
| Country | Area | GDP | Inflation | Life.expect | Military | Pop.growth | Unemployment |
|---|---|---|---|---|---|---|---|
| Austria | 83871 | 41600 | 3,5 | 79,91 | 0,8 | 0,03 | 4,2 |
| Belgium | 30528 | 37800 | 3,5 | 79,65 | 1,3 | 0,06 | 7,2 |
| Bulgaria | 110879 | 13800 | 4,2 | 73,84 | 2,6 | -0,8 | 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 |

# Red de Kohonen

Queremos **asociar países con cualidades sociopolíticas similares**, en base a los datos otorgados. Las redes de Kohonen son adecuadas para este tipo de problemáticas, ya que nos permiten **agrupar N entradas en un conjunto de clases de tamaño KxK**.

Tenemos control sobre los siguientes parámetros:

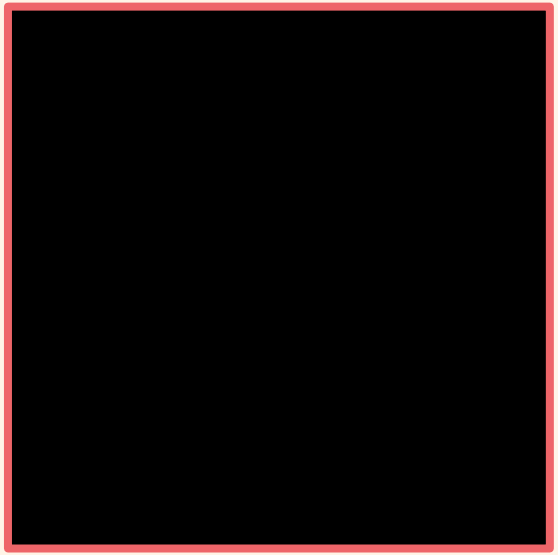
- K de salida
- El radio con el que se actualizan los pesos
- La tasa de aprendizaje y su evolución



# Self Organizing Map (SOM)

La **Red de Kohonen es un ejemplo de un SOM**. Los mapas autoorganizados permiten **preservar las propiedades topológicas** del espacio de entrada **disminuyendo su dimensionalidad**.

Mediante **aprendizaje competitivo**, se busca **agrupar los datos** que se introducen en la red; clasificándolos entre aquellos que comparten propiedades.



# Aprendizaje competitivo

Cuando un ejemplo entrenante es presentado a la red, su **distancia euclidiana a todos los vectores de pesos es calculada**.

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

$$W_{\hat{k}} = \arg \max_{1 \leq j \leq N} \{e^{-\|X^p - W_j\|^2}\}$$

La **neurona cuyo vector de pesos es más similar a la entrada es la neurona "ganadora"**. Los pesos de esta y sus neuronas vecinas en la cuadrícula del SOM son ajustados hacia el vector de entrada.

**Actualización de los pesos de las neuronas vecinas de $\hat{k}$**

utilizando la regla de de Kohonen:

- Si $j \in N_{\hat{k}}(t) \rightarrow W_j^{t+1} = W_j^t + \eta(t) * (X^p - W_j^t)$
- Si $j \notin N_{\hat{k}}(t) \rightarrow W_j^{t+1} = W_j^t$

# Estandarización

Antes de entrenar la red, **es necesario estandarizar el conjunto de datos**. Debemos primero calcular:

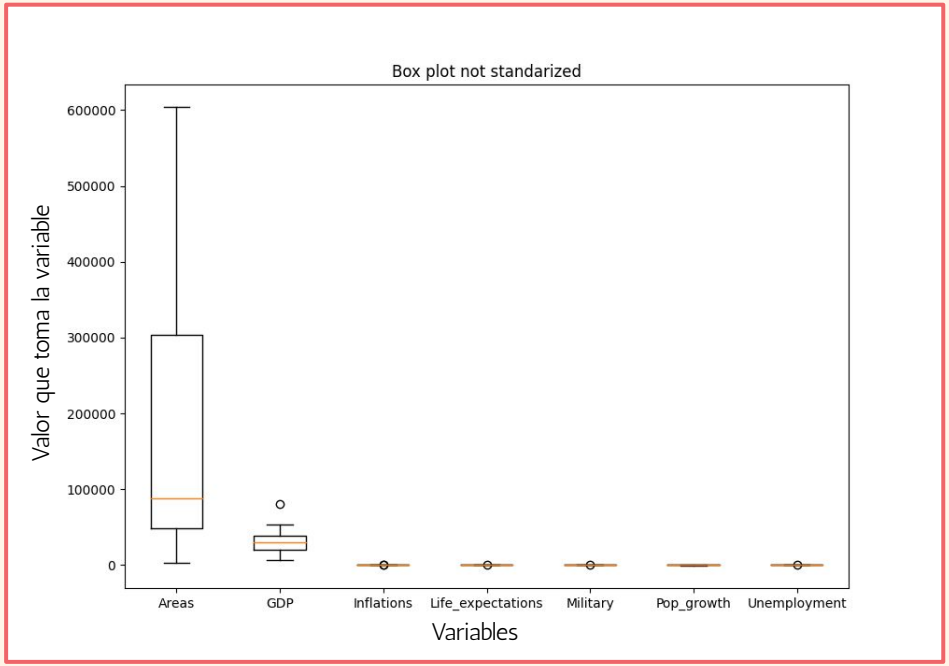
$$\bar{X}_i = \frac{1}{n}\sum_{j=1}^{n} X_i^j$$

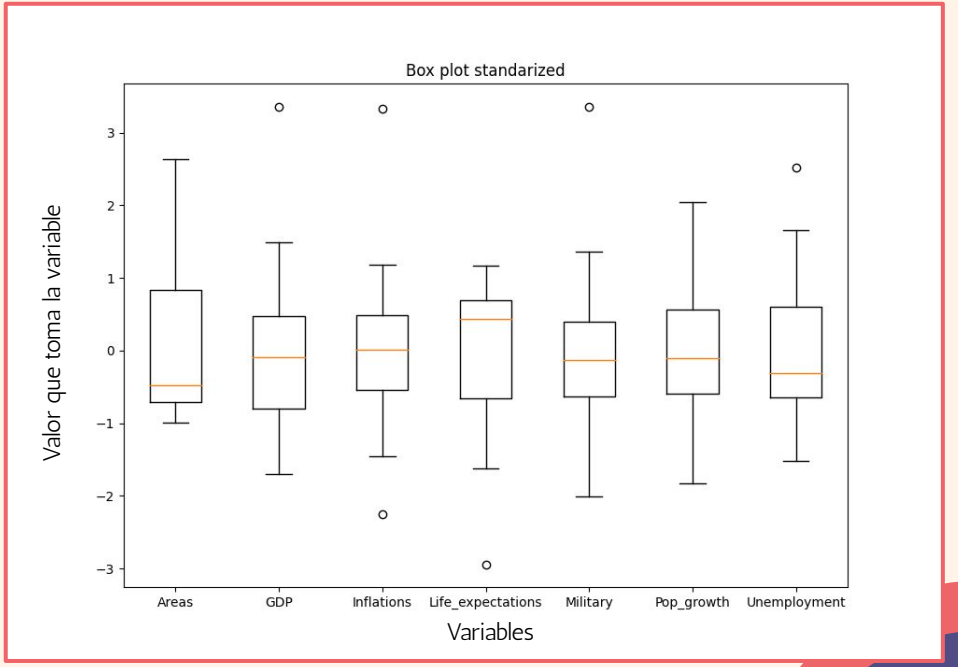
$$s_i = \frac{1}{n-1}\sum_{j=1}^{n}(X_i^j - \bar{X}_i)^2$$

Para luego calcular para cada una de las variables:

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

# Estandarización



# Implementación del KohonenSolver

- Valores de la red "default":
  - K: 3
  - Learning Rate: 0.2
  - Radio de vecinos: 1
  - Factor decrecimiento: 2
  - Epocas: 500 * 28
- Los pesos son inicializados con valores random del conjunto de entrenamiento.
- Un radio muy alto implica una convergencia muy rápida y sesgada.
- El learning rate arranca en 0.2 y se va cortando a la mitad iteración a iteración

# Resultados Parte 1.a

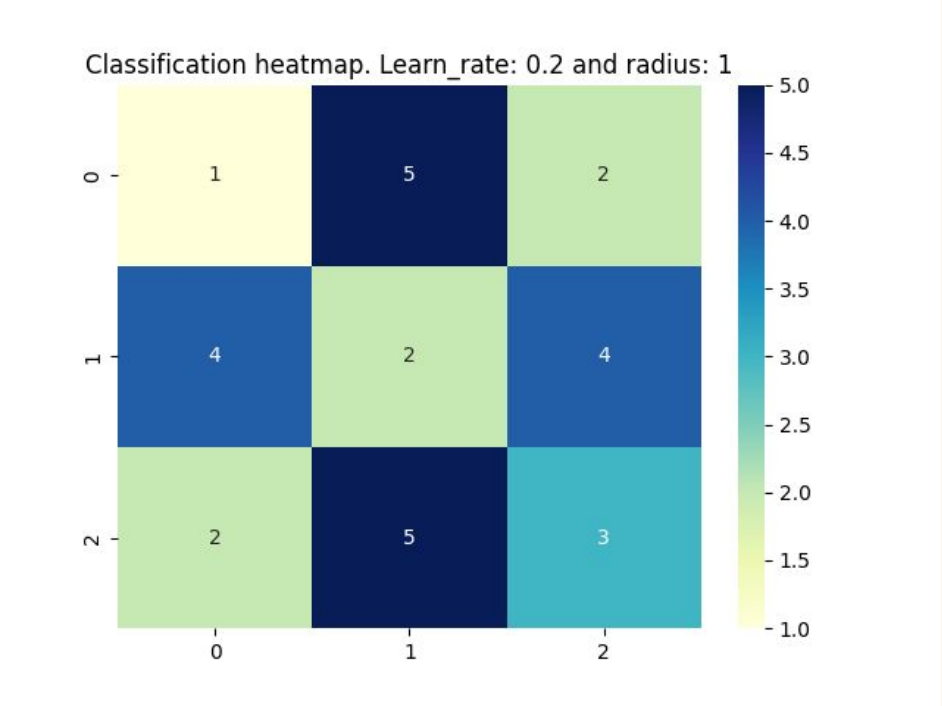
1) **Asociar países** que posean las mismas características geopolíticas, económicas y sociales.

2) Realizar al menos un **gráfico que muestre los resultados**.

3) Realizar un gráfico que muestre las **distancias promedio entre neuronas vecinas**.

4) **Analizar la cantidad de elementos** que fueron asociados a cada neurona.

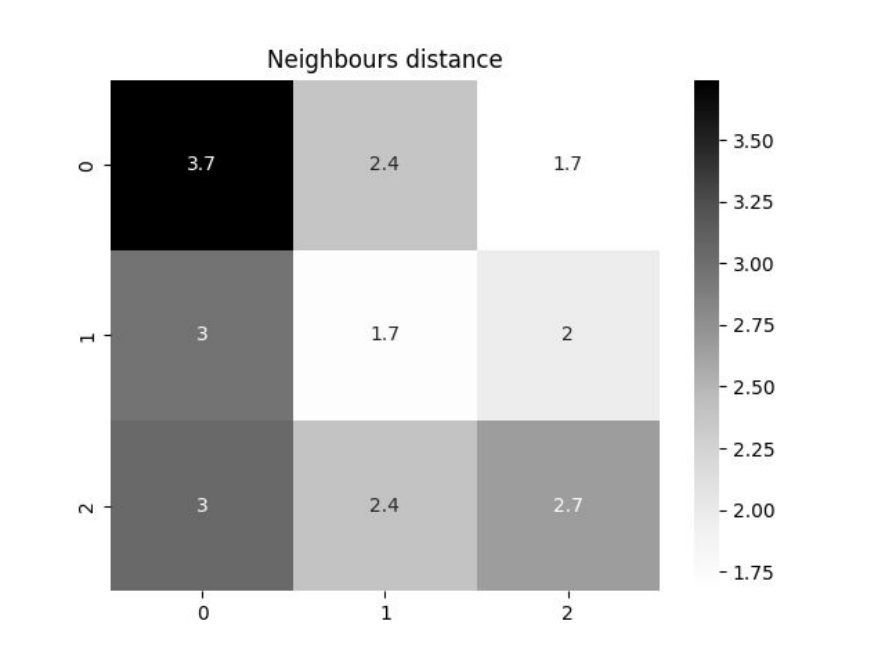
# Resultados obtenidos (1.a.1, 2 y 4)




| | | |
|---|---|---|
| Greece | Austria, Belgium, Czech Rep., Iceland, Slovenia | Latvia, Ukraine |
| Finland, Italy, Spain, United Kingdom | Croatia, Portugal | Bulgaria, Estonia, Hungary, Poland |
| Lithuania, Slovakia | Denmark, Ireland, Luxembourg, Netherlands, Switzerland | Germany, Norway, Sweden |

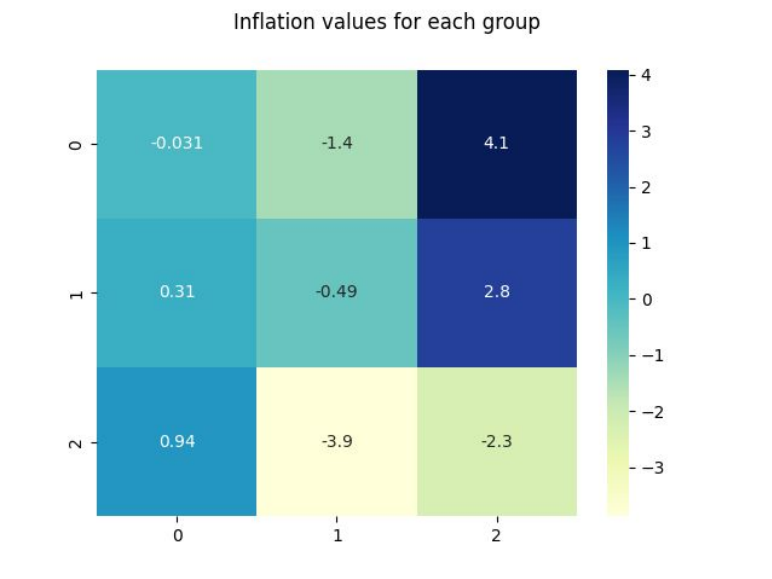
# Resultados obtenidos (1.a.2)

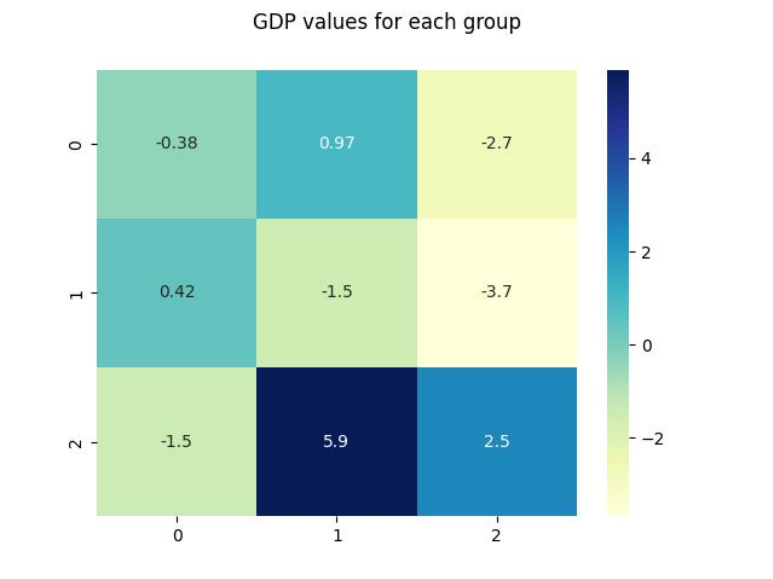
# Resultados obtenidos (1.a.3)




Radio:1
Learn rate: 0.2

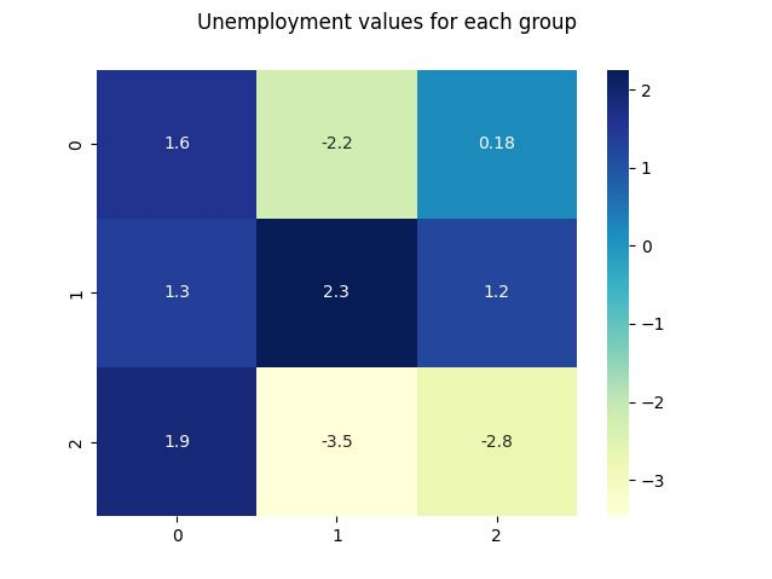
# Relación entre variables

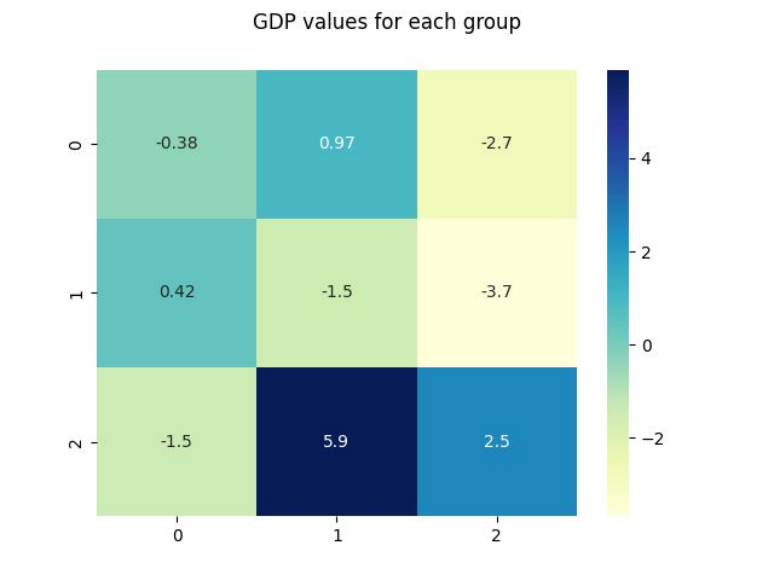



Se observa que, en general, países con **alta inflación**, tienen un **GDP negativo**,mientras que países con **baja inflación** tienen un **GDP mayor**. Es decir, que la inflación y el GDP tienen una correlación negativa

# Relación entre variables



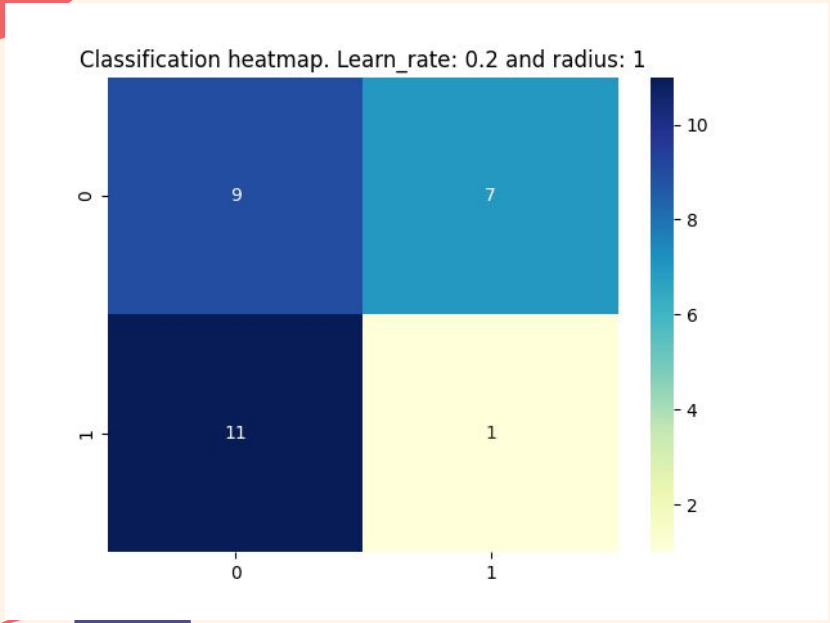

Se observa que, en general, países con **alto desempleo**, tienen un **GDP negativo**,mientras que países con **bajo desempleo** tienen un **GDP alto**. Es decir, que el desempleo y el GDP tienen una correlación negativa

# Elección del K

¿Por qué elegimos un **K=3**? Todo viene del hecho de que tenemos 28 países.

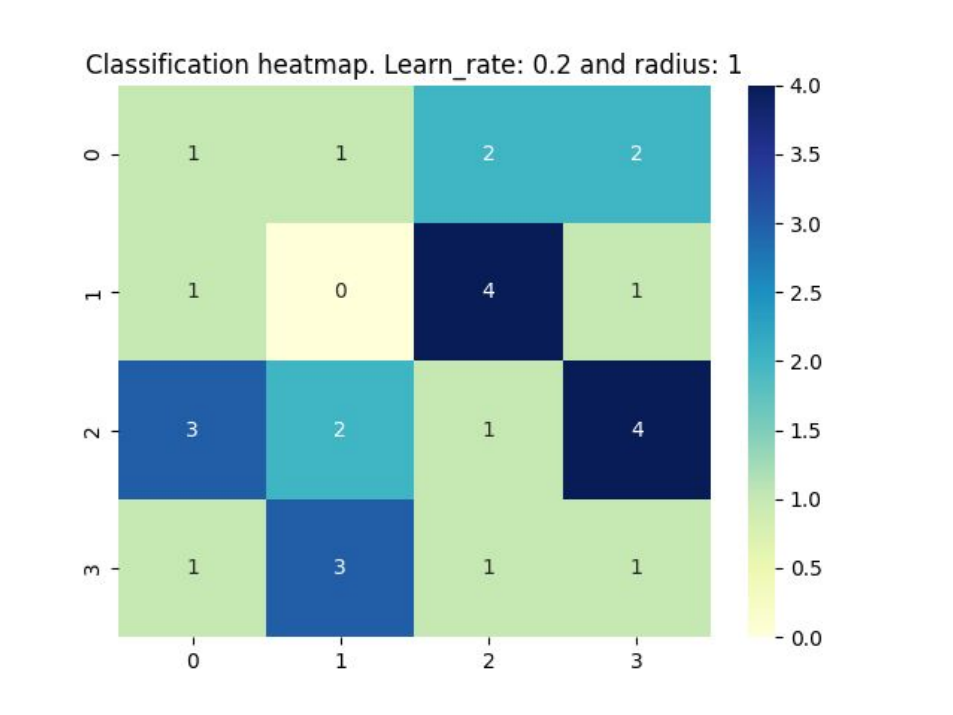
- **Si eligiéramos K=2** tendríamos solo 4 clases de datos, lo cual es muy poco para hacer algún tipo de inferencia en cuanto a sus relaciones
- **Si eligiéramos K=4** tendríamos solo 16 clases de datos, lo cual nos da un promedio de 1.75 países por clase. Esto es bastante poco y casi siempre resulta en neuronas muertas.

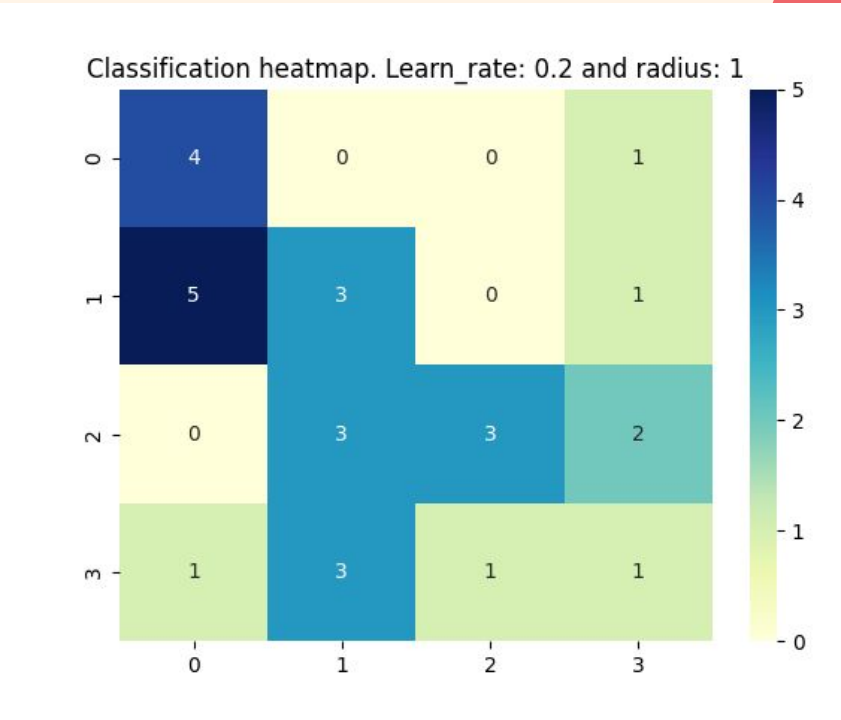
# Otros Resultados obtenidos




| | |
|---|---|
| Belgium, Croatia, Greece, Iceland, Ireland, Portugal, Slovakia, Spain, United Kingdom | Bulgaria, Estonia, Hungary, Latvia, Lithuania, Poland, Ukraine |
| Austria, Czech Republic, Denmark, Finland, Germany, Italy, Netherlands, Norway, Slovenia, Sweden, Switzerland. | Luxembourg |

# Otros resultados obtenidos



# Componentes Principales y regla de Oja

# Componentes principales

- Buscan **eliminar la redundancia de información** de un set 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 posible**.
    - De esta forma, construimos variables nuevas ponderadas en base a las originales

$$y_1 = \sum_{j=1}^{p} a_{1j}(x_j - \bar{x}_j) = a_{11}(x_1 - \bar{x}_1) + \ldots + a_{1p}(x_p - \bar{x}_p)$$

$y_1$: PCA-1, a: autovector/cargas, $x_i$: valor variable i

e.j. $$Y_1 = 0{,}50x_1 + 0{,}22x_2 + 0{,}35x_3 + 0{,}33x_4 + 0{,}48x_5 + 0{,}49x_6$$

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

# Algoritmo para el cálculo de las PCA

1. Tomar el conjunto de datos X y poner , si no están, las variables en columnas
2. Restar la media de cada conjunto de variables y obtener un conjunto de datos con media 0
3. Calcular matriz de covarianza
4. Calcular autovalores y autovectores de la matriz de covarianza. Ordenar los autovalores de mayor a menor
5. Formar las matriz E tomandos los autovectores de los mayores autovalores
6. Calcular las nuevas variables Y = ( X - X) E

**OBS**: Este cálculo es muy pesado computacionalmente

# Aprendizaje NO Supervisado

- Algunos modelos de redes neuronales permiten **calcular las componentes principales en forma iterativa**, ofreciendo ventajas computacionales.

- Utilizando un **perceptrón lineal simple con la regla de aprendizaje Hebbiano**, logramos reducir considerablemente el problema de cómputo que teníamos previamente.

**Actualizar los pesos**

$\Delta w_j = \eta y^n x_j^i$, $\eta$ es la tasa de aprendizaje.

$w_j^{n+1} = w_j^n + \eta y^n x_j^i$

**OBS**: el cálculo del delta w utiliza únicamente
η:tasa de aprendizaje
y: la excitación de la neurona
$x_j$: la entrada

# Oja

- El **Dr. Erkki Oja** demostró que, si la red anterior converge, **el vector de pesos final es un punto sobre la dirección de máxima variación de los datos** ⇒ La primer componente principal

- **Para que la función SIEMPRE converga**, Oja propuso que la actualización de los pesos sea:

$$\Delta w_j = \eta(y * x_j^n - y^2 * w_j^n)$$

Obtenido al derivar la división del peso de la componente n+1 por su norma

- Garantiza **convergencia al autovector** correspondiente al mayor autovalor de la matriz de correlaciones de los datos de entrada

⇒ **Método iterativo NO supervisado que garantiza convergencia a la 1era PCA**

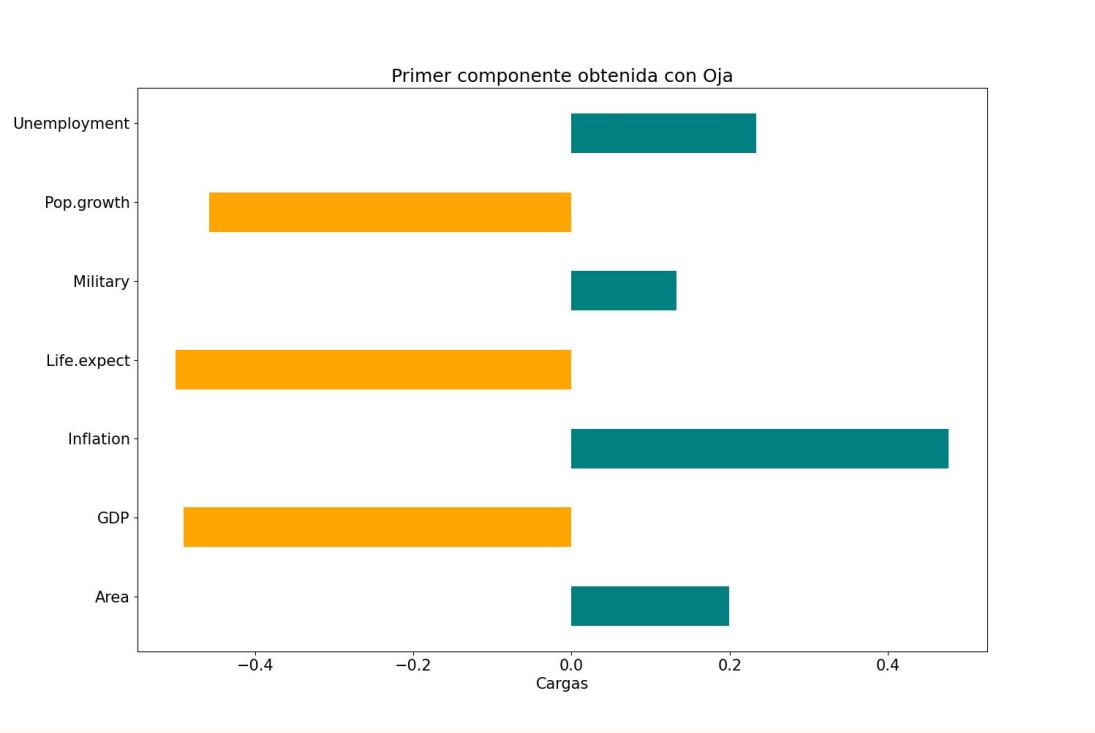
# Resultados Parte 1.b

1) **Calcular la primer componente principal** para este conjunto de datos.

2) **Interpretar el resultado** de la primer componente.

3) **Comparar el resultado** del ejercicio 1.b.1 con el resultado de calcular la primer componente principal **con una librera.**

# Resultados obtenidos (1.b.1)




**Autovector obtenido**: [ 0.19976932 -0.49045031 0.4767865 -0.50017386 0.13269894 -0.4576239 0.2335993 ]

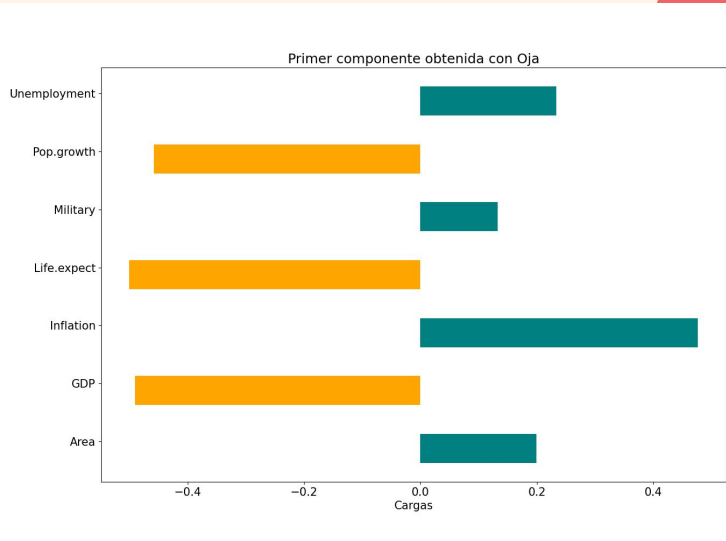
# Interpretación del resultado (1.b.2)

La primera componente me permite transformar las 7 variables poblacionales en un solo número que se caracteriza de la siguiente manera:

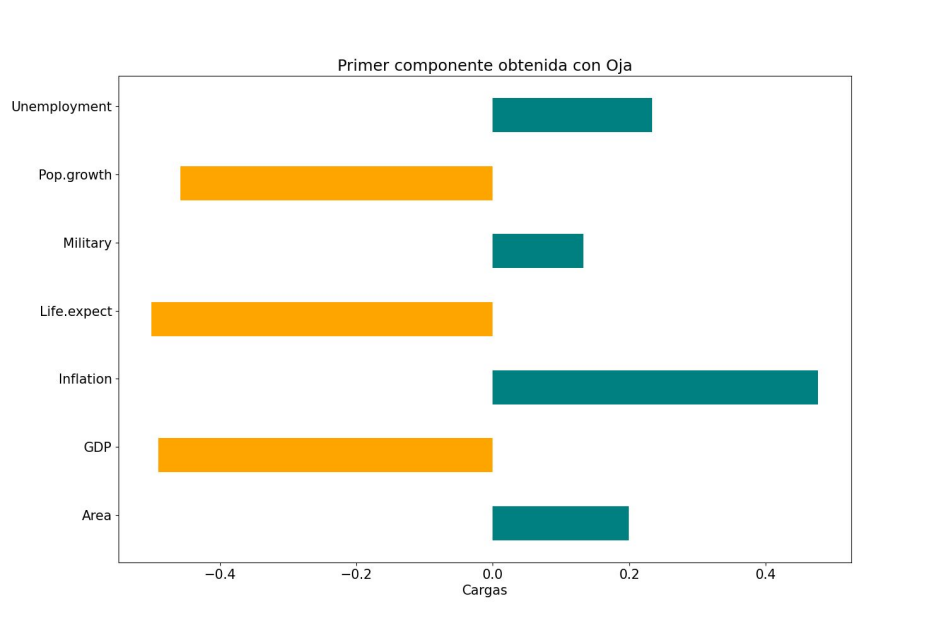
- Se ve influido positivamente por el desempleo, la inflación y, en menor medida, la presencia militar y el area del pais
- Se ve influido negativamente por el crecimiento poblacional, la expectativa de vida, y el PBI.

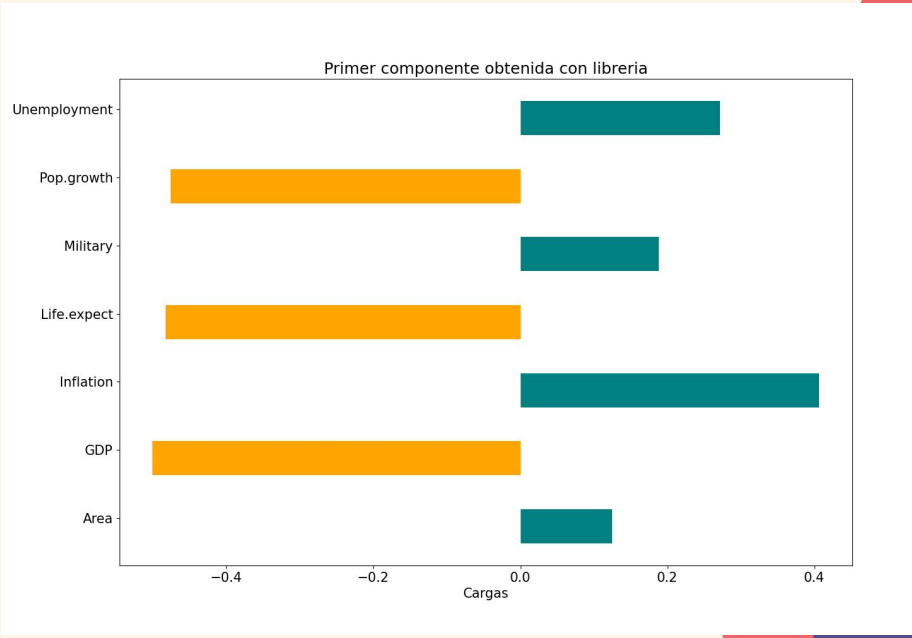
Pareceria ser que un pais con valores intensamente negativos tiene una frontera socioeconomica mas positiva.

Este nuevo valor conforma la varianza máxima en sus datos y es una buena forma de ordenar los países.



# Resultados obtenidos (1.b.3)




**Libreria**: *sklearn.decomposition import PCA*

Error: 0.78

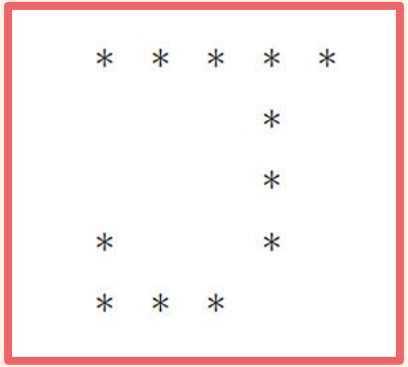
# 02

# Patrones numéricos

# Patrones numéricos

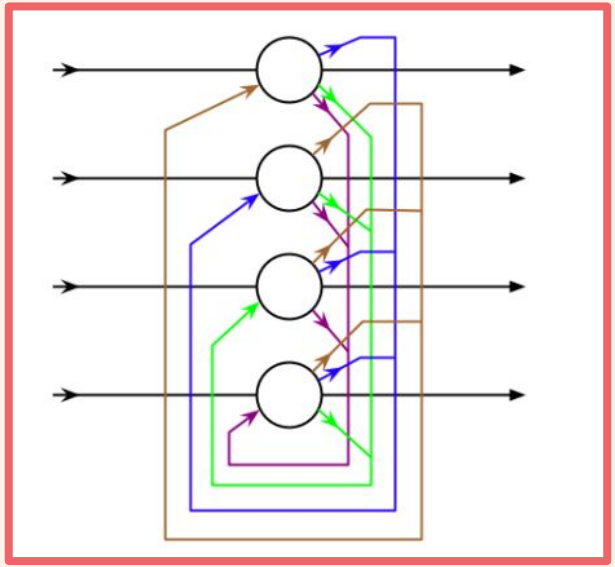
Tenemos patrones de números 5x5, representados con 1 y -1. La idea es almacenar 4 patrones de letras y poder asociar matrices ruidosas con los patrones almacenados.

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ -1 & -1 & -1 & 1 & -1 \\ -1 & -1 & -1 & 1 & -1 \\ 1 & -1 & -1 & 1 & -1 \\ 1 & 1 & 1 & -1 & -1 \end{pmatrix}$$



# Red de Hopfield

Las redes de Hopfield son un sistema de neuronas conectadas todas entre sí. Podemos entrenarlas con un conjunto de patrones que pueden ser almacenados en la red y luego reconocidos, con un nivel de eficacia que depende que se cumplan ciertos requisitos.



OBS:

$$\forall i, j,\ w_{ij} = w_{ji}$$
$$\forall i\ w_{ii} = 0,$$

# Estados y función de activación

Cada neurona tiene 2 estados posibles:

**$S_i$ es el estado de la neurona $i$**

- $S_i = +1$ ⇒ Activada
- $S_i = -1$ ⇒ Desactivada

La neurona $i$ modifica su estado $S_i$ de acuerdo a la regla:

$$S_i = \begin{cases} +1 & si\ h_i > 0 \\ -1 & si\ h_i < 0 \end{cases}$$

Si $h_i = 0$ entonces la neurona $i$ permanece en el estado previo.

$$S_i = sgn(h_i)$$

Siendo $h_i$, la función de activación de la neurona i:

$$h_i = \sum_{j=1}^{N} w_{ij} S_j,\ i \neq j$$

# Pesos Sinápticos (Vector de pesos w)

$$w_{ij} = \begin{cases} \frac{1}{N} \sum_{\mu=1}^{P} \xi_i^\mu \xi_j^\mu & j \neq i \\ 0 & j = i \end{cases}$$

donde

- $w_{ij}$ es el peso sináptico entre la neurona $i$ y la $j$.
- $\xi_i^\mu \in \{-1, 1\}$

Siendo ξ los patrones N-dimensionales de entrenamiento

# Inicializacion e Iteración hasta la convergencia

- Dado un vector de consulta ζ desconocido

$$S_i(0) = \zeta_i$$

⇒ El estado inicial del perceptrón es el vector de consulta ζ

- Se itera hasta la convergencia:

Actualizar los elementos del vector de estado $S(t)$ de acuerdo a la regla:

$$S_i(t+1) = sgn\left(\sum_{j=1}^{N} w_{ij} S_j(t)\right), \; j \neq i$$

Repetir la iteración hasta que el vector de estados $S$ permanezca estable.

# Estados espurios

- Hopfield demostró que **su red está asociada a una función de Energía**
  - Los **mínimos locales** de esta función **representan los patrones almacenados**

$$H = -\frac{1}{2}\sum_{i,j} w_{ij} S_i S_j$$

- Esta función H, **puede tener otros mínimos locales** que NO son los patrones almacenados ⇒ **Estados Espurios**

# Resultados Parte 2

a) **Almacenar 4 patrones de letras.** Realizar un programa que aplique el modelo de Hopfield para **asociar matrices ruidosas de 5 × 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**..

# Resultados obtenidos (2.a)

Patrón almacenado: **A G I L**

- Al no aplicar mutación a los patrones almacenados, el algoritmo es capaz de clasificar de manera correcta

Letra a predecir

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



Letra escogida luego de 1 iteración

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

# Resultados obtenidos (2.a)

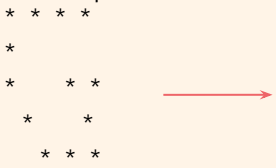
Patrón almacenado: **A G I L**

- Aplicando una **mutación de 0.1** observamos los siguientes resultados:

Letra a predecir:

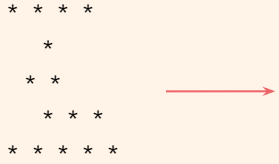
Letra predicha en 3 iteraciones:

Letra a predecir:

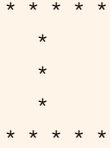


Letra predicha en 2 iteraciones:

Letra a predecir:

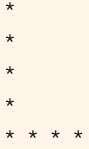


Letra predicha en 2 iteraciones:



Letra a predecir:

Letra predicha en 2 iteraciones:



**OBS**: En estos 4 casos, **la clasificación fue correcta** ya que se la letra que se predice es una mutación de la que se predijo

# Resultados obtenidos (2.a)

Patrón almacenado: **A G I L**

- Aplicando una **mutación de 0.2** observamos los siguientes resultados:

Letra a predecir:

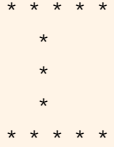
Letra predicha en 2 iteraciones:

Letra a predecir:

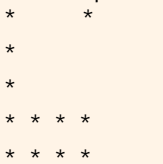
Letra predicha en 2 iteraciones:

Letra a predecir:

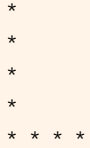
Letra predicha en 2 iteraciones:



Letra a predecir:



Letra predicha en 2 iteraciones:



**OBS**: En estos 4 casos, **la clasificación fue correcta** ya que se la letra que se predice es una mutación de la que se predijo

# Resultados obtenidos (2.a)

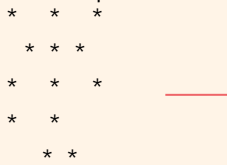
Patrón almacenado: **A G I L**

- Aplicando una **mutación de 0.4** observamos los siguientes resultados:

Letra a predecir:

Letra predicha en 2 iteraciones:

Letra a predecir:



Letra predicha en 6 iteraciones:

Letra a predecir:

Letra predicha en 4 iteraciones:

Letra a predecir:

Letra predicha en 7 iteraciones:

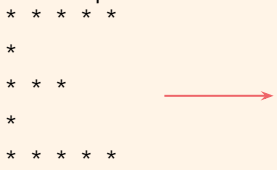
**OBS**: En estos 4 casos, **solo en 1 la clasificación fue correcta**. En otro caso, clasificó de manera incorrecta a otro grupo. Mientras que **en los últimos 2 los patrones cayeron en patrones espurios**

# Resultados obtenidos (2.a)

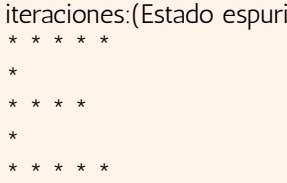
Patrón almacenado: **V S J E**

- Sin aplicar **ningún tipo de mutación**, el algoritmo no es capaz de identificar todas a las letras que tiene almacenadas
- Esto se debe a que algunos **patrones almacenados son muy similares** entre sí

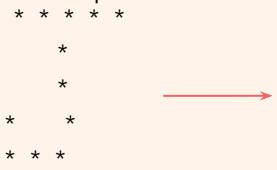
Letra a predecir:



Letra predicha en 2 iteraciones:(Estado espurio)

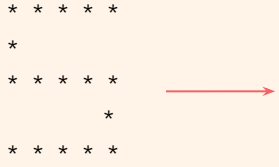


Letra a predecir:

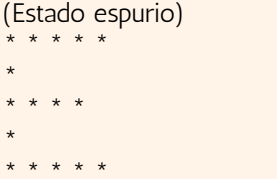


Letra predicha en 1 iteración:

Letra a predecir:

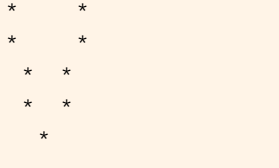


Letra predicha en 4 iteraciones: (Estado espurio)



Letra a predecir:

Letra predicha en 1 iteración:



**OBS**: La J y V son bien clasificadas, esto se debe a que son muy distintas a los demás patrones con los que se entrenó. En cambio, la S y la E van al mismo estado espurio, ya que son similares entre sí.

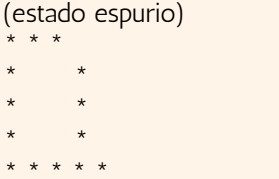
# Resultados obtenidos (2.b)

Patrón almacenado: **D O Q C**

- Sin aplicar **ningún tipo de mutación**, ya el algoritmo no es capaz de identificar a las letras que tiene almacenadas
- Esto se debe a que los **patrones almacenados son muy similares** entre sí

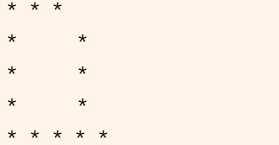
Letra a predecir:

Letra predicha en 2 iteraciones: (estado espurio)



Letra a predecir:

Letra predicha en 2 iteraciones: (estado espurio)



Letra a predecir:

Letra predicha en 4 iteraciones: (Estado espurio)

Letra a predecir:

Letra predicha en 7 iteraciones: (Estado espurio)

# Gracias por su atención.

# Relación entre variables