Punto 1

Backpropagation:

Cabe aclarar que en el script de Backpropagation, hay veces que no se llega a un mínimo global, de forma que se diseñó el script para que si no se cumple con una meta de ECM = 0.07, reinicie el proceso. De esta forma, se comparará el tiempo necesario para cumplir con ese ECM.

Corrida	Tiempo de ejecución	ECM
1	4.33s	0.069
2	1.50s	0.069
3	1.50s	0.069
4	0.75s	0.069
5	2.63s	0.069
Promedio	2.14s	-

Puede observarse que se obtiene, en un promedio de 2.14 segundos, un ECM del orden de 0.07.

Algoritmos Genéticos:

El objetivo de estas corridas será comparar, para la misma meta de ECM, el tiempo de ejecución obtenido para el ajuste particular de los parámetros utilizados en el algoritmo.

Corrida	Tiempo de ejecución	ECM
1	20.59s	0.066
2	34.44s	0.069
3	14.50s	0.062
4	3.86s	0.051
5	24.17s	0.052
Promedio	19.51	-

Se obtuvo, que para la misma meta que en Backpropagation, el tiempo de ejecución es mucho mayor. No solo eso, sino que la variabilidad del mismo es mucho más amplia, variando desde aproximadamente 4 segundos a 35 segundos.

Simulated Annealing:

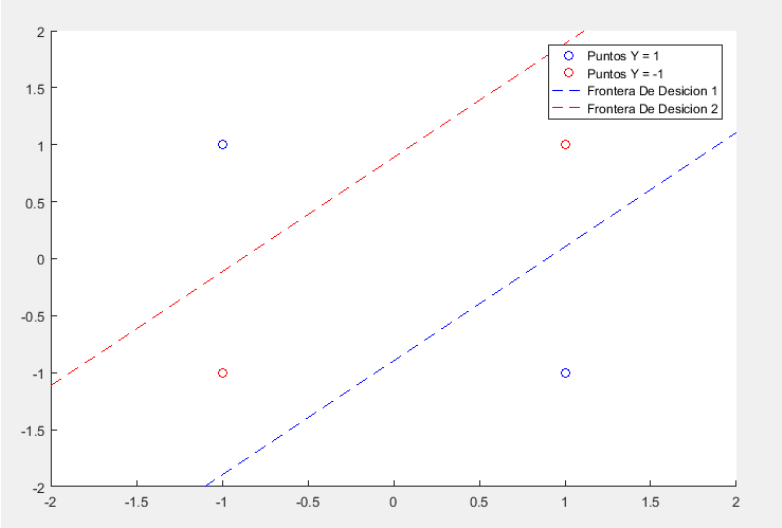
Finalmente se implemento el algoritmo de Simulated Annealing. A diferencia de los casos anteriores, no se impuso una meta de ECM, sino que el algoritmo se basa simplemente en bajar la temperatura, y ver a que se llega.

Corrida	Tiempo de ejecución	ECM
1	3.22 s	9.01e-07
2	3.26s	5.51e-05
3	3.17s	2.68e-05
4	3.46 s	1.08e-05
5	3.33s	7.19e-06
Promedio	3.08s	2.01e-05

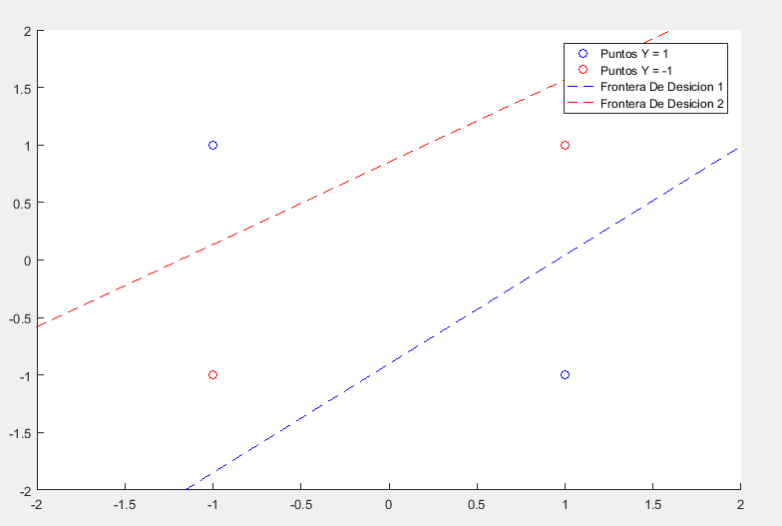
Puede observarse que se obtuvieron los mejores resultados. Se logra obtener en aproximadamente 3 segundos un error del orden de 2e-05. Otra ventaja que tiene el algoritmo es que se ejecuta en aproximadamente casi el mismo tiempo en todos los casos, dejando como variable el ECM al que se llega.

Comparación De División De Las Entradas:

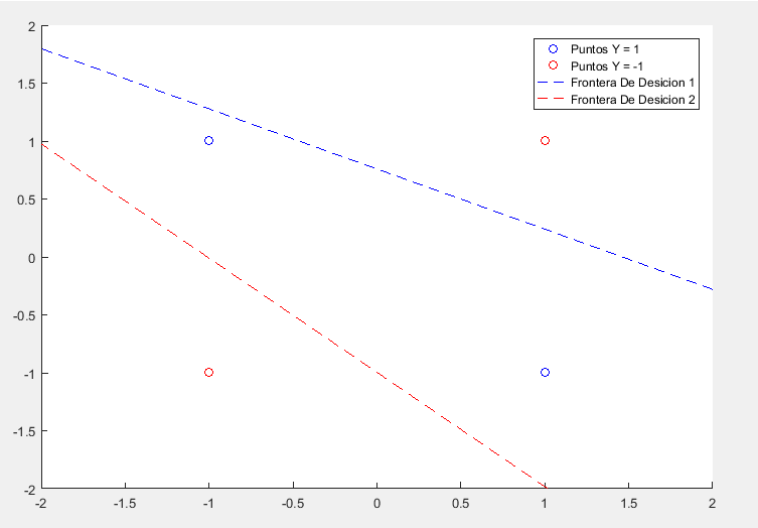
En las siguientes 3 figuras puede verse como la primera capa de neuronas divide las entradas, para cada uno de los casos. Puede observarse que todos llegan a resultados similares:



Backpropagation.



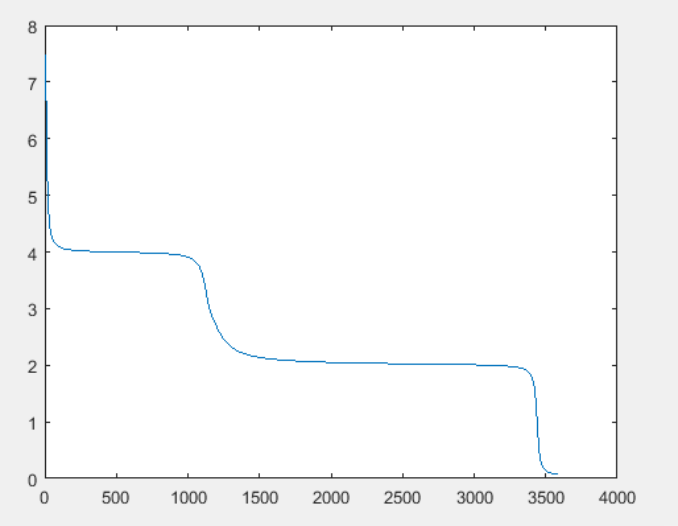
Genéticos.



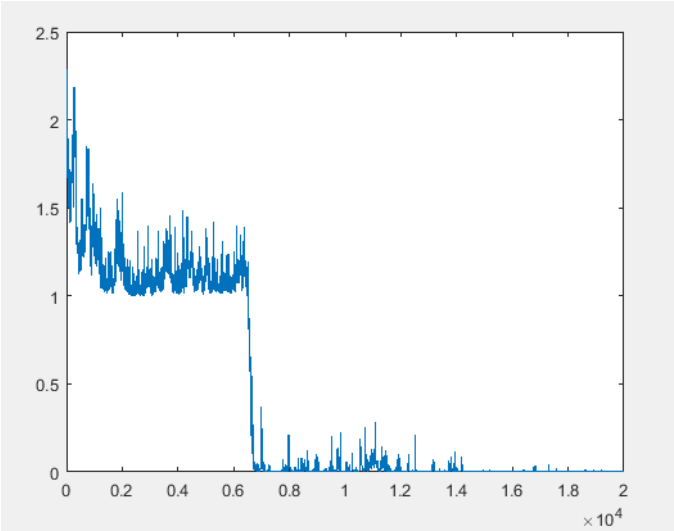
Simulated Annealing.

Comparación De Evolución Del ECM

Se generaron graficas de evolución del ECM para Backpropagation y Simulated Annealing. Puede observarse que a diferencia de Simulated Annealing, en Backpropagation el error es siempre decreciente.



Backpropagation – ECM vs Iteración



Simulated Annealing – ECM vs Iteración

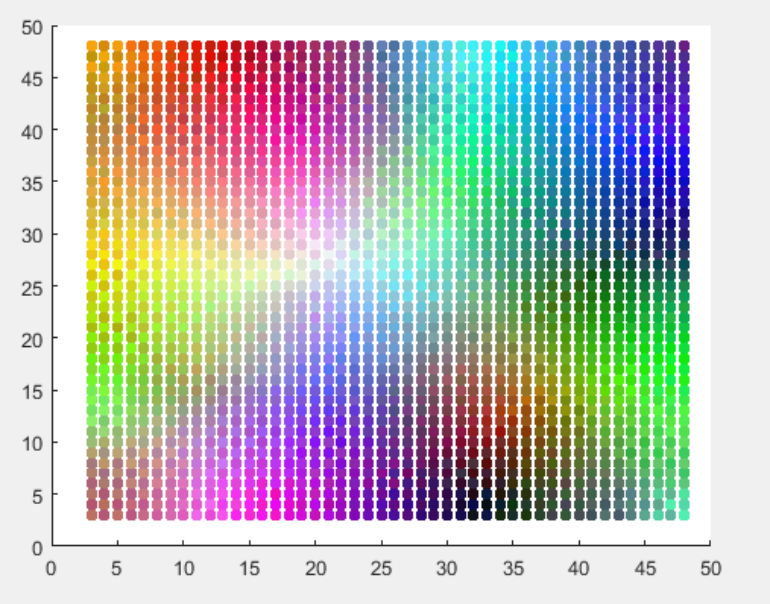
Punto 2

El algoritmo del punto 2 es una red de Kohonen que ordena colores. Básicamente el algoritmo funciona así:

- Define una matriz de NxN neuronas con 3 entradas cada una, para cada componente de color RGB.
- Inicializa los pesos sinápticos con valores aleatorios.
- Realiza muchas iteraciones tomando una muestra de color en cada una.
- Para cada muestra de color, obtiene la neurona ganadora.
- Actualiza la neurona ganadora, y a sus vecinas. Las vecinas se actualizan en 3 etapas dependiendo de en cual iteración se esté. A medida que va avanzando, va reduciendo la vecindad en cada una de las 3 etapas.
- Grafica los pesos sinápticos como colores.

Cada neurona dispara para un color en particular. Si la red logró aprender correctamente, debería ordenar los colores, de forma que colores cercanos activan neuronas cercanas. De esta forma en el grafico final deberíamos ver colores agrupados según su tipo.

Al correr el algoritmo por primera vez obtenemos el siguiente resultado:

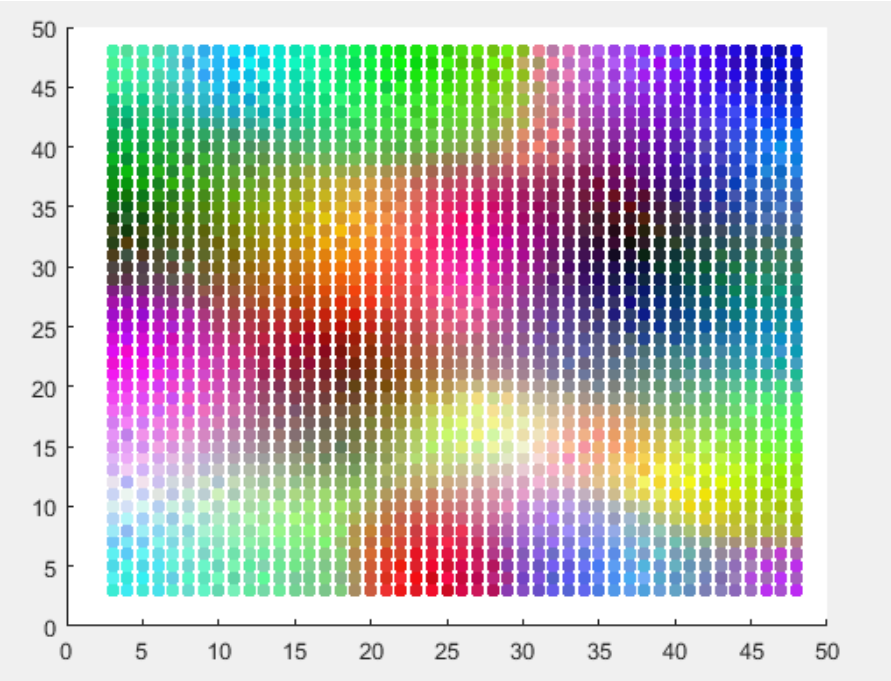


a = 1

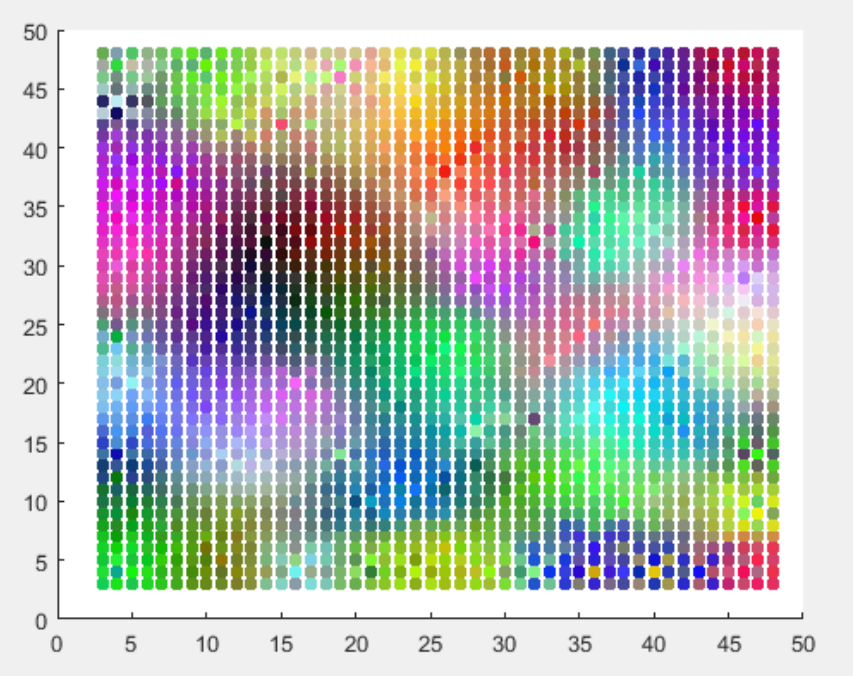
Se ve que los colores están agrupados, pero aun así hay algunas tonalidades similares que se agruparon en regiones distintas.

Modificando el valor de **a** podemos hacer más fuerte o débil la interacción con las vecinas. Si el valor de **a** es muy pequeño la interacción será más débil, perdiendo la capacidad de ordenamiento.

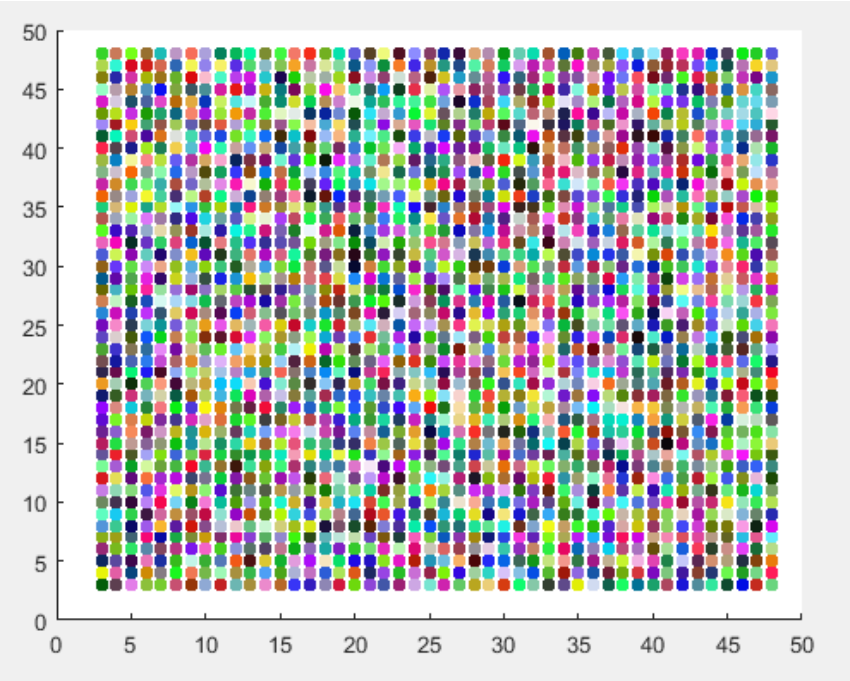
En las siguientes figuras podemos observar lo que sucede para valores más pequeños de **a**.



a = 0.5



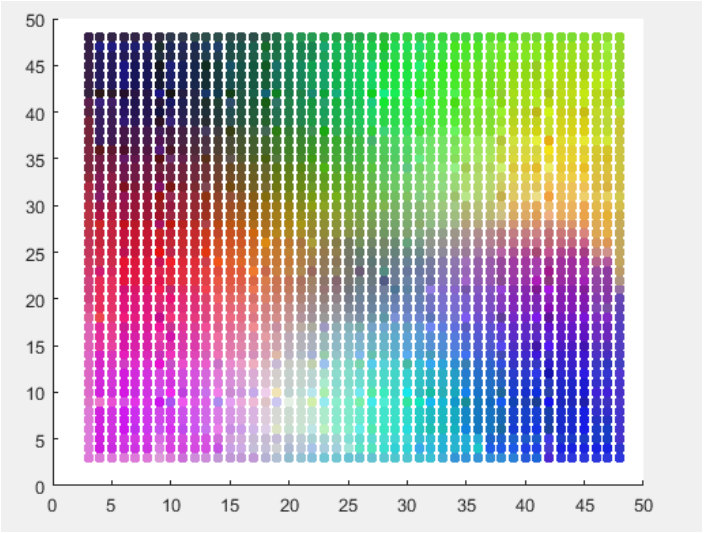
a = 0.1



a = 0.0

Se ve que a medida que **a** baja, se pierde la capacidad de ordenar. Ya en la última figura, eliminamos toda interacción entre vecinas, obteniéndose algo totalmente desordenado.

Por otro lado se probó disminuir el valor **eta**, para hacer un aprendizaje más lento, se aumentó el número de iteraciones y se utilizaron vecindades de ancho 8, 5 y 3, obteniéndose el siguiente resultado, aunque en un tiempo de ejecución mayor:



a = 1; **eta** = 0.01; Iteraciones = 600000

Se ve una leve mejora respecto al original, pero no demasiada. Se obtienen agrupaciones más grandes, pero a un costo mucho mayor.

Anastópulos Matías – 95120 – Redes Neuronales – Segundo Parcial

Archivos Entregados:

- ej_backpropagation/xor2.m
- ej_geneticos/xor2.m
- ej_geneticos/fitness.m
- ej_simulated_annealing/xor2.m
- kohonen_modificado.m
- Informe.pdf