TP3: Métodos de Aprendizaje NO Supervisado

Introducción

- Aprendizaje no supervisado:
  - Red de Kohonen
  - Regla de Oja
  - Modelo de Hopfield discreto

Problema 1

- El conjunto de datos europe.csv corresponde a características económicas, sociales y geográficas de 28 países de Europa

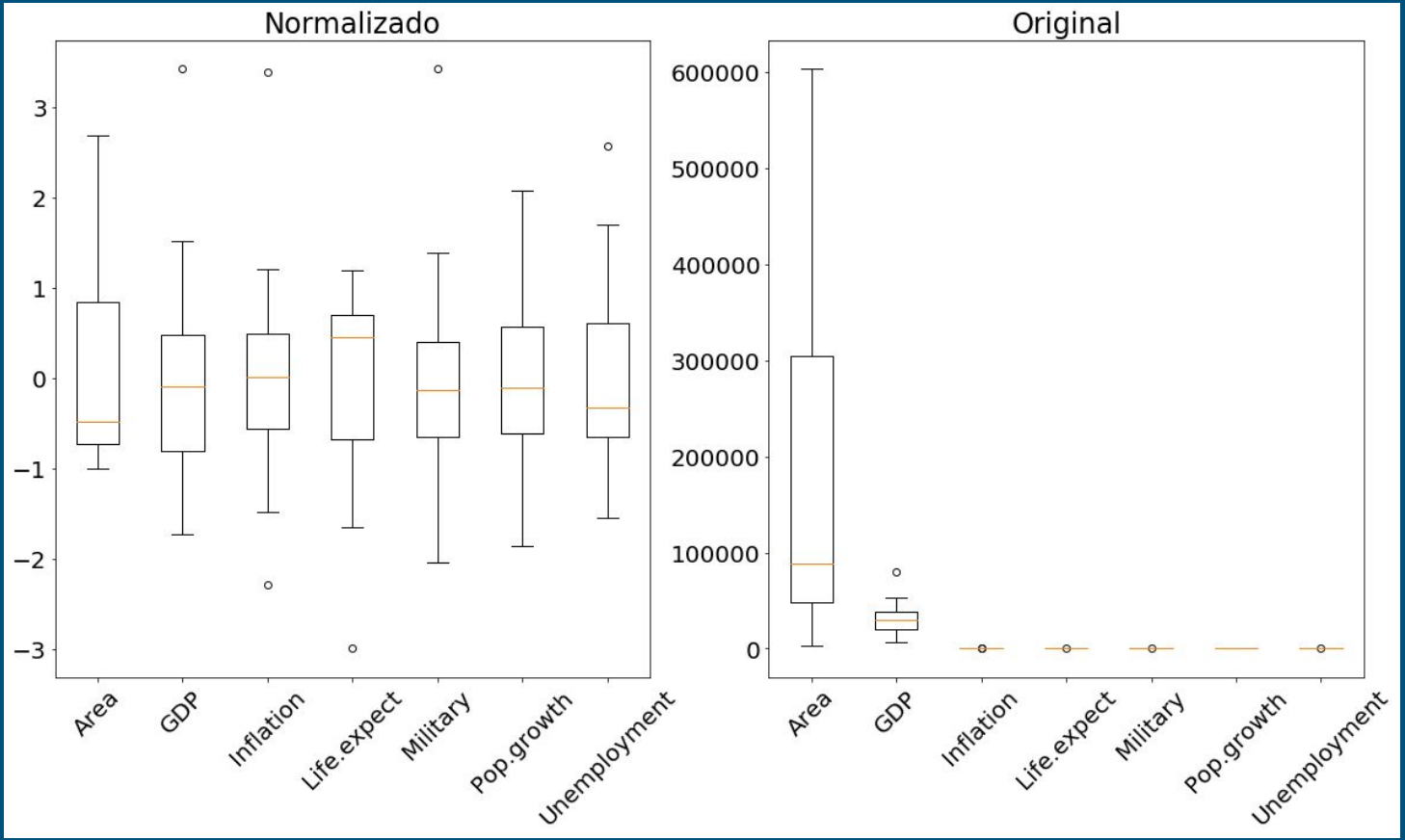
En base a 7 variables

1. Implementar la red de Kohonen
2. Implementar una red neuronal utilizando la regla de Oja

Implementación Kohonen

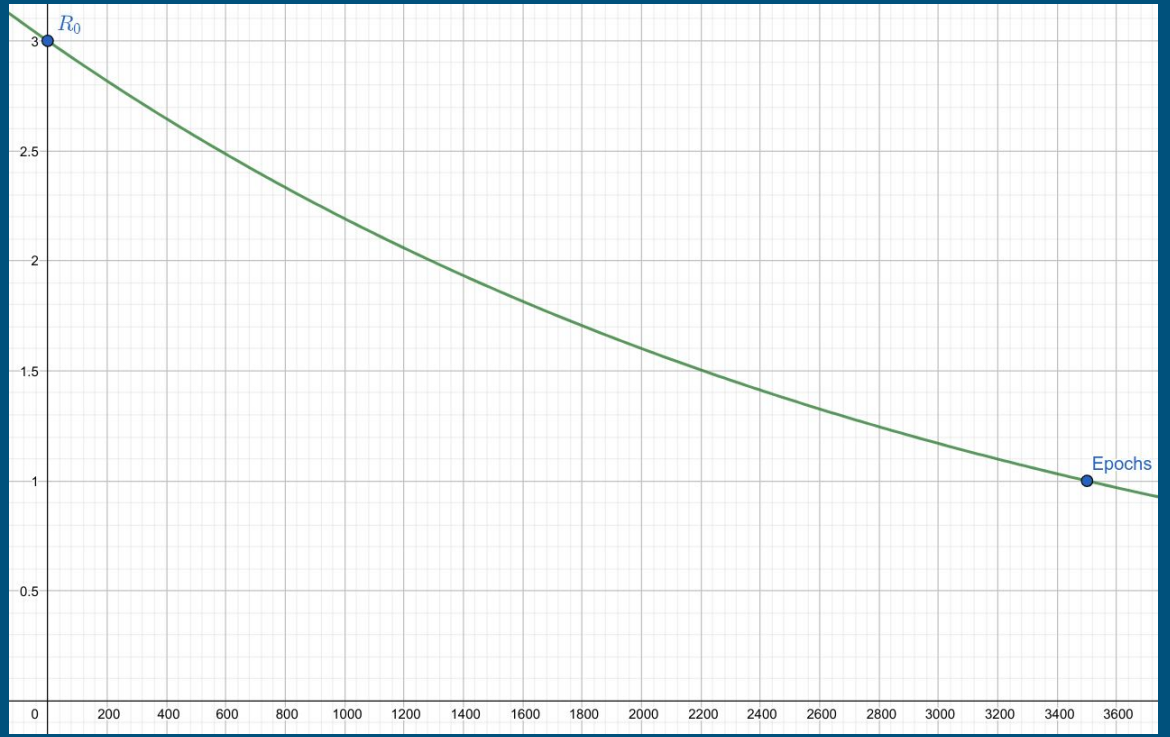
- Matriz de k*k, cada elemento tiene su vector W con 7 pesos
  - k < 6 (más neuronas que países → neuronas muertas)
- $R \to 1$ con $t \to \infty$
- $\eta \to 0$ con $t \to \infty$
- Cada iteración:
  - calculamos neurona ganadora
  - encontramos y actualizamos neuronas vecinas

Datos


Normalizado
3
2
1
0
−1
−2
−3
Area
GDP
Inflation
Life.expect
Military
Pop.growth
Unemployment
Original
600000
500000
400000
300000
200000
100000
0
Area
GDP
Inflation
Life.expect
Military
Pop.growth
Unemployment

Implementación Kohonen

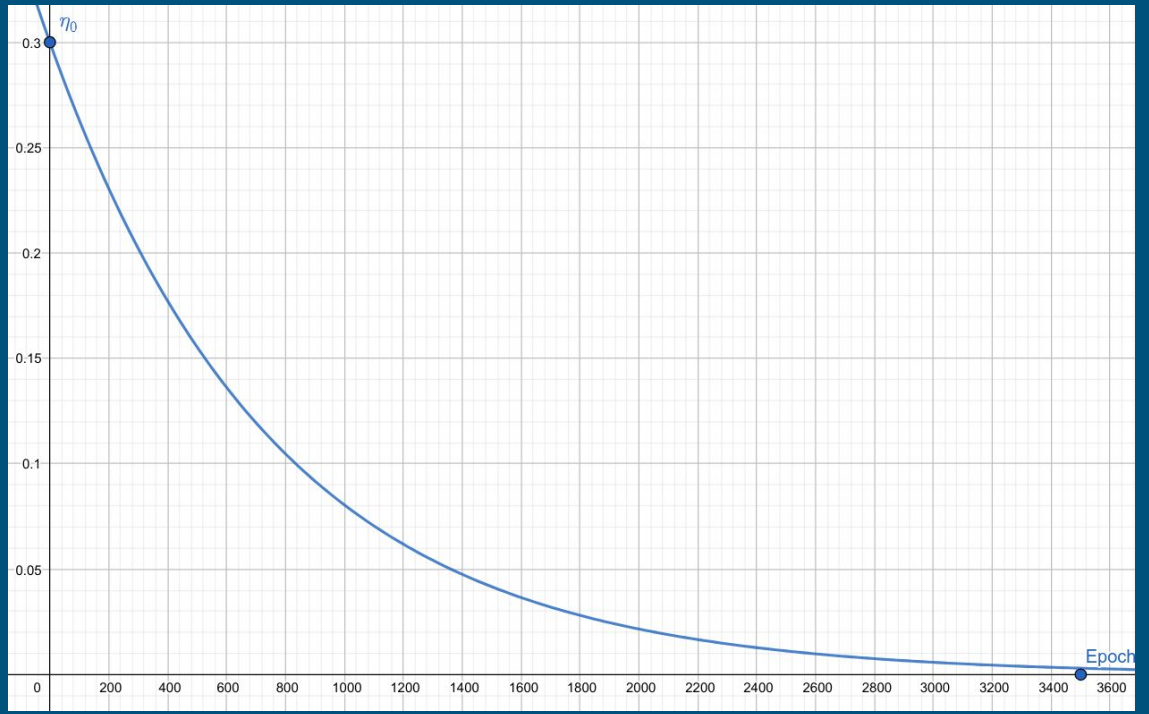
$$Radio(i) = R_0 * e^{\frac{-i*ln(R_0)}{epochs}}$$



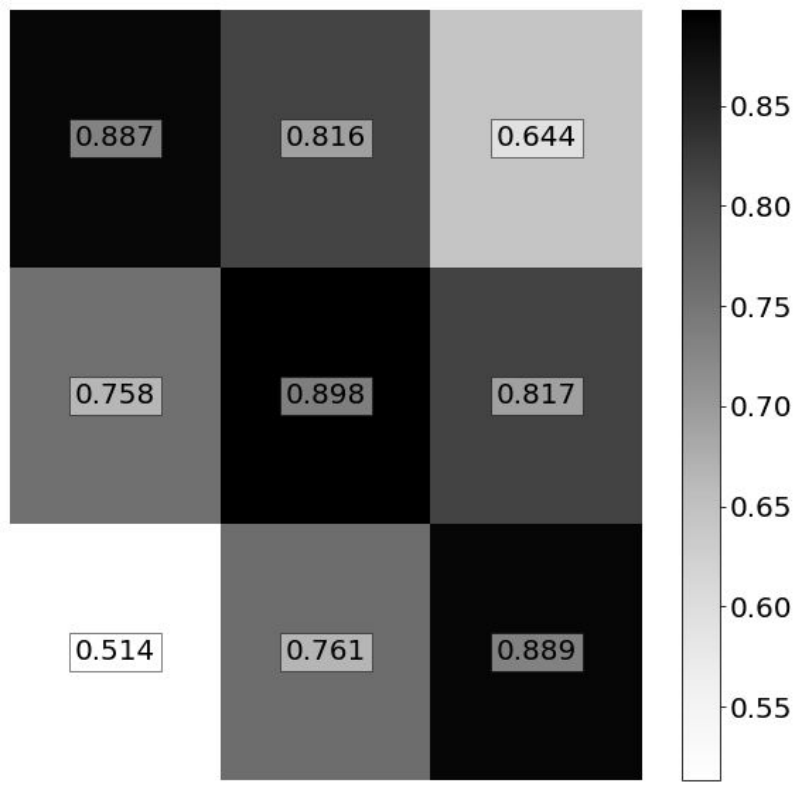
Implementación Kohonen

$$\eta(i) = \eta_0 * e^{\frac{i*ln(k)}{epochs}}$$

K = 1%



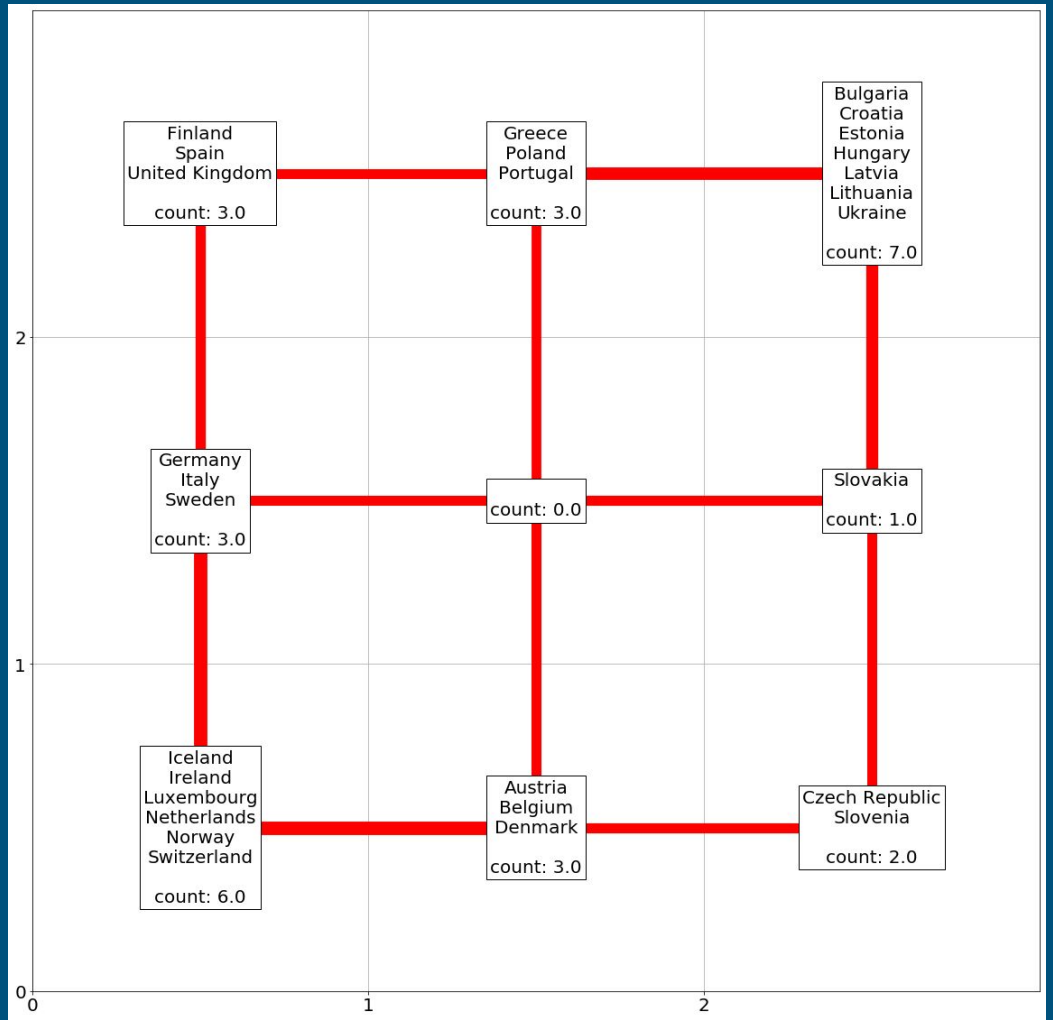
Matriz U distancia promedio entre neuronas



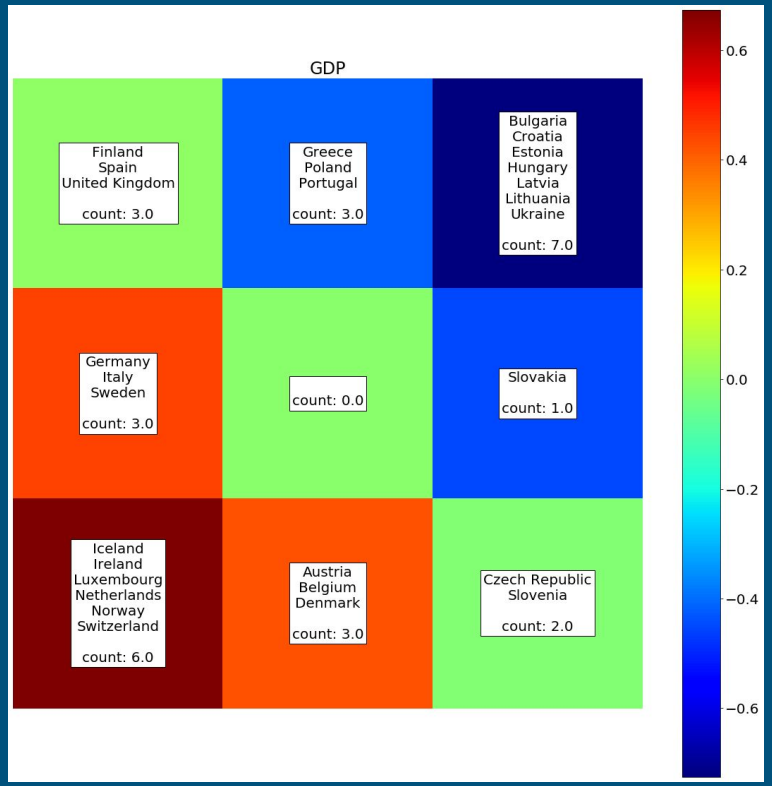
k= 3
R_0 = 3
η_0 = 0.01

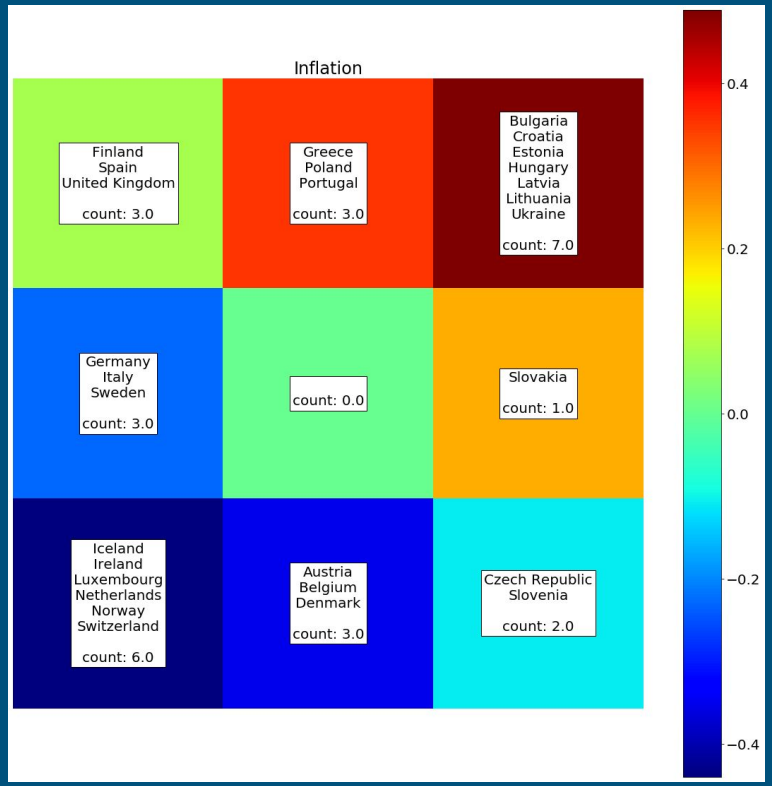
Elementos asociados a cada neurona

k= 3
R_0 = 3
η_0 = 0.01

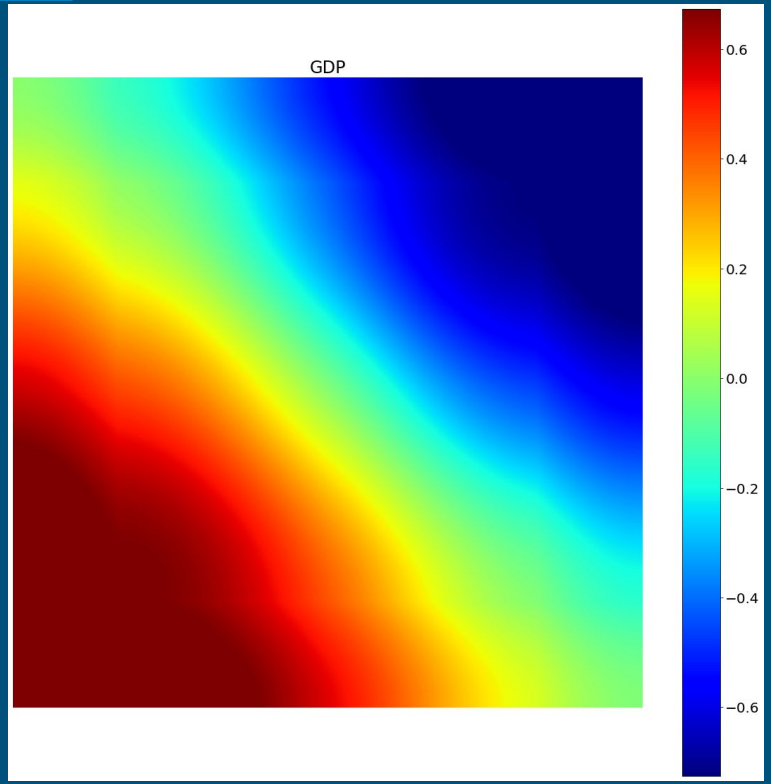


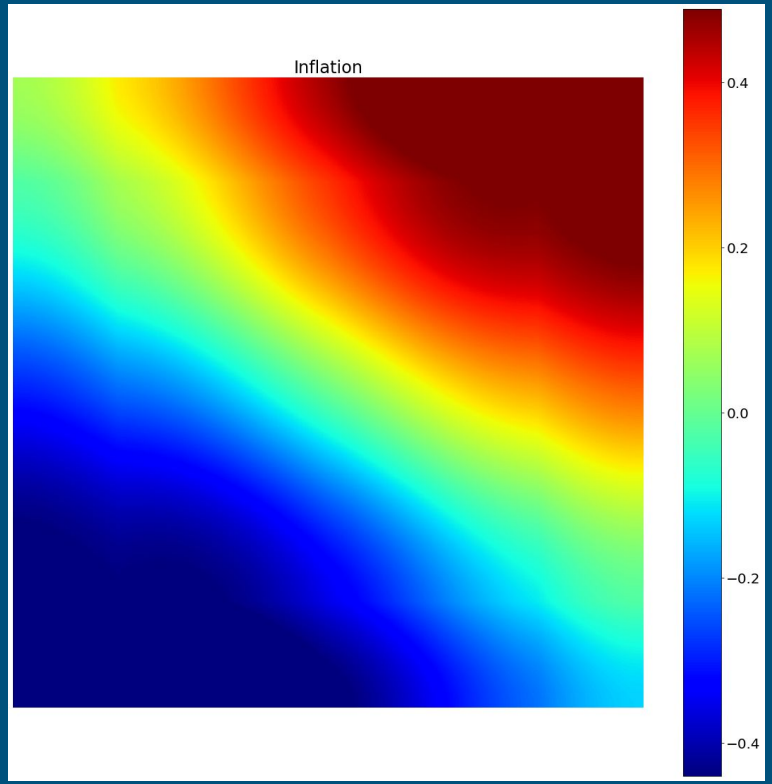
GDP vs Inflation (Componente)



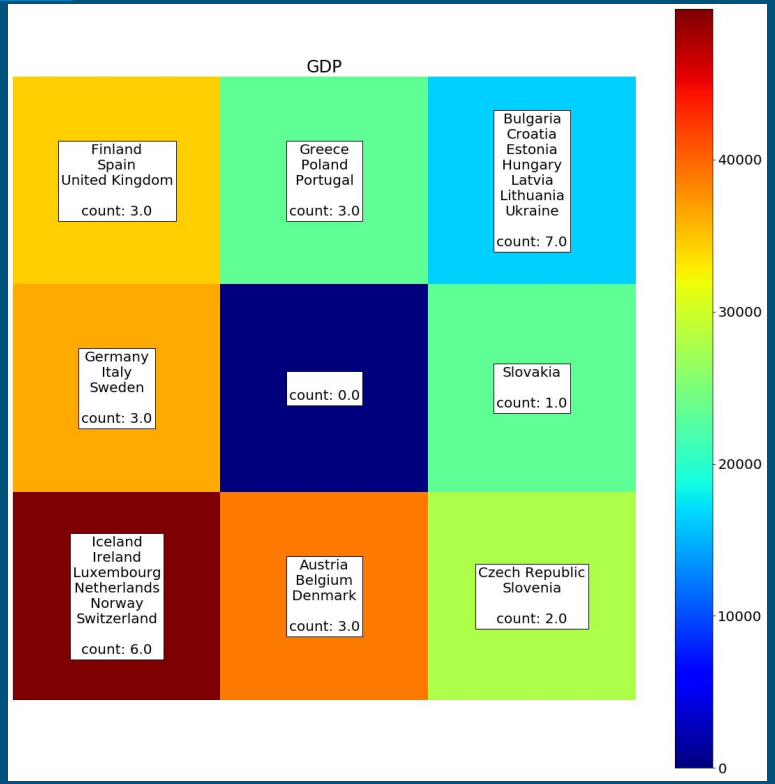


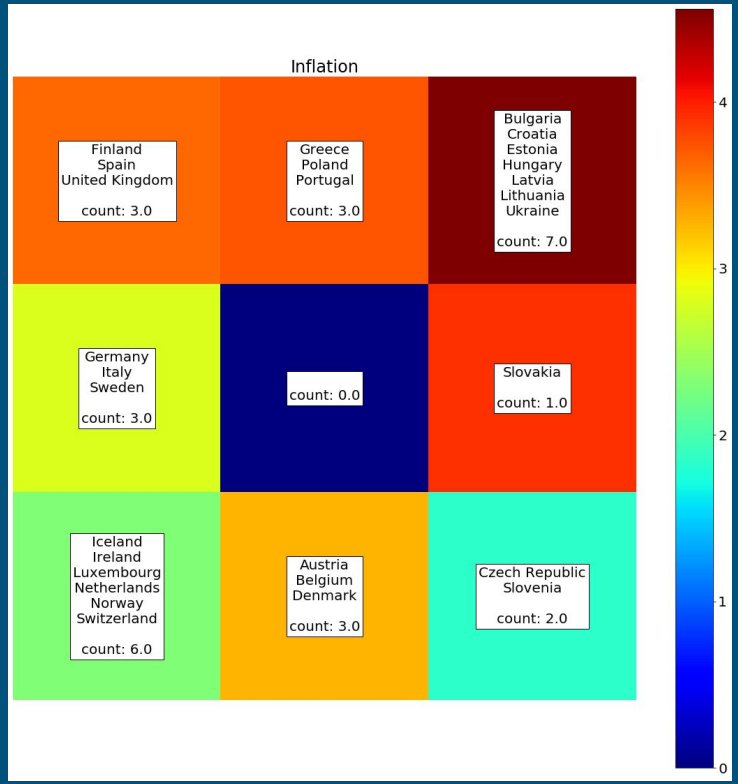
GDP vs Inflation (Componente)



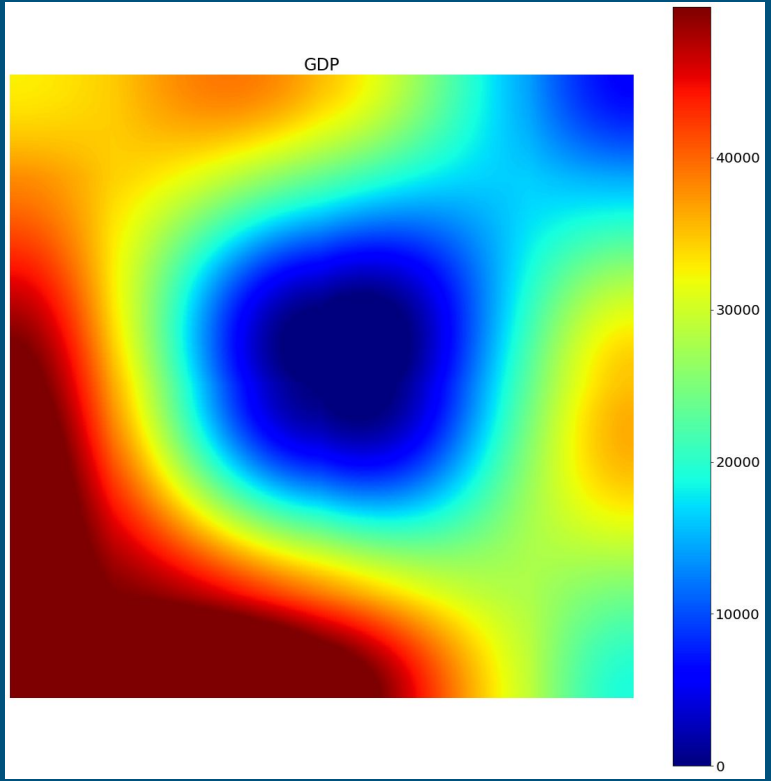


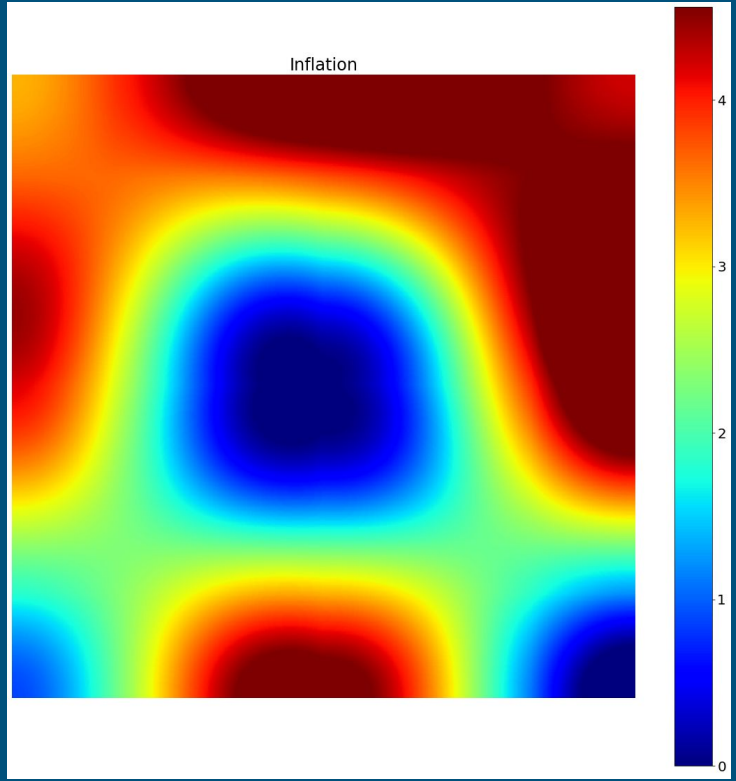
GDP vs Inflation (Promedio)



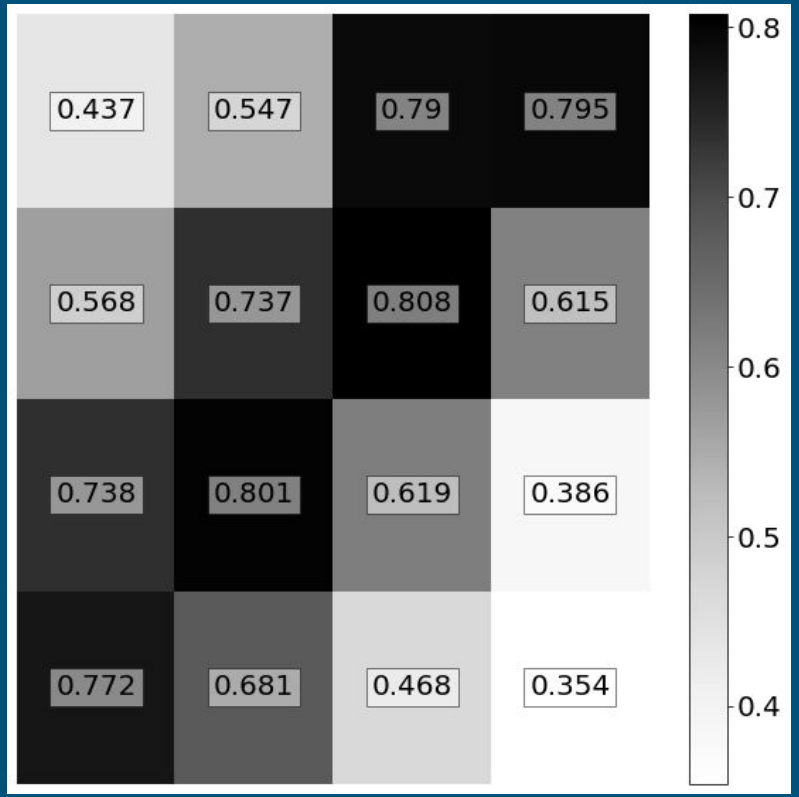


GDP vs Inflation (Promedio)





Matriz U distancia promedio entre neuronas



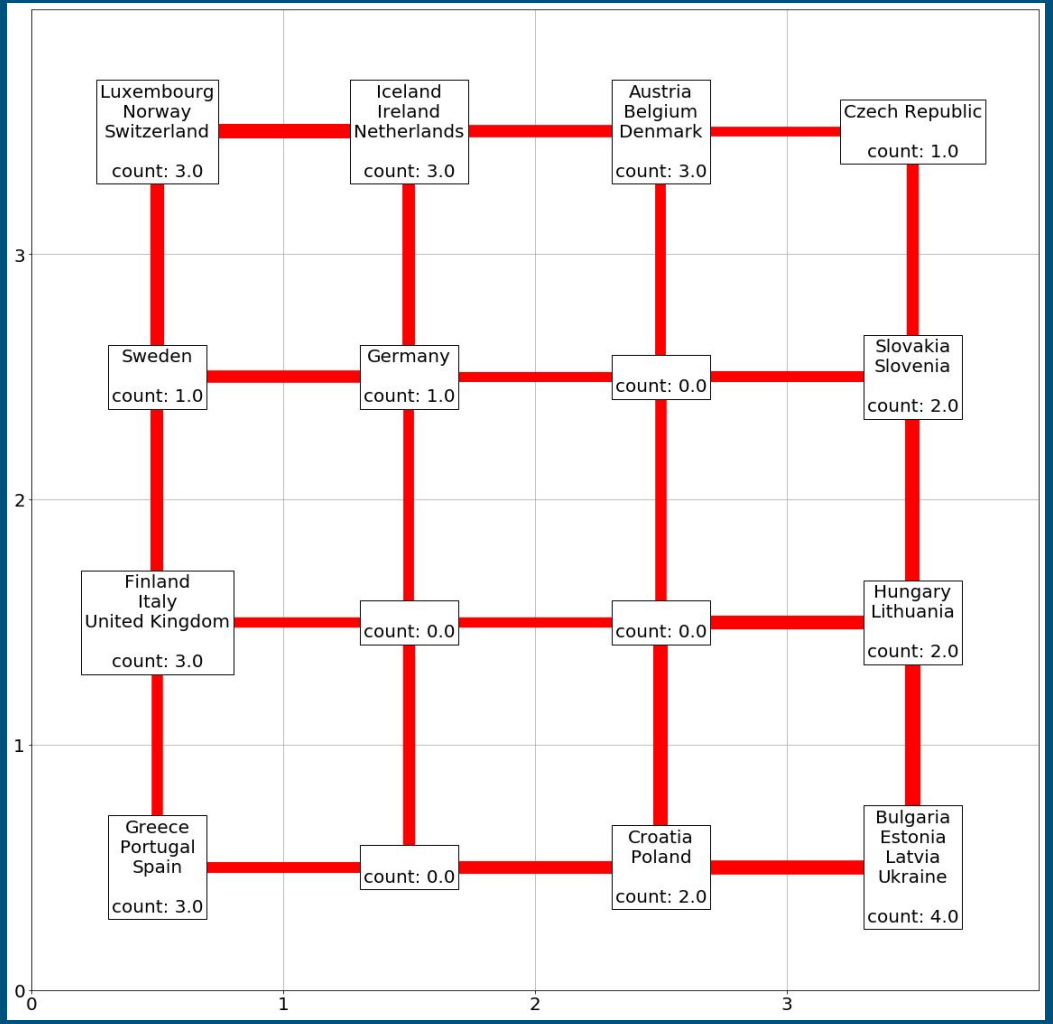
$k = 4$

$R_0 = 4$

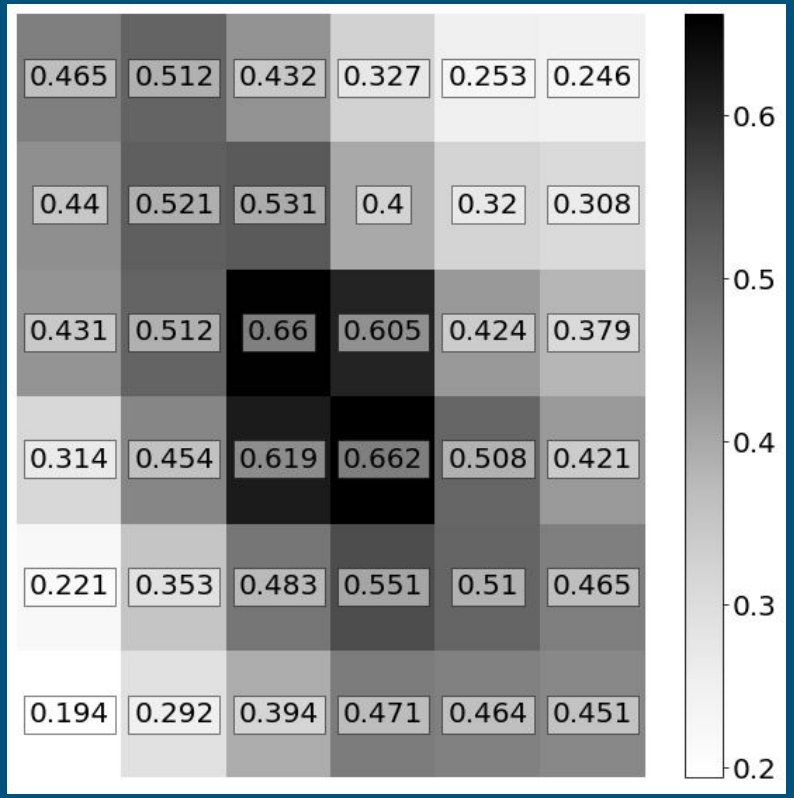
$\eta_0 = 0.01$

Elementos asociados a cada neurona

k= 4
R_0 = 4
η_0 = 0.01



Matriz U distancia promedio entre neuronas



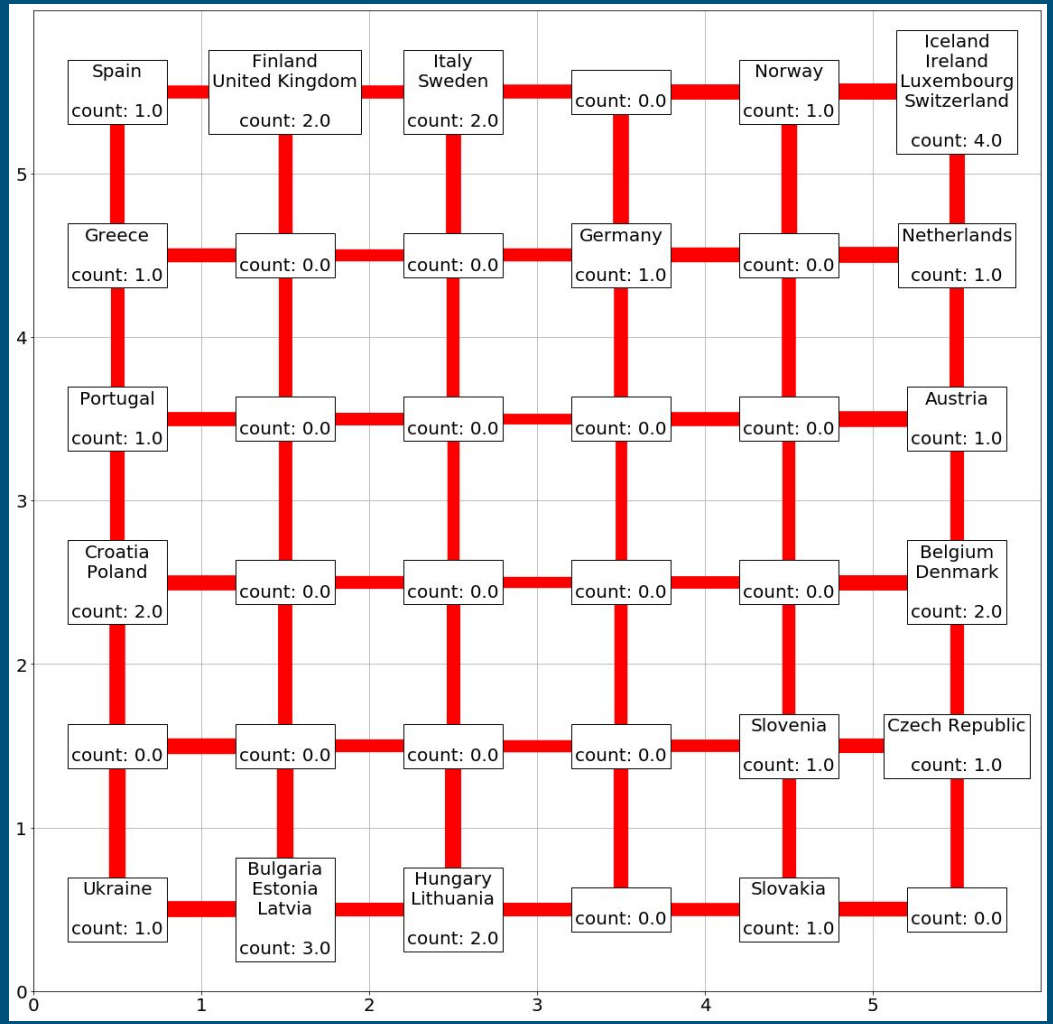
$k = 6$

$R_0 = 6$

$\eta_0 = 0.01$

Elementos asociados a cada neurona

k= 6
R_0 = 6
η_0 = 0.01



Conclusiones Kohonen

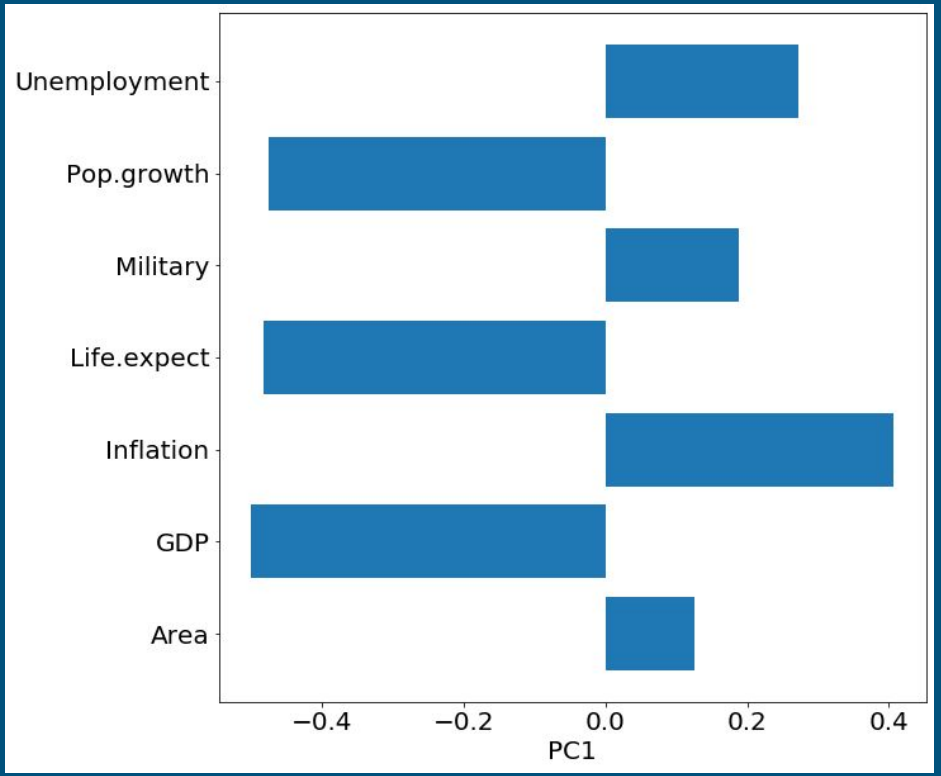
- Con más partículas que países hay neuronas muertas N=28, K<= 5
- Aún con neuronas muertas, si se tienen en cuenta los vecindarios, los grupos son coherentes con la distribución de la componente principal
- Usar un learning rate inicial cercano a 1, lleva a todas las neuronas cerca del último Xp de la época y por lo tanto la matriz U tiene distancias chicas pero no agrupa bien
- Distancias chicas no son indicador buena o mala agrupación
- El radio inicial debe ser “grande” para que no se formen grupos aislados
- No olvidar estandarizar los datos

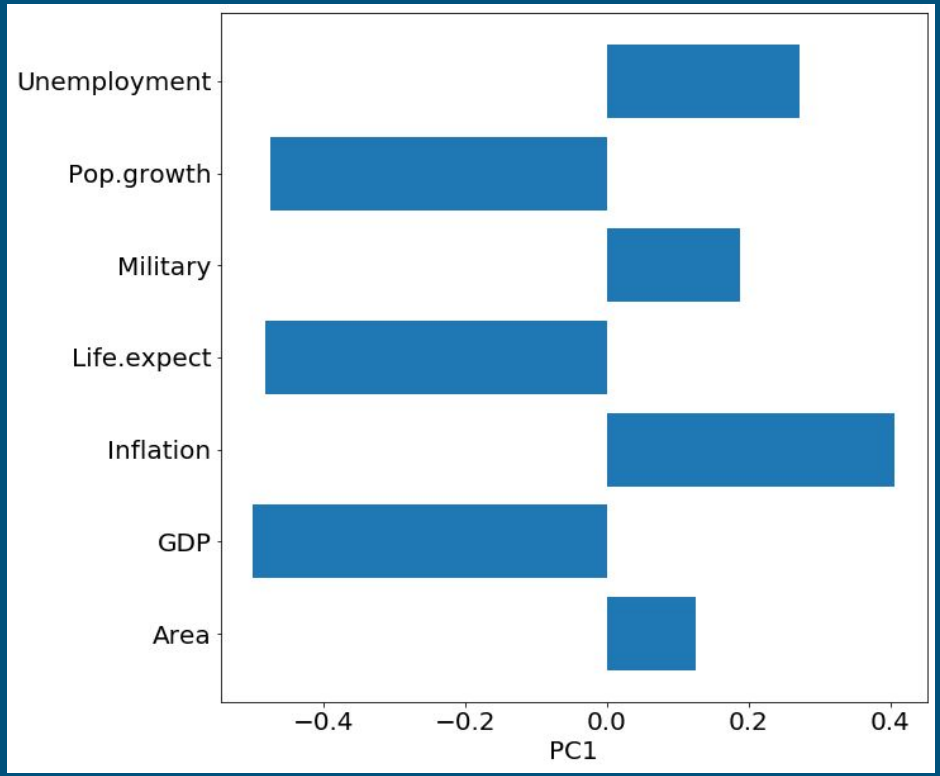
Implementación Oja

- Perceptrón simple
  - regla de actualización de Oja
- comparamos los resultados con el PCA utilizando la misma librería que usamos para el ejercicio obligatorio
  - sklearn

Resolución Oja VS PCA

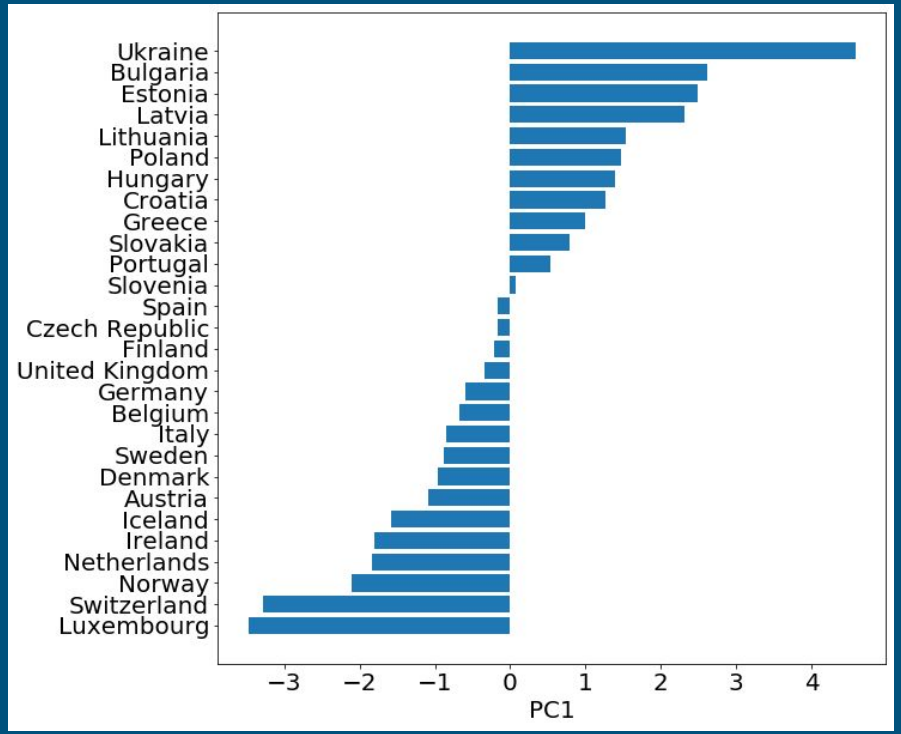
epochs=2000
lr =1e-4

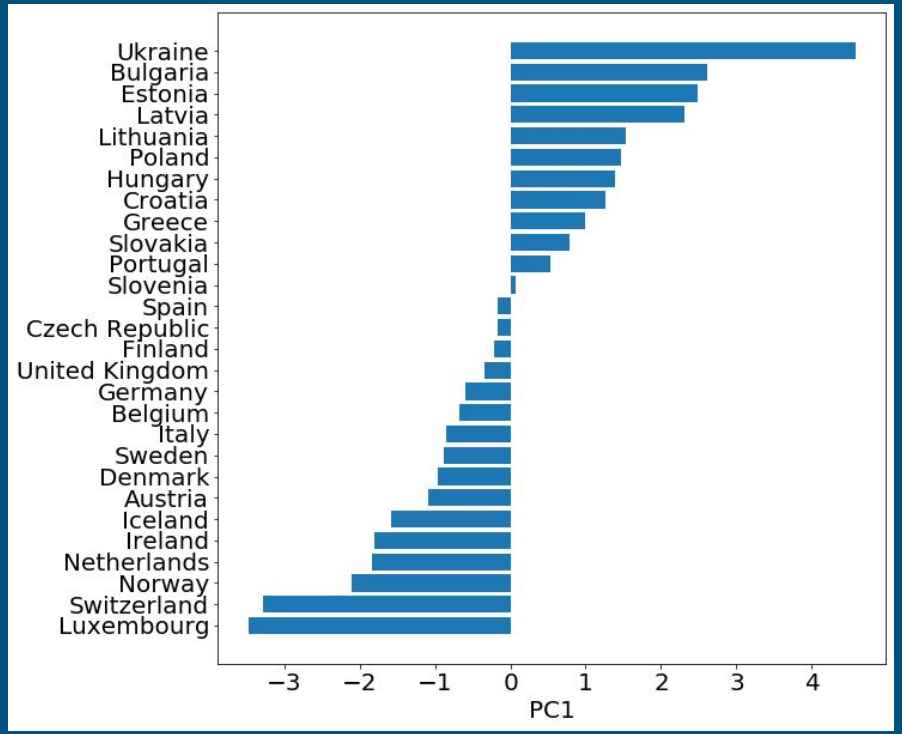




Resolución Oja VS PCA

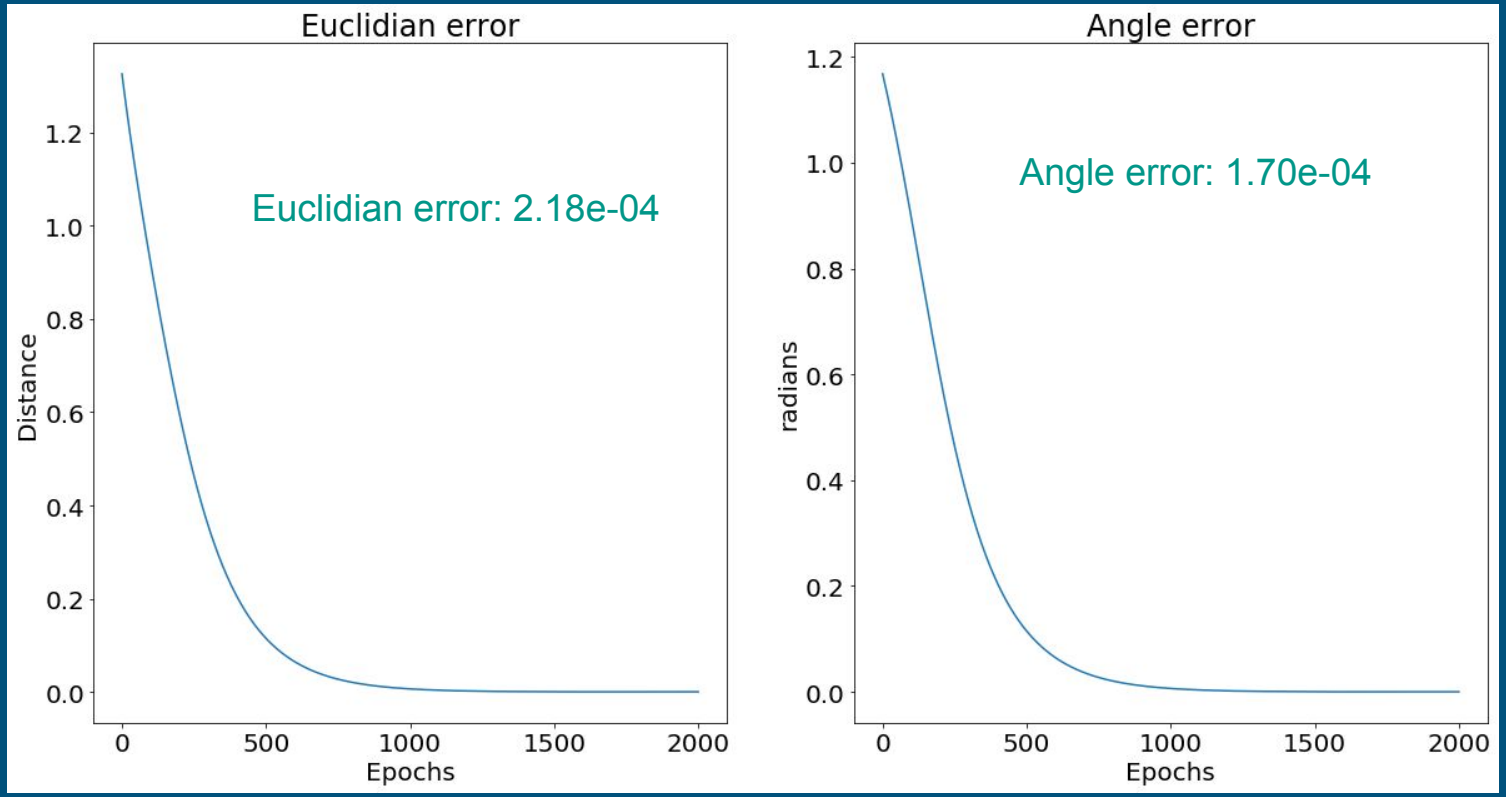
epochs=2000
lr =1e-4





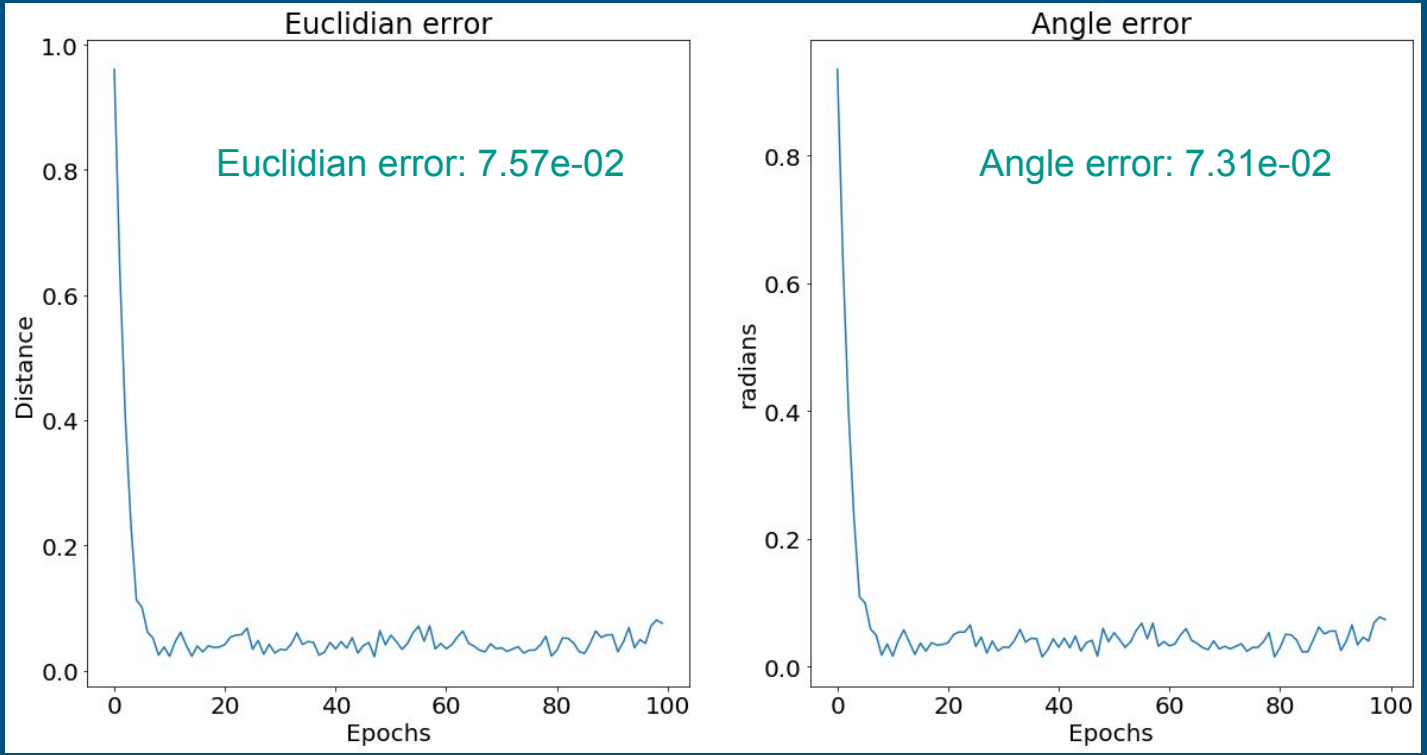
Resolución Oja VS PCA

epochs=2000
lr =1e-4



Resolución Oja VS PCA

epochs=100
lr =1e-2



Conclusiones Oja

- Tanto Oja como PCA llegan a estados parecidos
- Es inestable el método de Oja , entonces se debe colocar un learning rate muy chiquito
- Datos sin estandarizar no sirven de mucho en este método

Problema 2: Hopfield

- Almacenar 4 patrones de letras. Realizar un programa que aplique el modelo de Hopfield
  - Hacer alteraciones ruidosas de los patrones almacenados e ingresarlos al modelo, ver como progresan hasta su estado final
  - Ingresar un patrón muy ruidoso e identificar un estado espúreo.

Alfabeto utilizado

Implementación Hopfield

- Seleccionamos las 4 letras
  - [‘A’, ’L’, ’T’, ’X’]
  - Nos dan el patrón más ortogonal

Entrena con las letras seleccionadas y genera la matriz de pesos

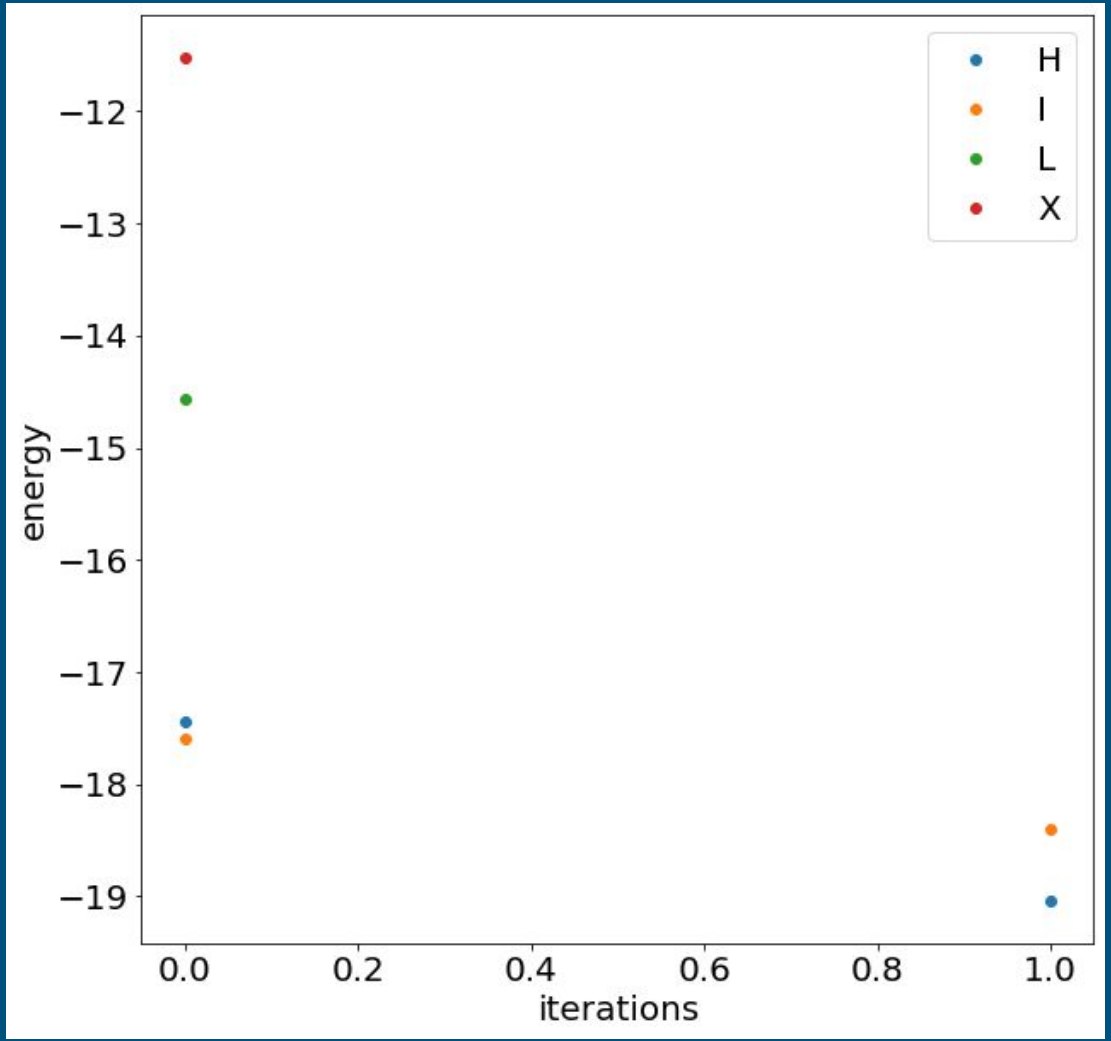
- Usamos el método predict para las letras con ruido y vemos el resultado

Elección del Patrón más ortogonal

- Calculamos el producto escalar normalizado entre cada par de letras
- Formamos N=26C4 grupos con todas las combinaciones de letras
- El score de cada grupo será el el producto escalar máximo entre cada par de letras
- Se selecciona el grupo con menor score

HILX

Menor producto escalar promedio

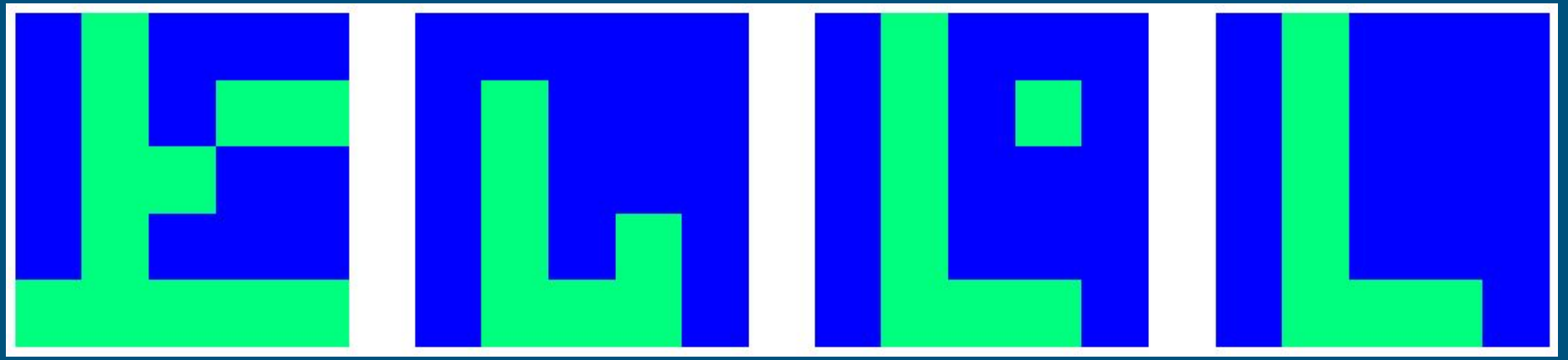


ALTX

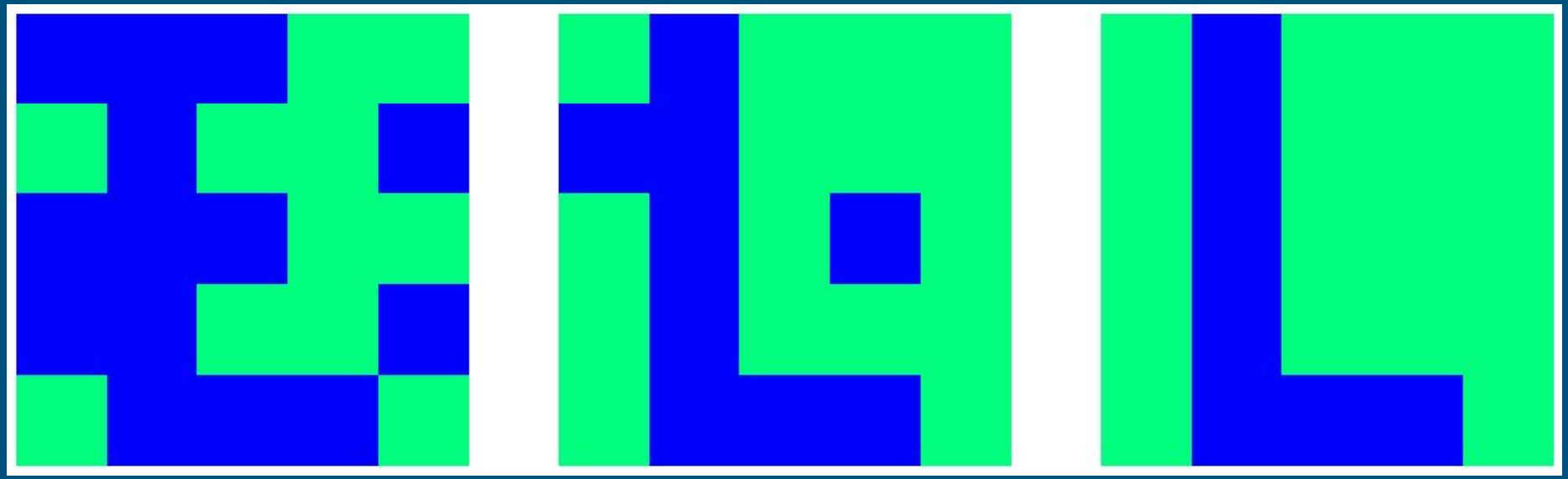
	Combinación	Max cos(θ)
0	[A, L, T, X]	0.36
1	[K, L, Q, X]	0.37
2	[K, L, Q, Y]	0.37
3	[L, T, U, X]	0.37
4	[L, Q, T, X]	0.38
5	[K, L, Q, T]	0.42
6	[K, L, T, X]	0.42
7	[K, Q, T, V]	0.42
8	[K, Q, T, X]	0.42
9	[D, K, M, V]	0.44

	A	L	T	X
A	1.00	0.20	0.36	0.27
L	0.20	1.00	0.25	0.25
T	0.36	0.25	1.00	0.33
X	0.27	0.25	0.33	1.00

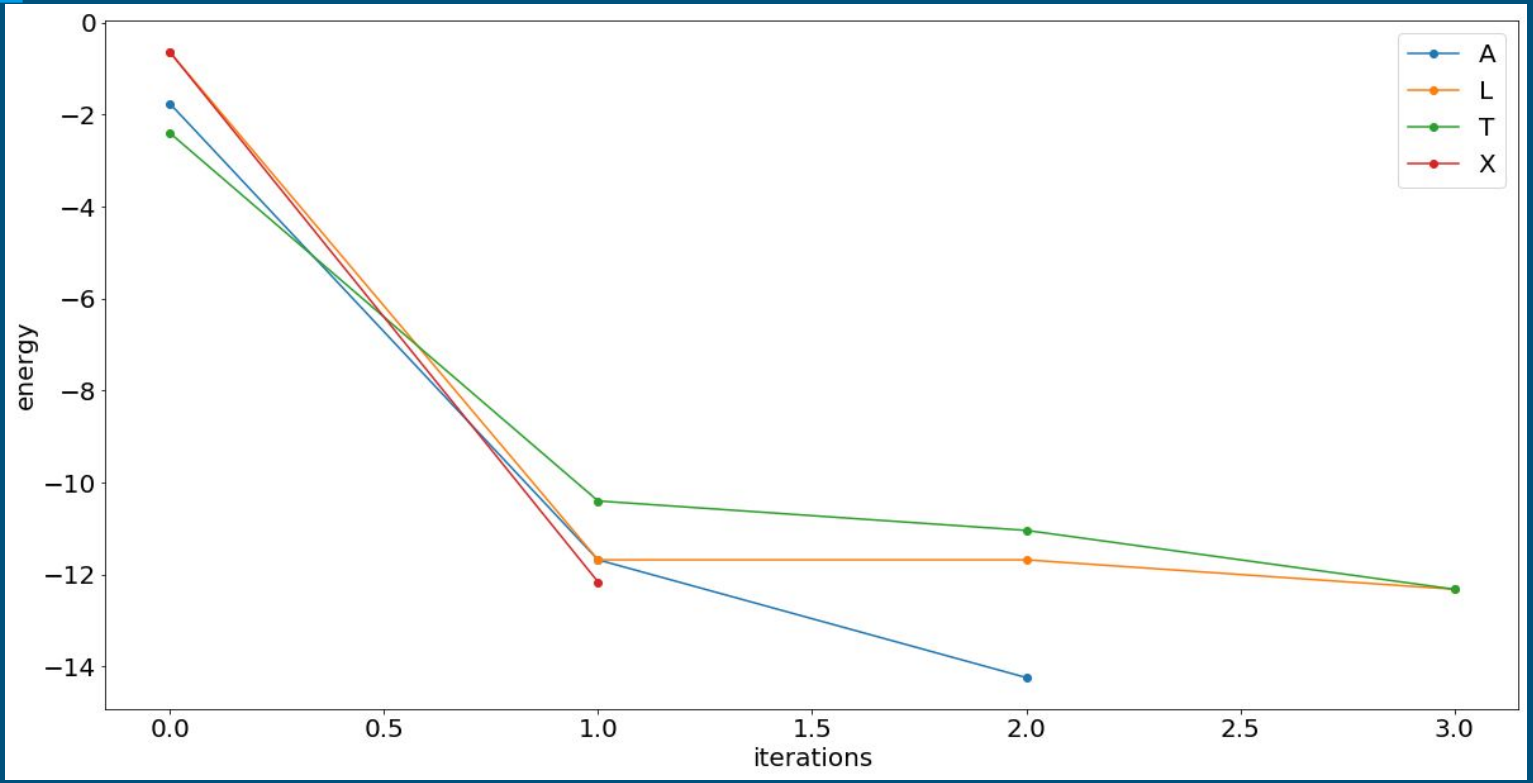
Entrada: L noise:20%

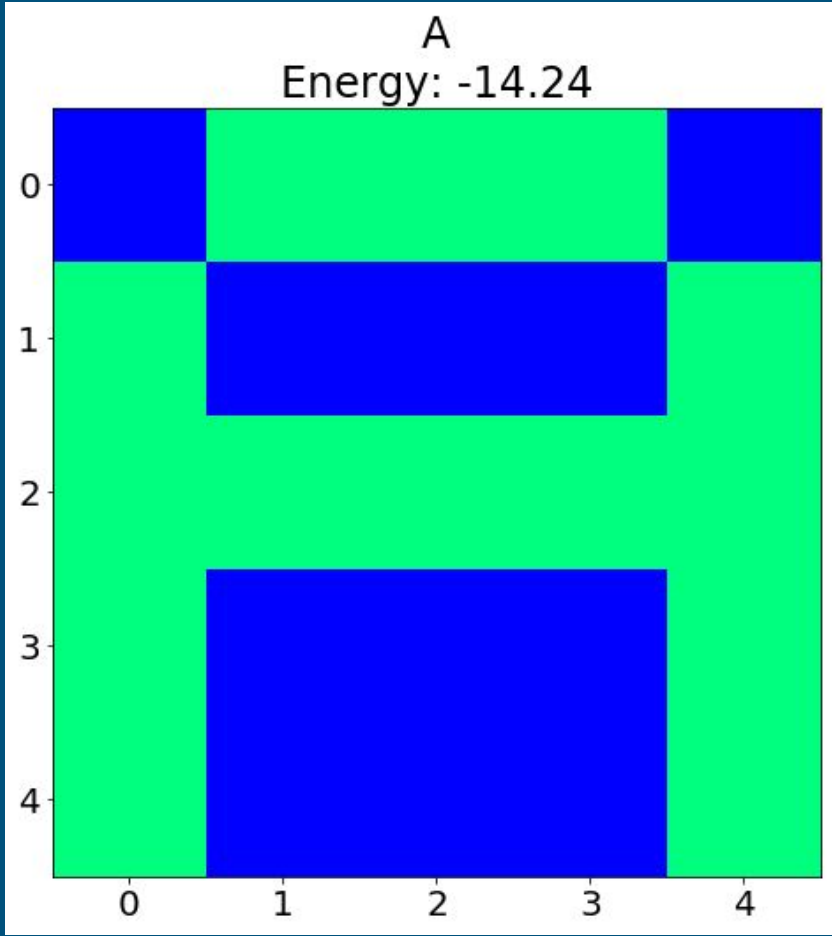


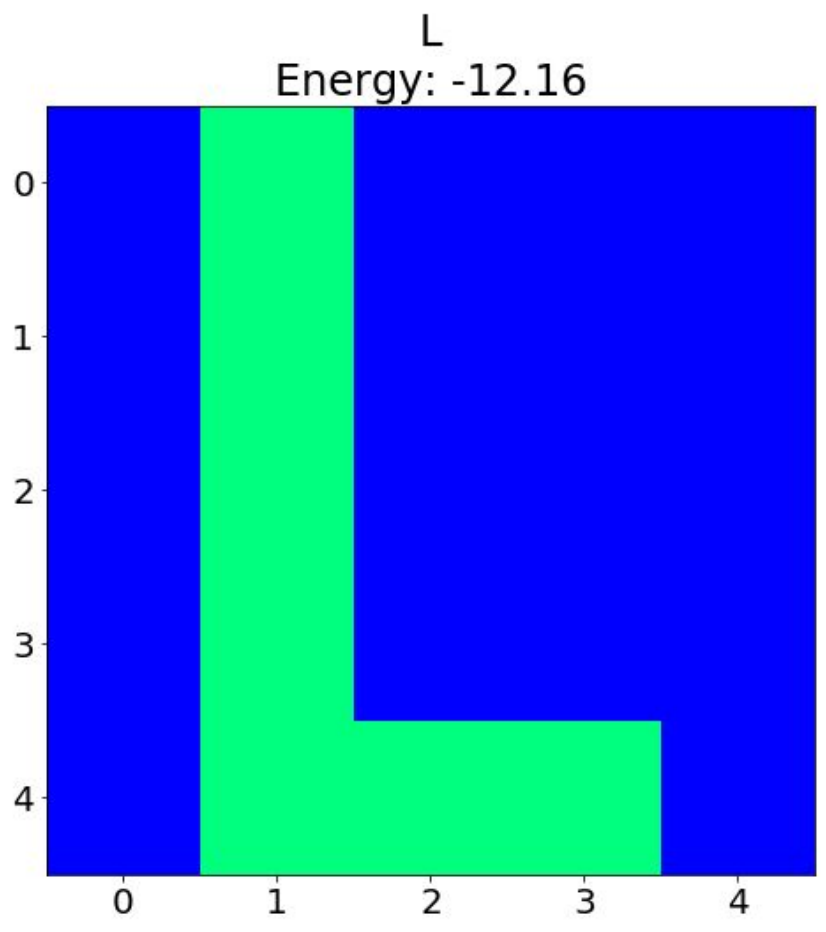
Entrada: L noise:50%

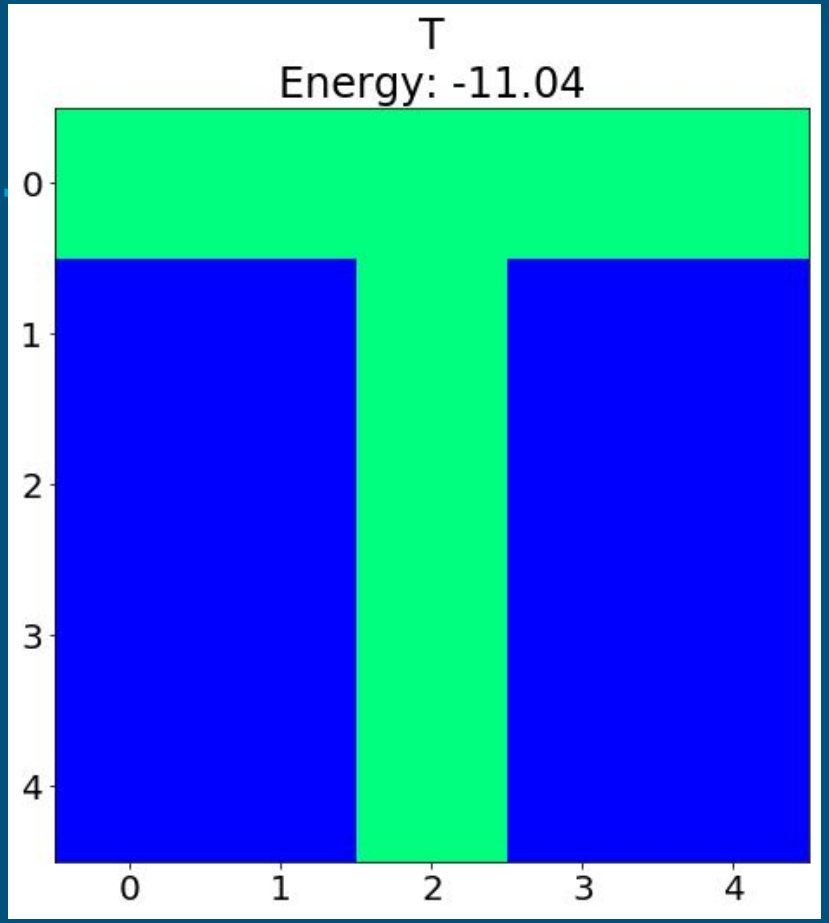


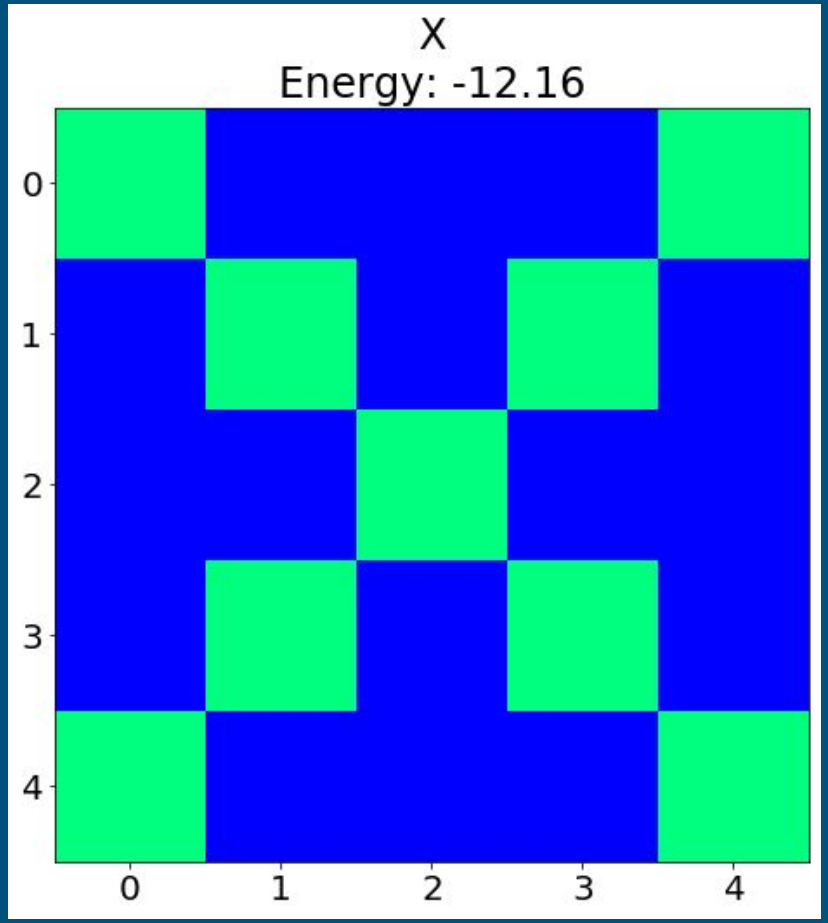
Energía vs Iteration




A
Energy: -14.24
0
1
2
3
4
0
1
2
3
4

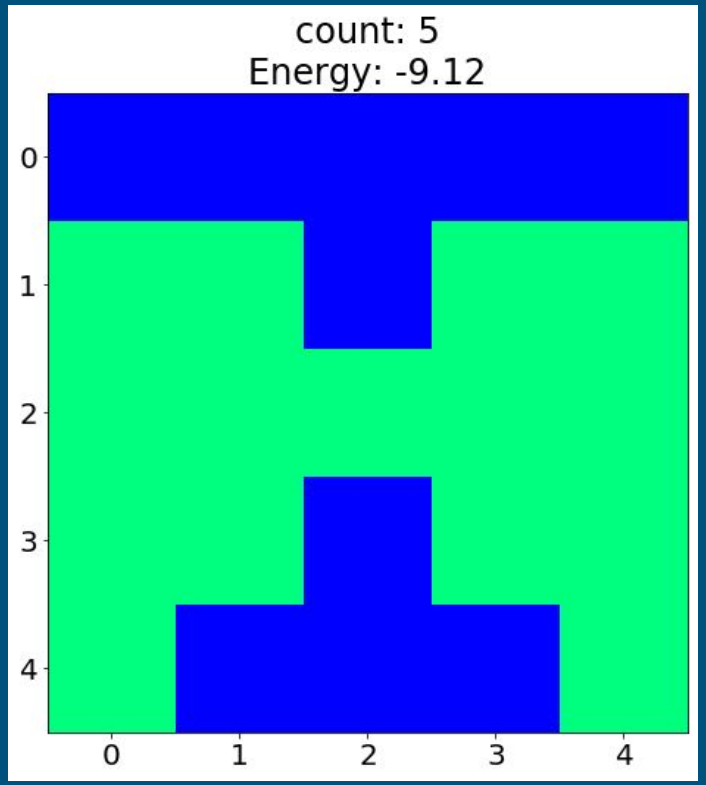

L
Energy: -12.16
0
1
2
3
4
0
1
2
3
4

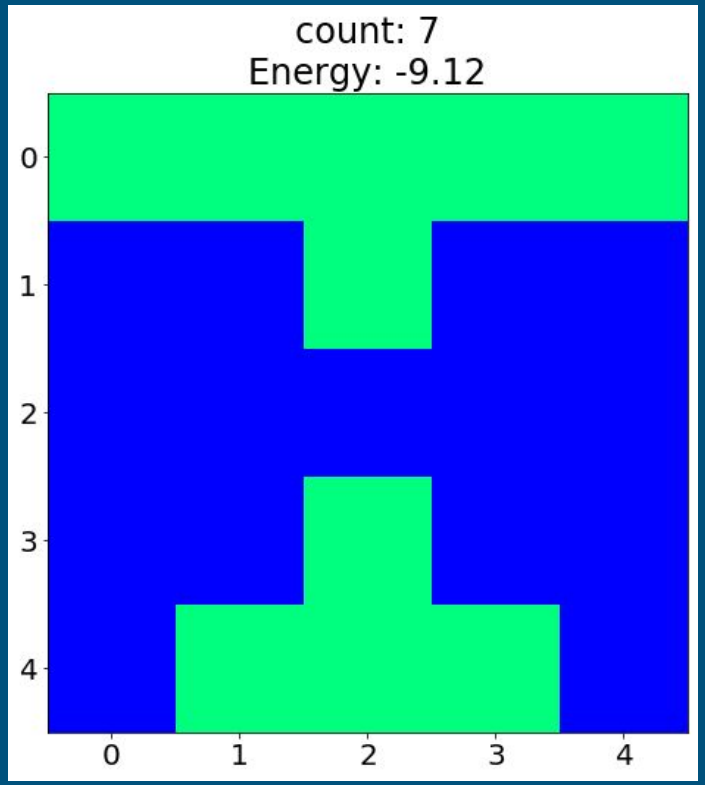

T
Energy: -11.04
0
1
2
3
4
0
1
2
3
4


X
Energy: -12.16
0
1
2
3
4
0
1
2
3
4

Estados Espúreos

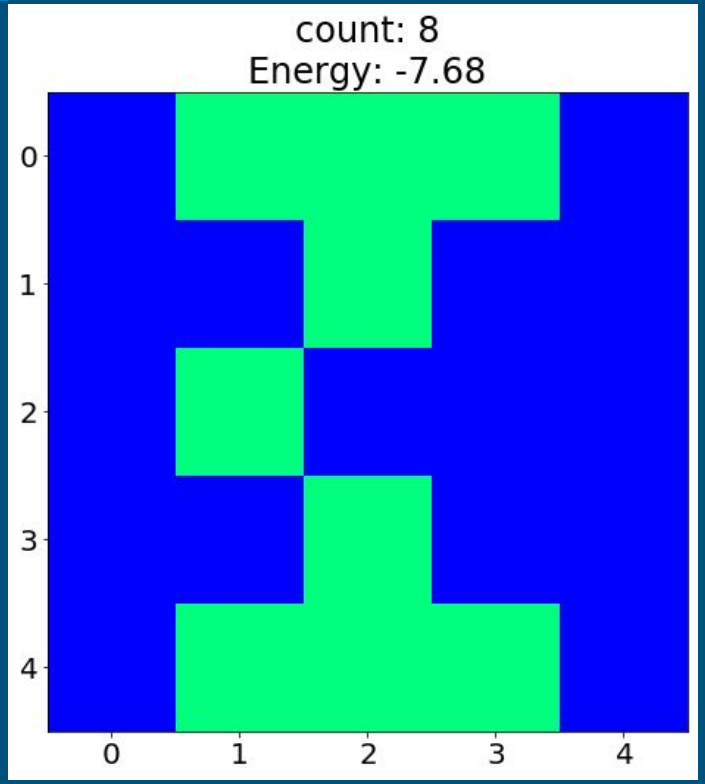
N = 2^{10} Noise = 50%

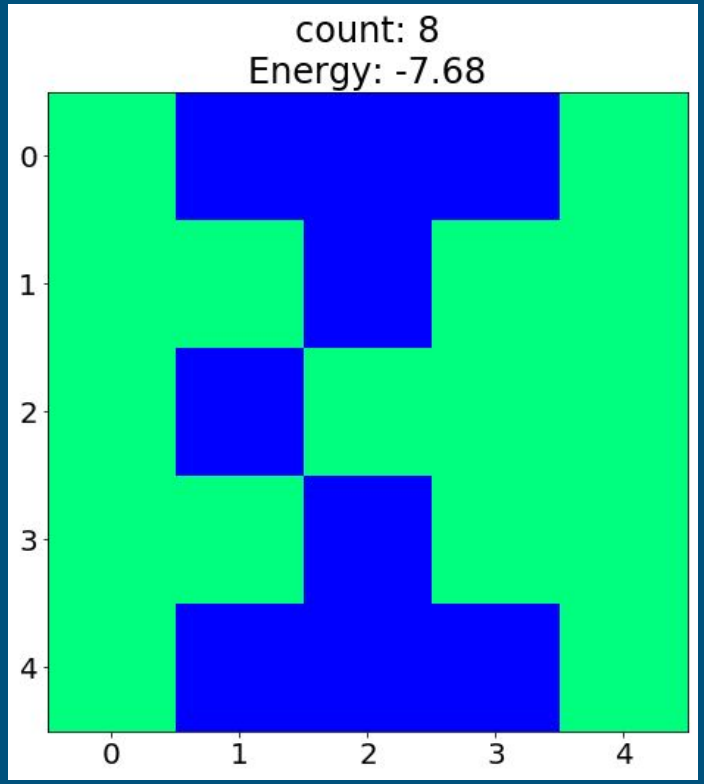




Estados Espúreos

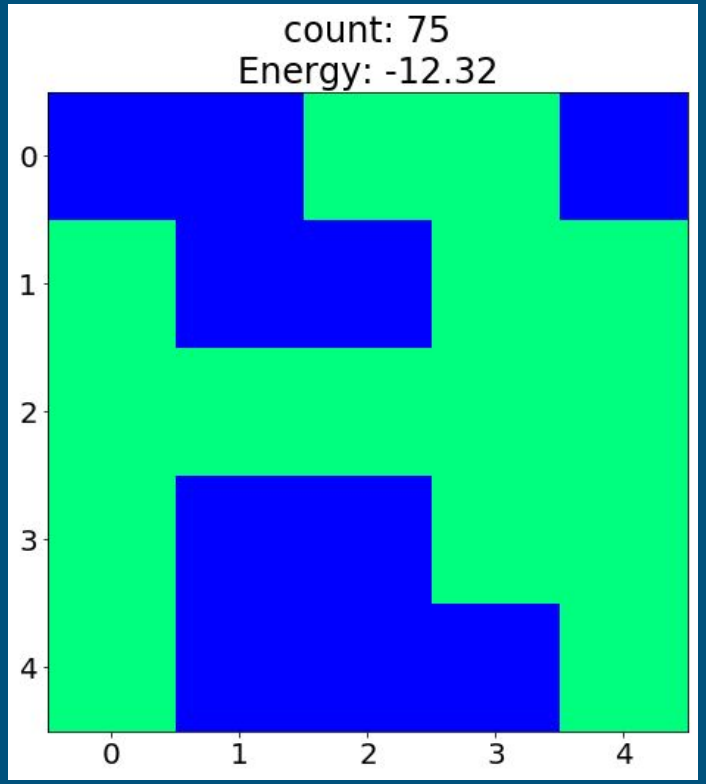
N = 2^{10} Noise = 50%

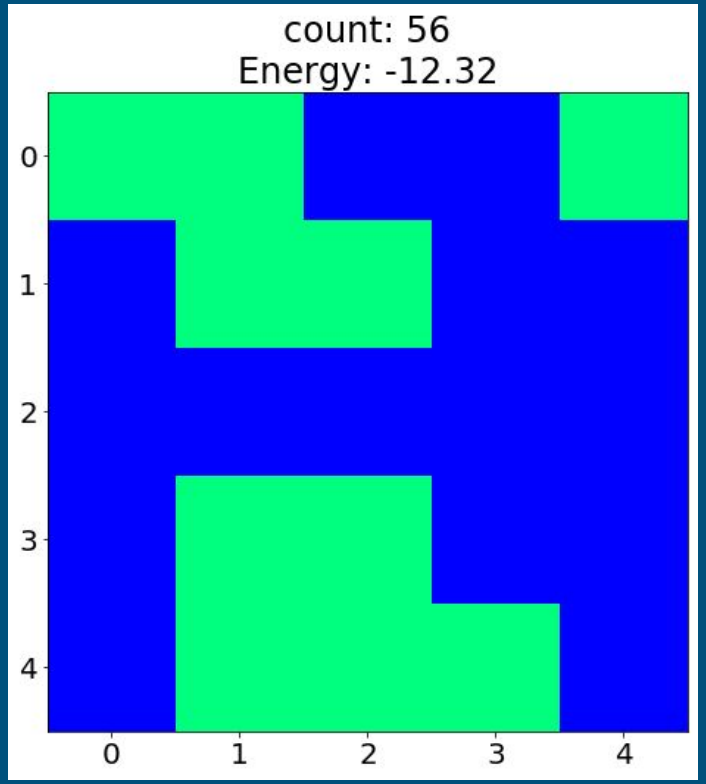




Estados Espúreos

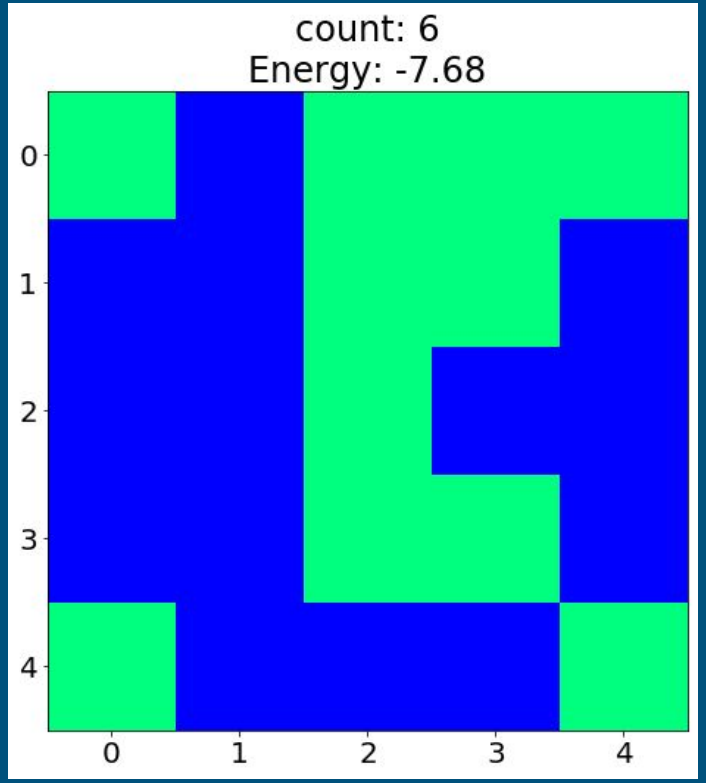
N = 2^{10} Noise = 50%





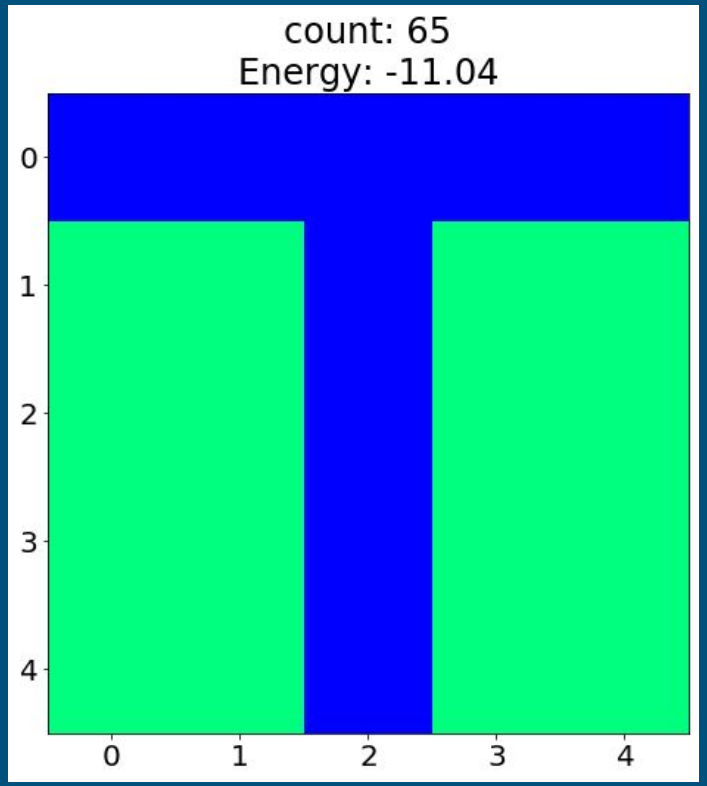
Estados Espúreos

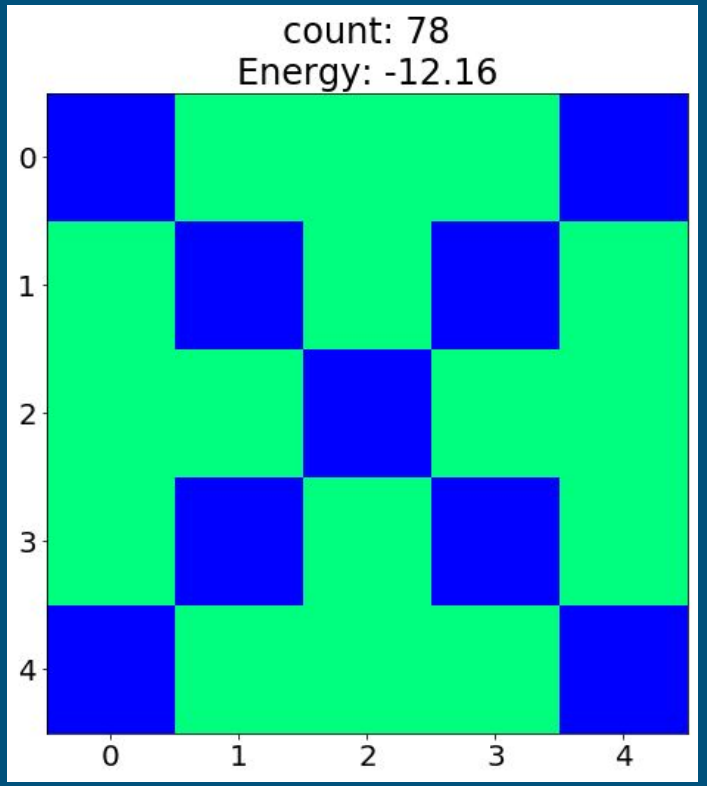
N = 2^{10} Noise = 50%



Estados Espúreos

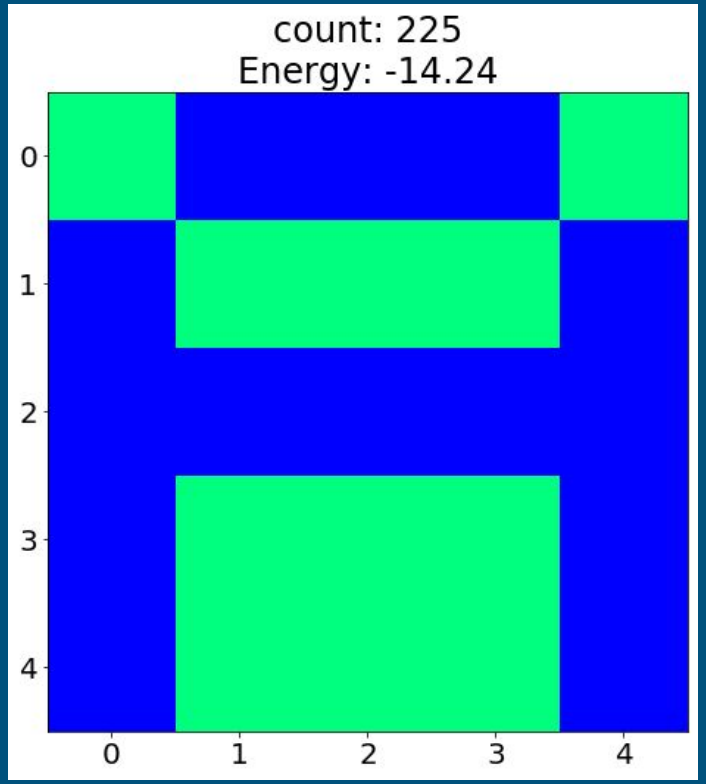
N = 2^{10} Noise = 50%

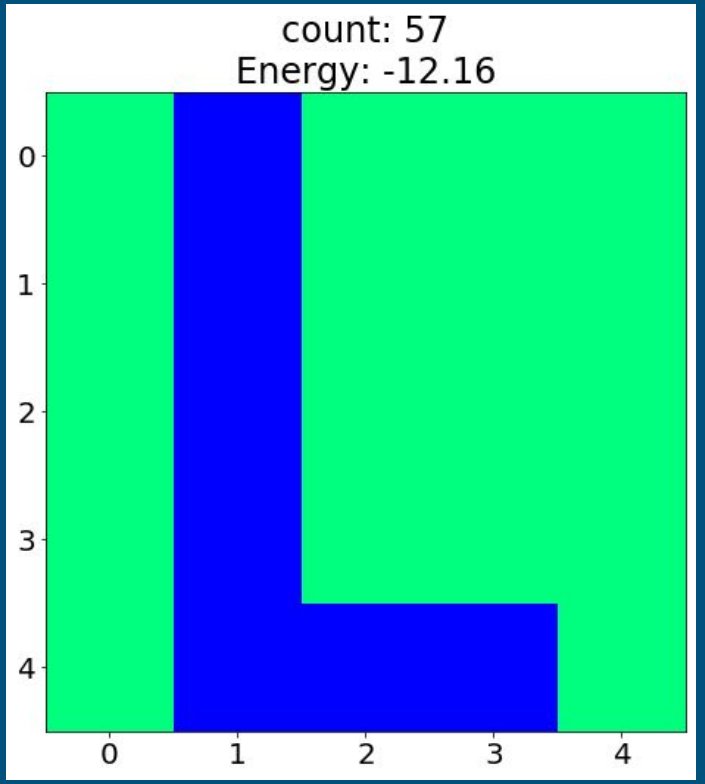




Estados Espúreos

N = 2^{10} Noise = 50%





Estados Espúreos Complementarios

$$S = (S_1, ..., S_n)$$

$$\overline{S} = (-S_1, ..., -S_n)$$

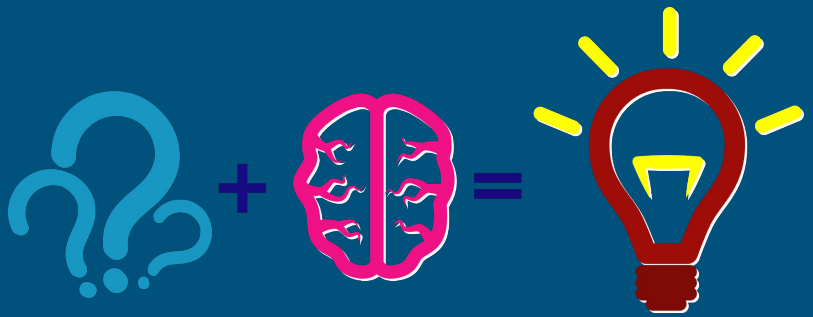
$$E(S) = -\frac{1}{2} \sum_{i,j} w_{ij} * S_i * S_j$$

$$E(\overline{S}) = -\frac{1}{2} \sum_{i,j} w_{ij} * (-S_i) * (-S_j)$$

$$= -\frac{1}{2} \sum_{i,j} w_{ij} * S_i * S_j = E(S)$$

Conclusiones Hopfield:

- Converge bien con poco ruido
- Muestra buena elección de las letras (ortogonal)
- Estados espúreos pueden ser los “complementarios” de las letras elegidas
  - demostrado con el cálculo de energía




THANKS!