



Universidad Simón Bolívar  
Departamento de computación  
Inteligencia artifical CI5437

## **Proyecto 2 Redes Neuronales**

Profesora :  
Carolina Martínez

Grupo:  
Carlos Farinha - 09-10270  
Javier López - 11-10552  
Nabil Márquez - 11-10683

05 de marzo del 2017

## Introducción

Las redes neuronales (también conocidos como sistemas conexionistas) son un enfoque computacional, que se basa en una gran colección de unidades neurales (también conocido como neuronas artificiales), para modelar libremente la forma en que un cerebro biológico resuelve problemas con grandes grupos de neuronas biológicas conectados por los axones. Cada unidad neuronal está conectada con otras, y los enlaces se pueden aplicar en su efecto sobre el estado de activación de unidades neuronales conectadas. Cada individuo de la unidad neuronal puede tener una función de suma, que combina conjuntamente los valores de todas las entradas.

Estos sistemas son auto-aprendizaje y formación, en lugar de programar de forma explícita, sobresalen en las zonas donde la solución o función de detección es difícil de expresar en un programa de ordenador tradicional.

Para este proyecto se quiere realizar la implementacion de una red neuronal multicapa capaz de identificar a que conjunto pertenece un punto dentro de un plano de dimensiones definidas de 20 x 20, dentro o fuera de un circulo de radio 6 ubicado en el centro de lienzo. Para ello se evalua la eficiencia de la red tomando en cuenta 6 conjuntos de diferentes para probar el aprendizaje de la red en diferentes casos.

La idea es poder proyectar un algoritmo que no solo identifique los patrones dentro del lienzo si no que su capacidad de respuesta sea optima en comparacion a sus homologos, a fin de tener una medida de eficiencia del mismo.

# Implementación

## Ejercicio 1

Haciendo uso de nuestro generador de ejemplos, creamos los conjuntos de prueba de tamaño 500, 1000 y 2000 con los siguientes comandos:

```
python generador.py 42 500  
python generador.py 5872 1000  
python generador.py 7219735 2000
```

Usando como semillas para la generación de números aleatorios *42*, *5872* y *7219735* respectivamente.

Para este ejercicio implementamos, en *neural\_net.py* la clase NeuralNetwork, que dada una lista con las dimensiones de cada capa de neuronas (incluyendo entrada y salida), internamente esto generara una lista de matrices representando las conexiones entre cada una de las capas. Los pesos son inicializados en valores al azar dados por una semilla fija.

La clase NeuralNet implementa los métodos *forward\_prop* que al recibir una entrada y el valor esperado de salida, realiza las multiplicaciones de matrices y activaciones necesarias para obtener un valor de salida, adicionalmente a esto calcula el costo asociado a esta propagación.

El método *predict* realiza una propagación y dado un *threshold* entrega la predicción más cercana a los valores de salida que entregó la red.

El método *train* dada una entrada, salidas esperadas, un tasa de aprendizaje  $\alpha$ , y un conjunto de validación, entrena la red almacenando todos los costos para los conjuntos de entrenamiento y validación.

*test* realiza el conteo de clasificaciones correctas y las métricas asociadas como precision, accuracy, sensibilidad, entre otros.

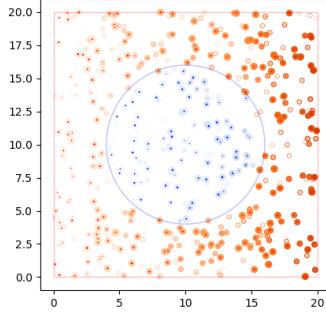
*plot\_prediction* dato un conjunto de puntos realiza un *scatter plot*, coloreando de rojos los puntos clasificados como afuera del círculo y como azules los internos. Adicionalmente se muestra el área del círculo.

*plot\_convergence* entrena la red y grafica las funciones de costo vs iteraciones para el conjunto de entrenamiento y de validación dados.

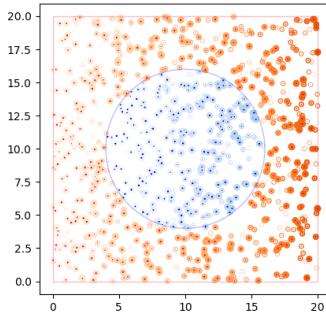
# Presentación de los resultados

## Caso Training

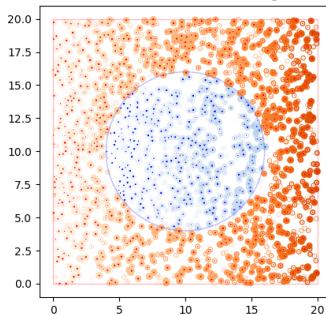
Red de una capa con 5 neuronas. Alpha = 0.3.  
Iter = 2000. Dataset = training500



Red de una capa con 6 neuronas. Alpha = 0.3.  
Iter = 2000. Dataset = training1000

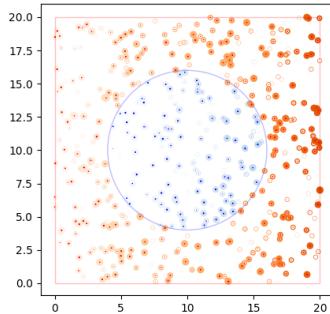


Red de una capa con 7 neuronas. Alpha = 0.3.  
Iter = 2000. Dataset = training2000

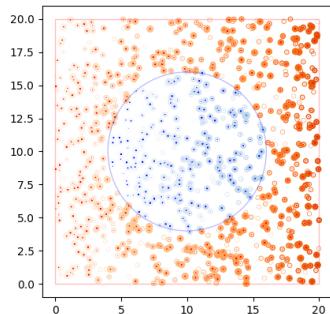


## Caso Test

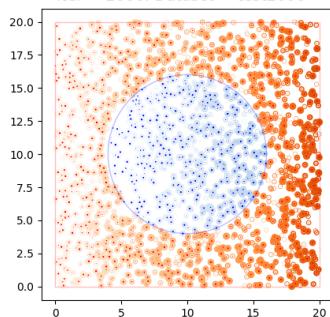
Red de una capa con 2 neuronas. Alpha = 0.3.  
Iter = 2000. Dataset = test500



Red de una capa con 3 neuronas. Alpha = 0.3.  
Iter = 2000. Dataset = test1000



Red de una capa con 4 neuronas. Alpha = 0.3.  
Iter = 2000. Dataset = test2000



## Ejercicio 2

### Errores de Configuración

Los errores de configuracion fueron las siguientes para los distintos dataset: Para entender la red que se esta trabajando se utiliza el formato (test v train)[0-10] donde se indica si el conjunto a evaluar es de test o training y el numero de neuronas en una capa sobre el cual se evalua en el caso de 2 capas se indica con una coma y el numero de neuronas de cada capa. Para poder comprender las tablas se indica la siguiente leyenda.

$t\_n = true\_negative$   $f\_p = false\_positive$   $t\_p = true\_positive$   $f\_n = false\_negative$   $a = accuracy$   $p = precision$   $r = recall$   $f\_s = f\text{-score}$

- Predicción para dataset de 500

dataset	t_n	f_p	t_p	f_n	a	p	r	f_s
train2	374.0	0.0	123.0	3.0	0.994	1.0	0.9761	0.9879
train3	374.0	0.0	122.0	4.0	0.992	1.0	0.9682	0.9838
train4	374.0	0.0	123.0	3.0	0.994	1.0	0.9761	0.9879
train5	374.0	0.0	122.0	4.0	0.992	1.0	0.9682	0.9838
train6	374.0	0.0	123.0	3.0	0.994	1.0	0.9761	0.9879
train7	374.0	0.0	122.0	4.0	0.992	1.0	0.9682	0.9838
train8	374.0	0.0	122.0	4.0	0.992	1.0	0.9682	0.9838
train9	374.0	0.0	122.0	4.0	0.992	1.0	0.9682	0.9838
test9	353.0	0.0	141.0	6.0	0.988	1.0	0.9591	0.9791
train10	374.0	0.0	123.0	3.0	0.994	1.0	0.9761	0.9879
test10	353.0	0.0	141.0	6.0	0.988	1.0	0.9591	0.9791
train2,2	373.0	1.0	124.0	2.0	0.994	0.992	0.9841	0.9880
test2,2	353.0	0.0	141.0	6.0	0.988	1.0	0.9591	0.9791
train5,5	373.0	1.0	125.0	1.0	0.996	0.992	0.9920	0.9920
test5,5	353.0	0.0	141.0	6.0	0.988	1.0	0.9591	0.9791
train10,10	746.0	2.0	252.0	0.0	0.998	0.992	1.0	0.9960
test10,10	704.0	2.0	284.0	10.0	0.988	0.993	0.9659	0.9793

- Predicción para dataset de 1000

dataset	t_n	f_p	t_p	f_n	a	p	r	f_s
train2	681.0	1.0	318.0	0.0	0.999	0.9968	1.0	0.998
test2	703.0	4.0	292.0	1.0	0.995	0.9864	0.996	0.991
train3	681.0	1.0	318.0	0.0	0.999	0.9968	1.0	0.998
test3	703.0	4.0	292.0	1.0	0.995	0.9864	0.996	0.991
train4	681.0	1.0	318.0	0.0	0.999	0.9968	1.0	0.998
test4	703.0	4.0	292.0	1.0	0.995	0.9864	0.996	0.991
train5	681.0	1.0	318.0	0.0	0.999	0.9968	1.0	0.998

dataset	t_n	f_p	t_p	f_n	a	p	r	f_s
test5	704.0	3.0	292.0	1.0	0.996	0.9898	0.996	0.993
train6	681.0	1.0	318.0	0.0	0.999	0.9968	1.0	0.998
test6	704.0	3.0	292.0	1.0	0.996	0.9898	0.996	0.993
train7	681.0	1.0	318.0	0.0	0.999	0.9968	1.0	0.998
test7	704.0	3.0	292.0	1.0	0.996	0.9898	0.996	0.993
train8	681.0	1.0	318.0	0.0	0.999	0.9968	1.0	0.998
test8	703.0	4.0	292.0	1.0	0.995	0.9864	0.996	0.991
train9	681.0	1.0	318.0	0.0	0.999	0.9968	1.0	0.998
test9	704.0	3.0	292.0	1.0	0.996	0.9898	0.996	0.993
train10	681.0	1.0	318.0	0.0	0.999	0.9968	1.0	0.998
test10	704.0	3.0	292.0	1.0	0.996	0.9898	0.996	0.993
train2,2	681.0	1.0	318.0	0.0	0.999	0.9968	1.0	0.998
test2,2	703.0	4.0	292.0	1.0	0.995	0.9864	0.996	0.991
train5,5	681.0	1.0	318.0	0.0	0.999	0.9968	1.0	0.998
test5,5	703.0	4.0	292.0	1.0	0.995	0.9864	0.996	0.991
train10,10	1362.0	2.0	636.0	0.0	0.999	0.9968	1.0	0.998
test10,10	1408.0	6.0	586.0	0.0	0.997	0.9898	1.0	0.994

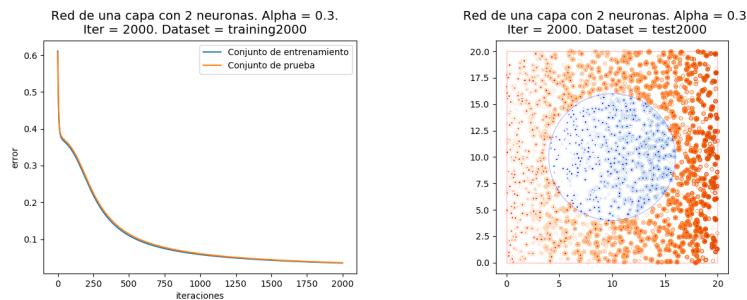
- Predicción para dataset de 2000

dataset	t_n	f_p	t_p	f_n	a	p	r	f_s
train2	1438.0	3.0	553.0	6.0	0.9955	0.994	0.9892	0.9919
test2	1426.0	3.0	563.0	8.0	0.9945	0.994	0.9859	0.9903
train3	1438.0	3.0	553.0	6.0	0.9955	0.994	0.9892	0.9919
test3	1426.0	3.0	563.0	8.0	0.9945	0.994	0.9859	0.9903
train4	1438.0	3.0	553.0	6.0	0.9955	0.994	0.9892	0.9919
test4	1426.0	3.0	564.0	7.0	0.995	0.994	0.9877	0.9912
train5	1438.0	3.0	553.0	6.0	0.9955	0.994	0.9892	0.9919
test5	1426.0	3.0	560.0	11.0	0.993	0.994	0.9807	0.9876
train6	1438.0	3.0	553.0	6.0	0.9955	0.994	0.9892	0.9919
test6	1426.0	3.0	564.0	7.0	0.995	0.994	0.9877	0.9912
train7	1438.0	3.0	553.0	6.0	0.9955	0.994	0.9892	0.9919
test7	1426.0	3.0	563.0	8.0	0.9945	0.994	0.9859	0.9903
train8	1438.0	3.0	553.0	6.0	0.9955	0.994	0.9892	0.9919
test8	1426.0	3.0	564.0	7.0	0.995	0.994	0.9877	0.9912
train9	1438.0	3.0	553.0	6.0	0.995	0.9946	0.9892	0.9919
test9	1426.0	3.0	564.0	7.0	0.995	0.9947	0.9877	0.9912
train10	1438.0	3.0	553.0	6.0	0.995	0.9946	0.9892	0.9919
test10	1426.0	3.0	561.0	10.0	0.993	0.9946	0.9824	0.9885
train2,2	1438.0	3.0	553.0	6.0	0.995	0.9946	0.9892	0.9919
test2,2	1426.0	3.0	564.0	7.0	0.995	0.9947	0.9877	0.9912
train5,5	1438.0	3.0	554.0	5.0	0.996	0.9946	0.9910	0.9928
test5,5	1426.0	3.0	565.0	6.0	0.995	0.9947	0.9894	0.9920

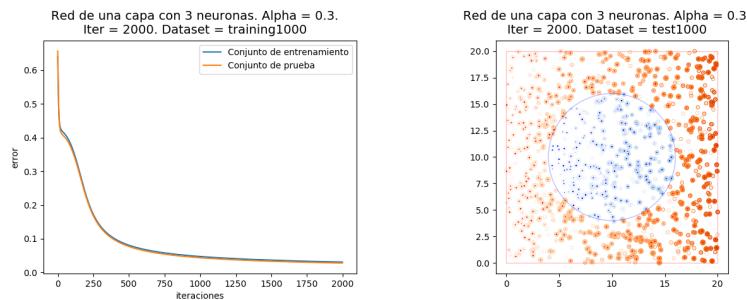
dataset	t_n	f_p	t_p	f_n	a	p	r	f_s
train10,10	2876.0	6.0	1112.0	6.0	0.997	0.9946	0.9946	0.9946
test10,10	2854.0	4.0	1128.0	14.0	0.995	0.9964	0.9877	0.9920

Los mejores conjuntos de entrenamiento encontrados variando en correlacion al numero de neuronas en la red fue el siguiente:

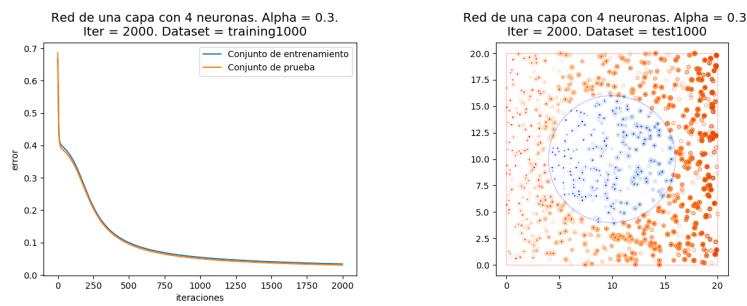
- Para 2 neuronas en 1 capa



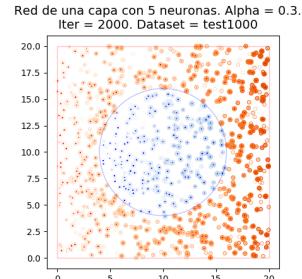
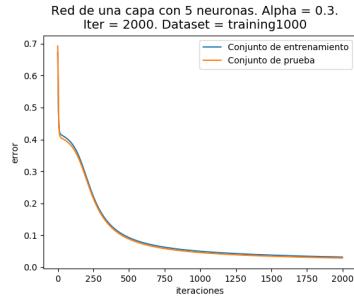
- Para 3 neuronas en 1 capa



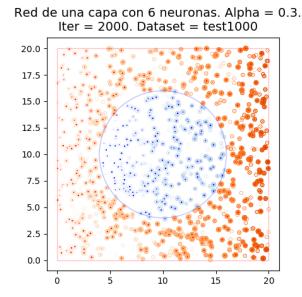
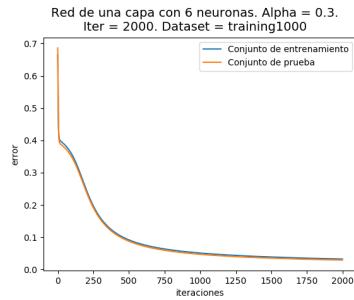
- Para 4 neuronas en 1 capa



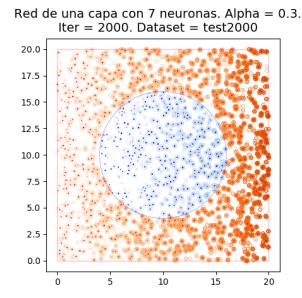
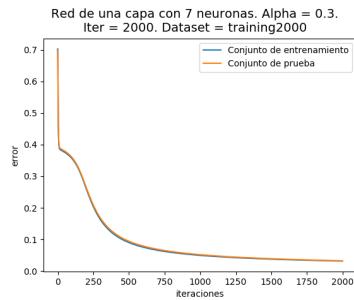
- Para 5 neuronas en 1 capa



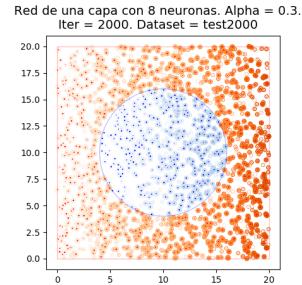
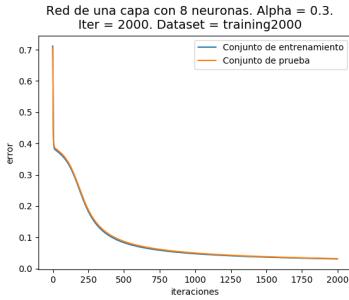
- Para 6 neuronas en 1 capa



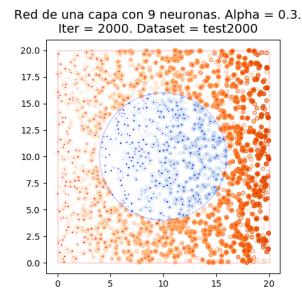
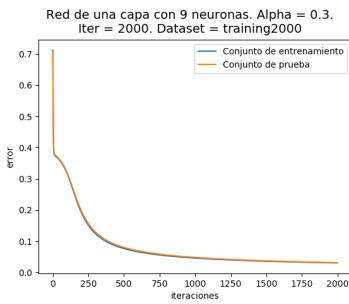
- Para 7 neuronas en 1 capa



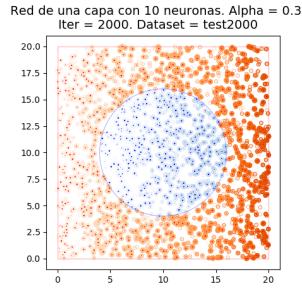
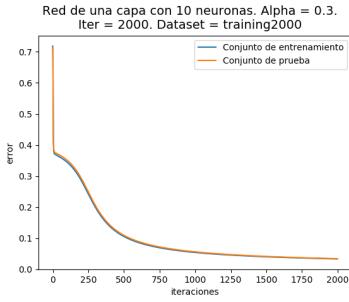
- Para 8 neuronas en 1 capa



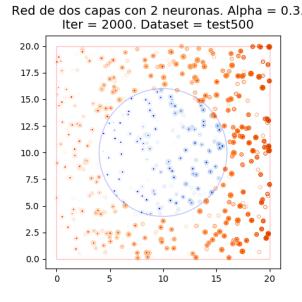
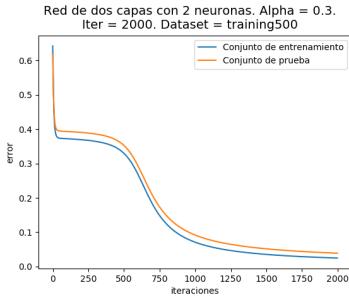
- Para 9 neuronas en 1 capa



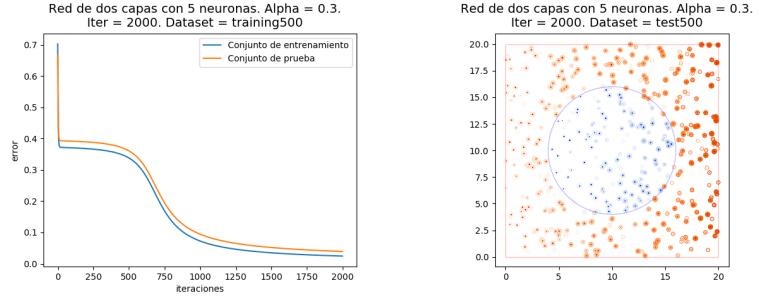
- Para 10 neuronas en 1 capa



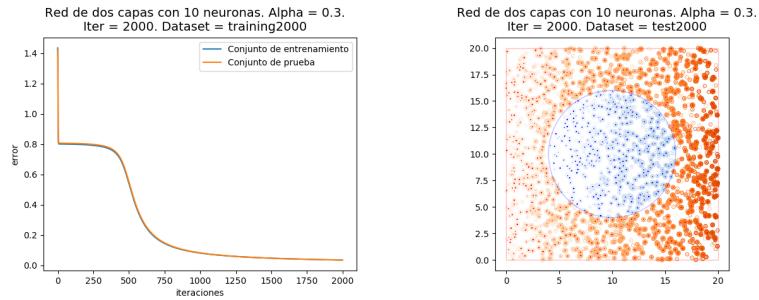
- Para 2 neuronas en 2 capas



- Para 5 neuronas en 2 capas

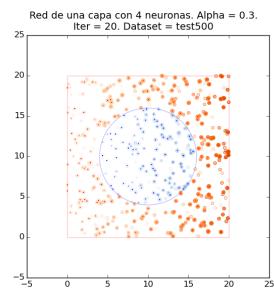


- Para 10 neuronas en 2 capas



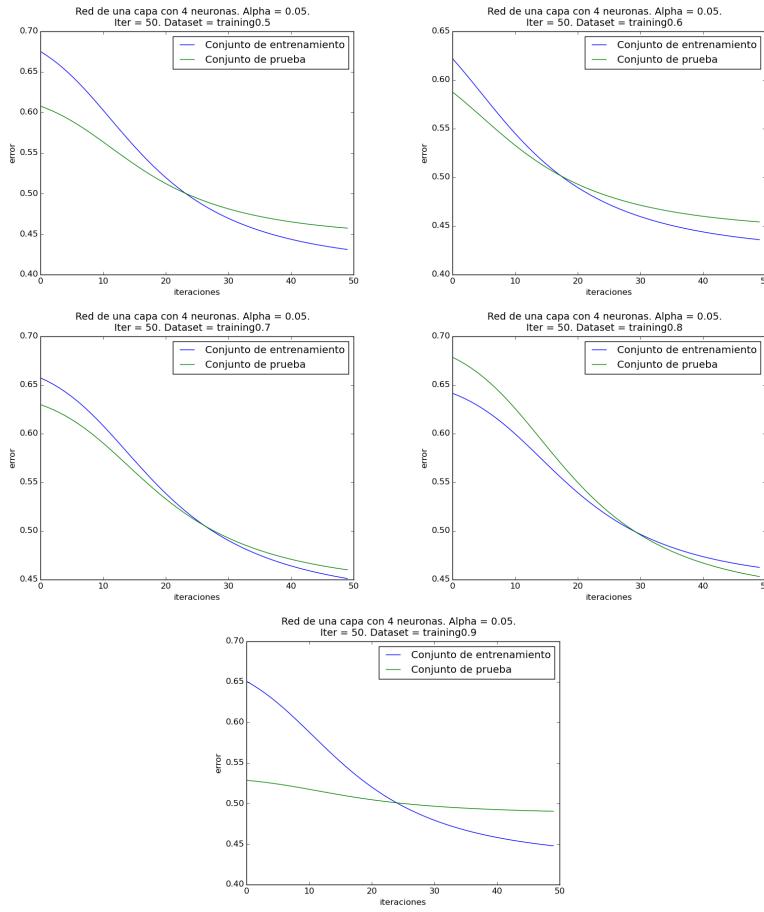
### Ejercicio 3

Entrenando la red sobre los datos del conjunto Iris Data Set se obtienen los siguientes resultados de manera general

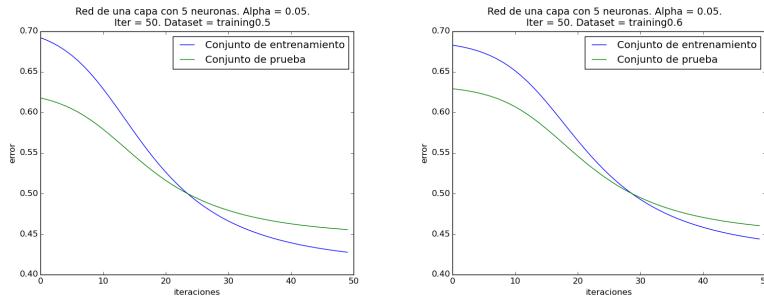


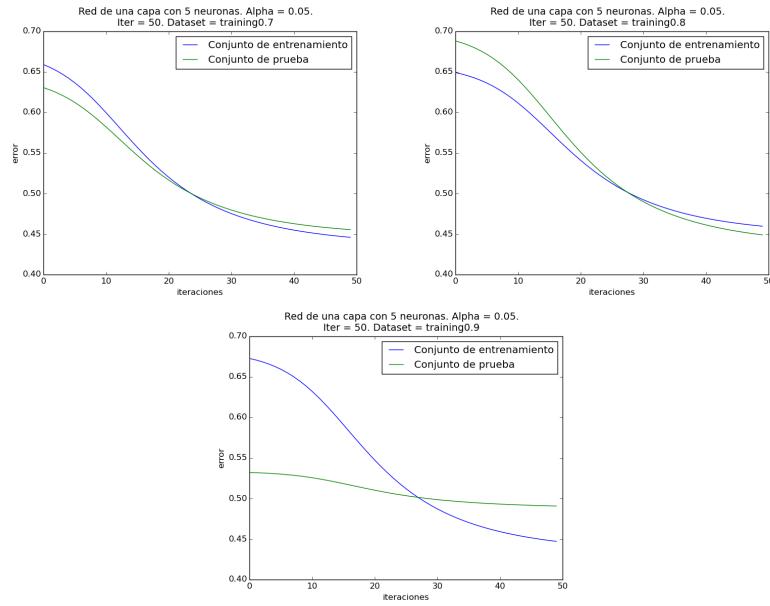
### Clasificador binario

- Para 4 neuronas en 1 capas

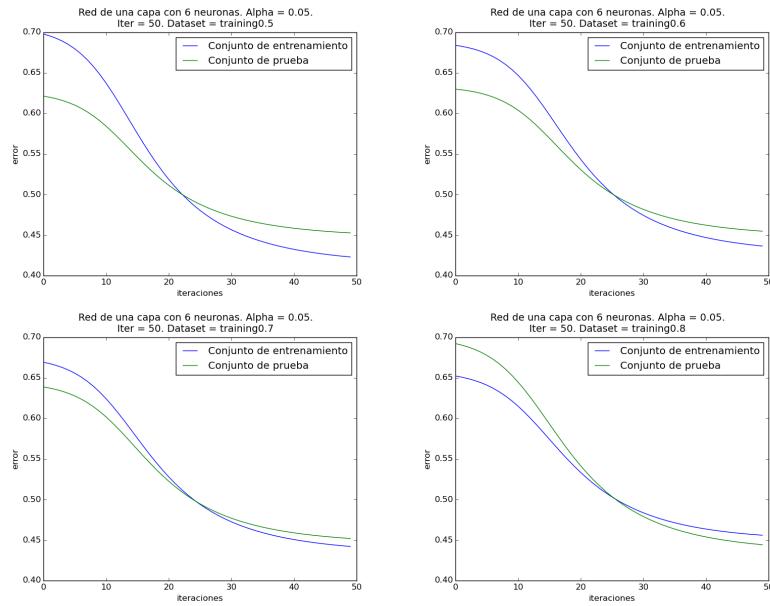


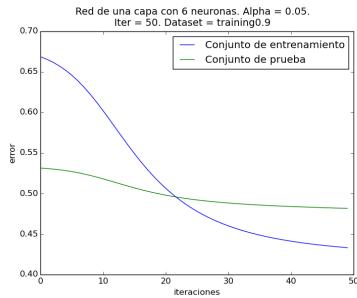
- Para 5 neuronas en 1 capas



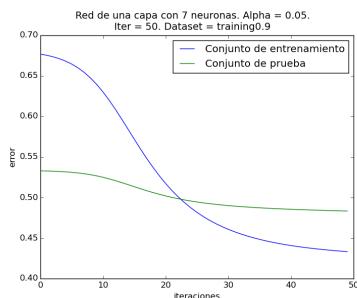
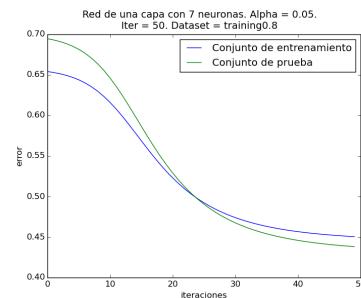
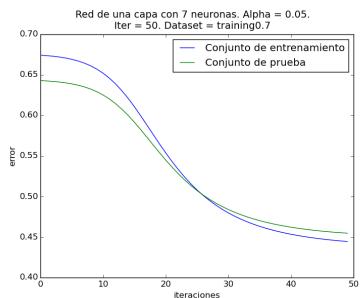
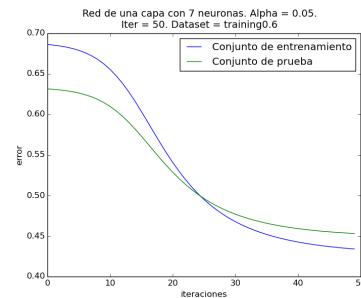
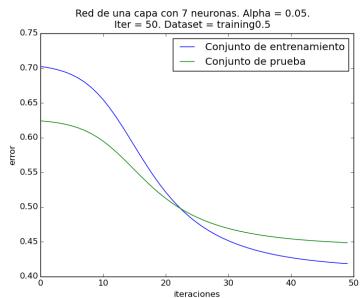


- Para 6 neuronas en 1 capas

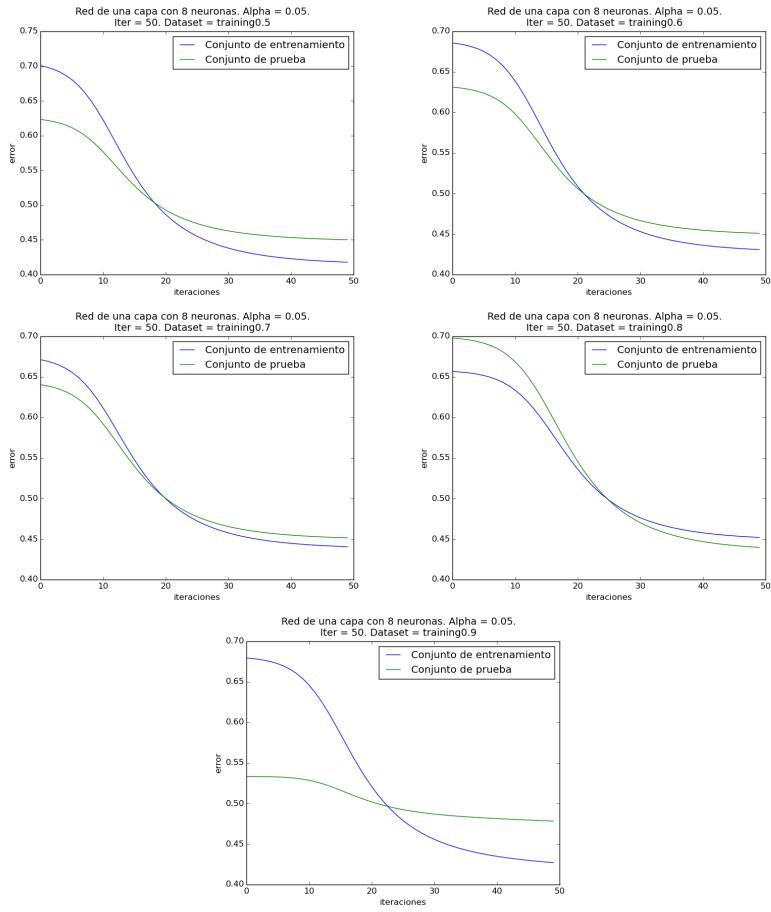




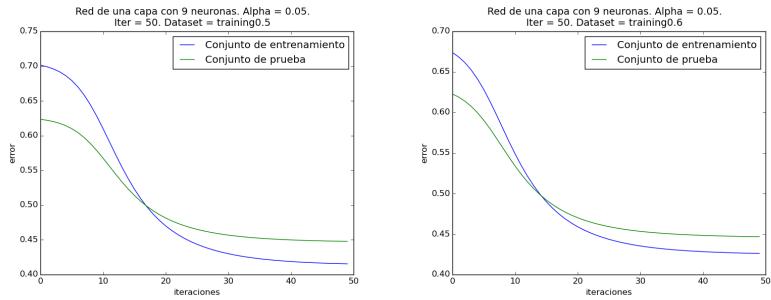
- Para 7 neuronas en 1 capas

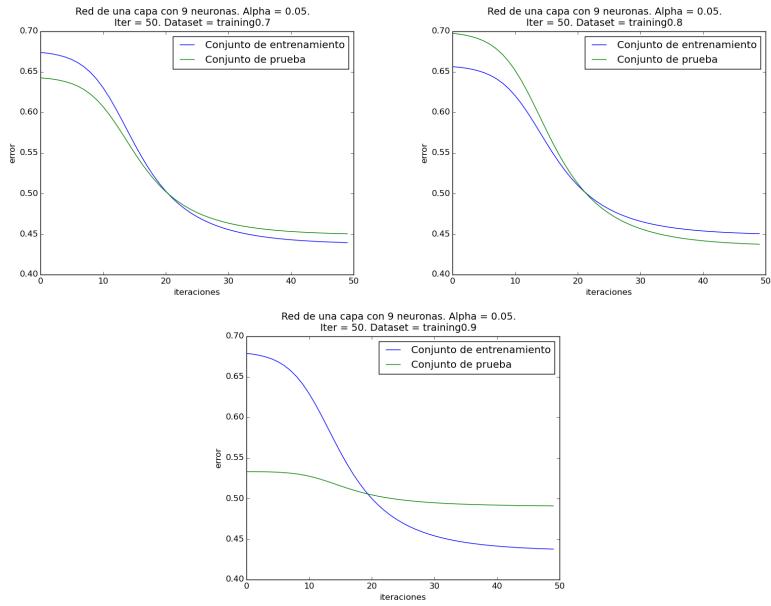


- Para 8 neuronas en 1 capas

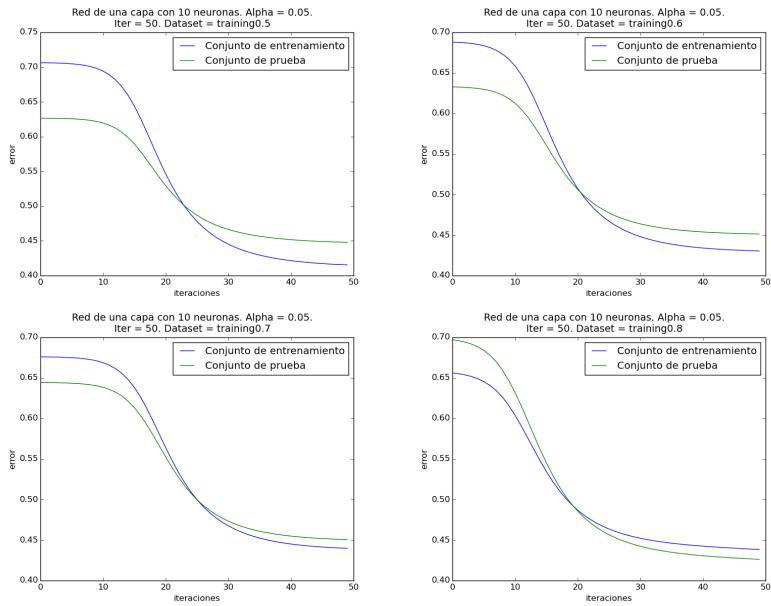


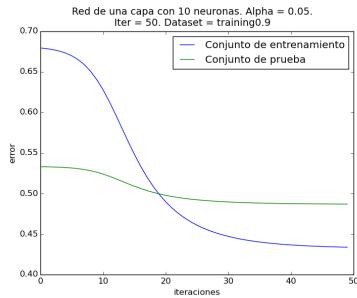
- Para 9 neuronas en 1 capas



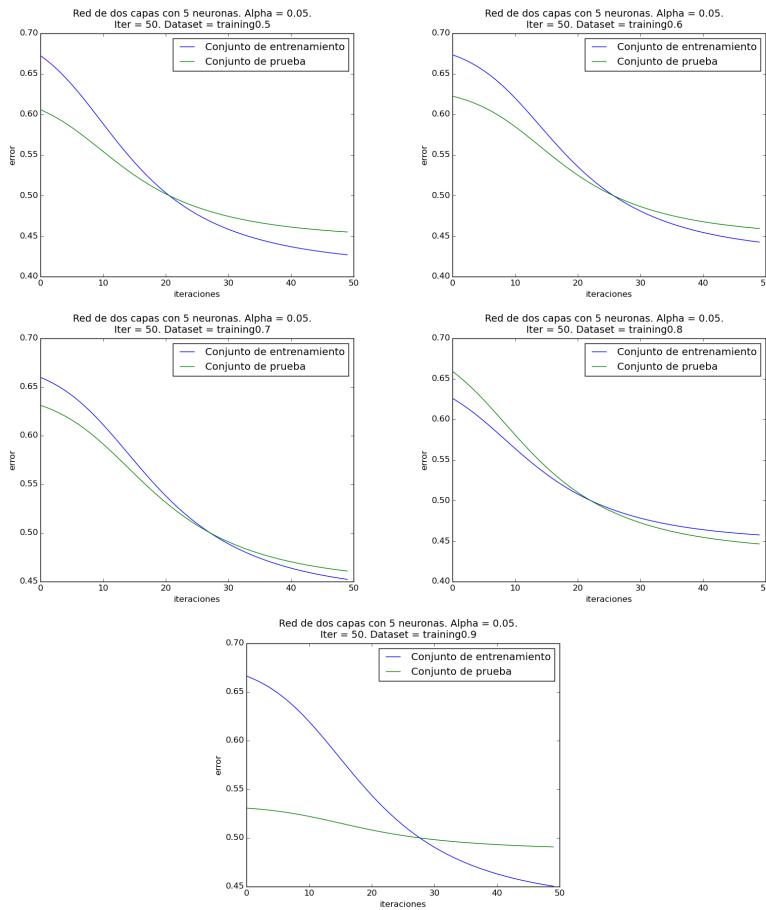


- Para 10 neuronas en 1 capas



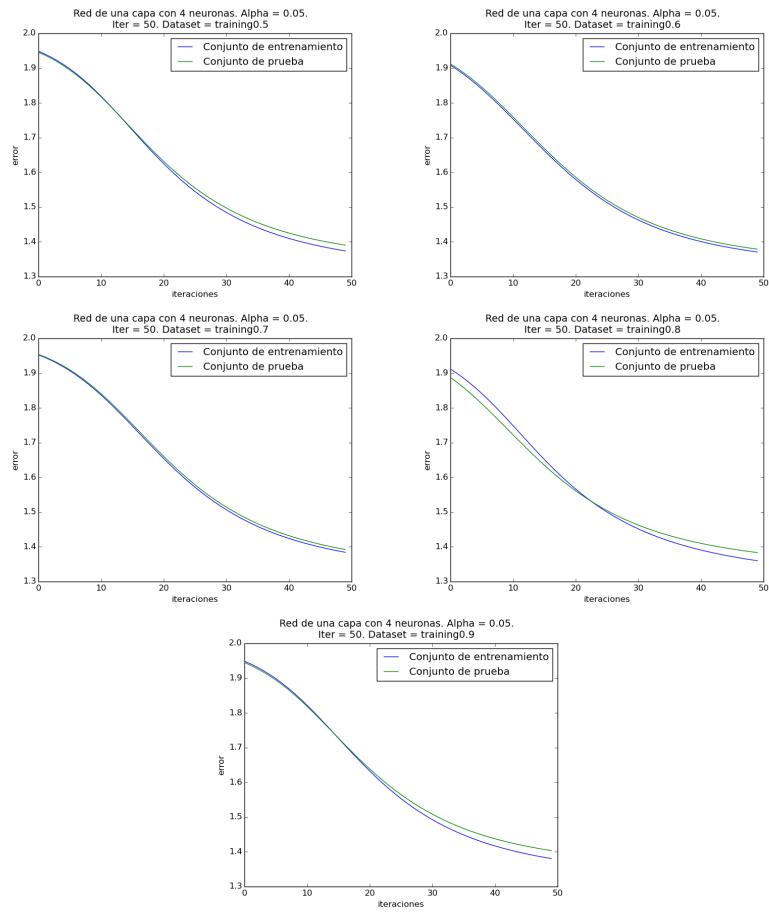


- Para 5 neuronas en 2 capas

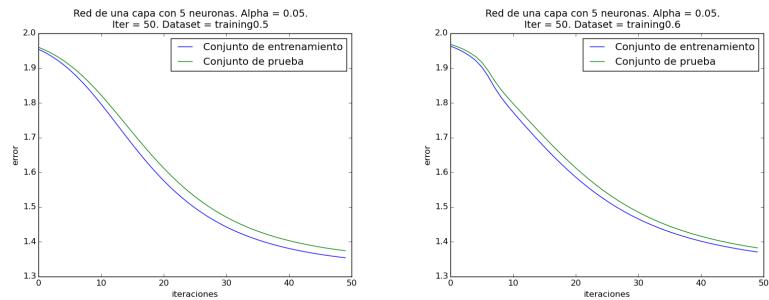


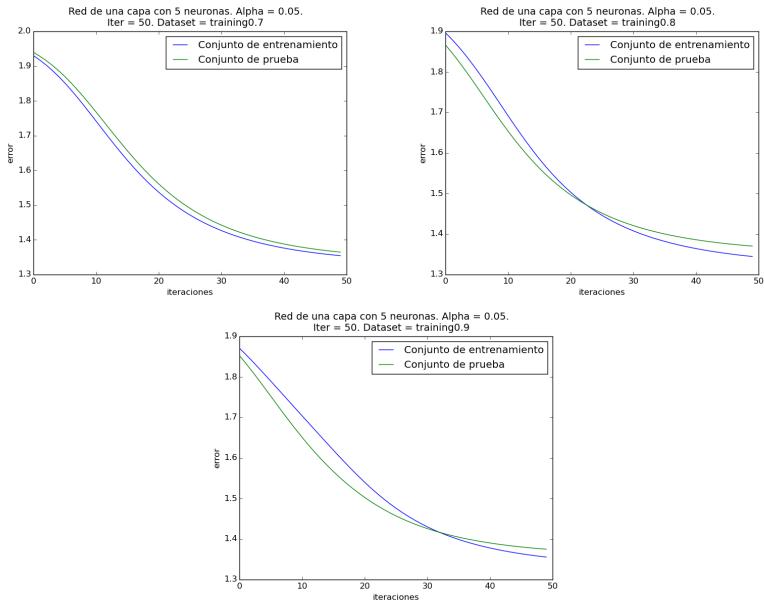
### Clasificador de 3 clases

- Para 4 neuronas en 1 capas

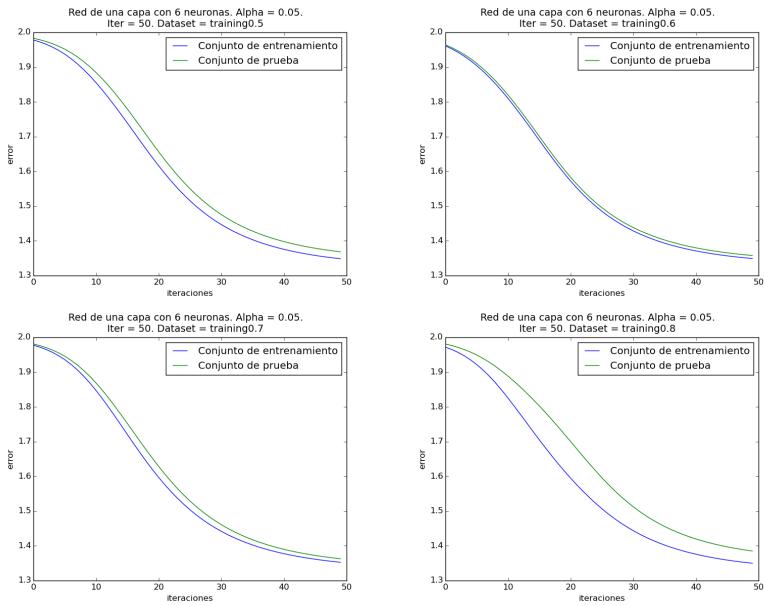


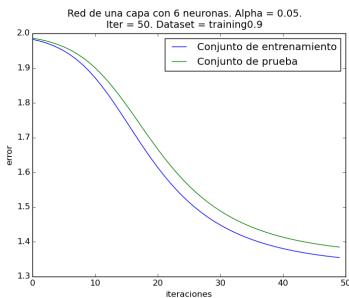
- Para 5 neuronas en 1 capas



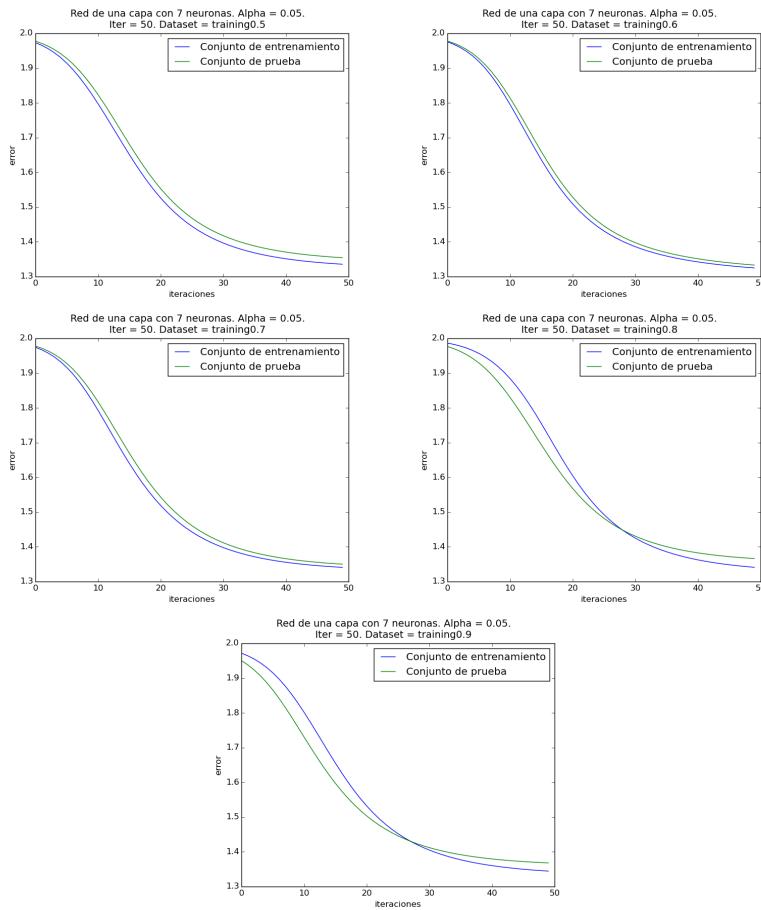


- Para 6 neuronas en 1 capas

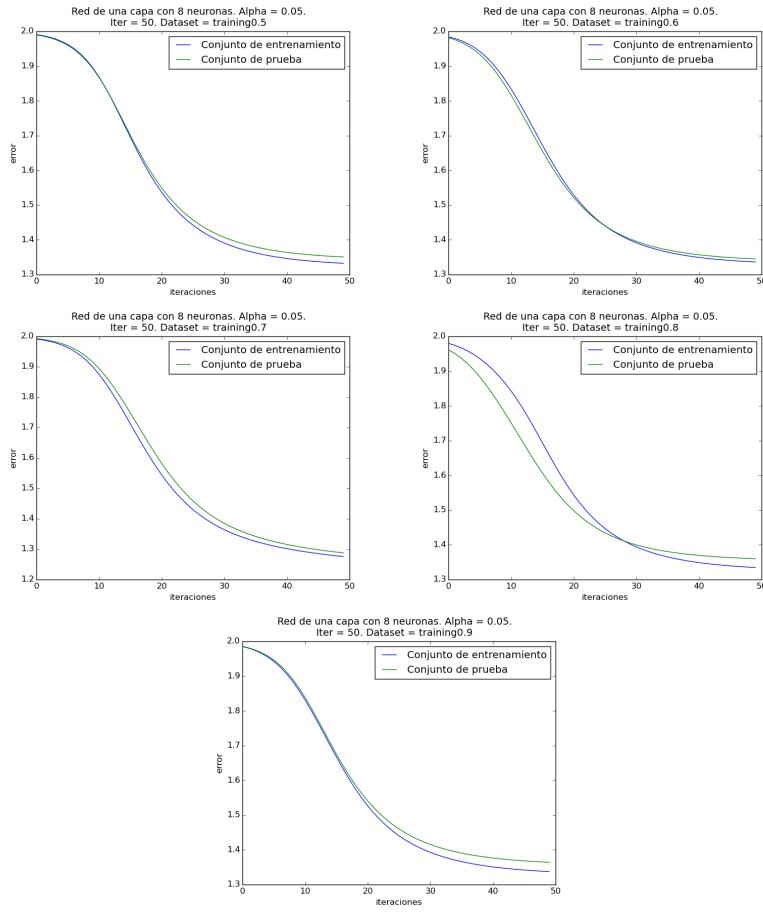




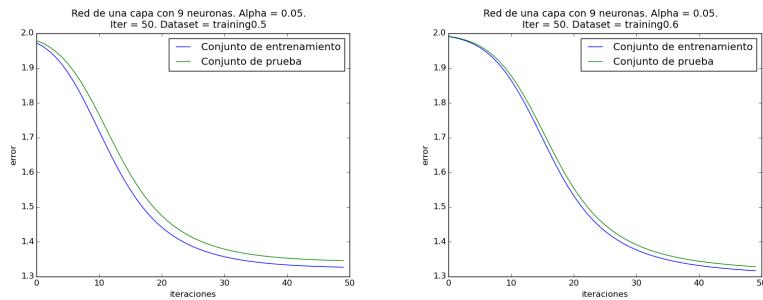
- Para 7 neuronas en 1 capas

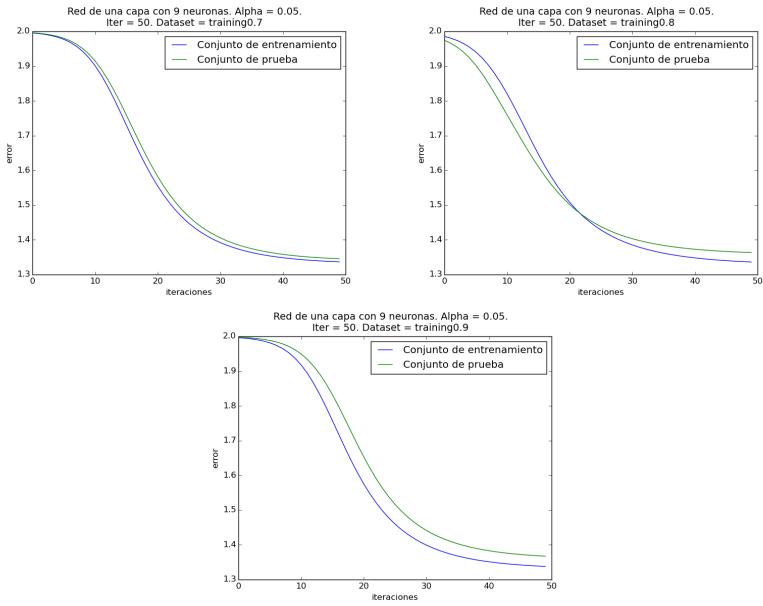


- Para 8 neuronas en 1 capas

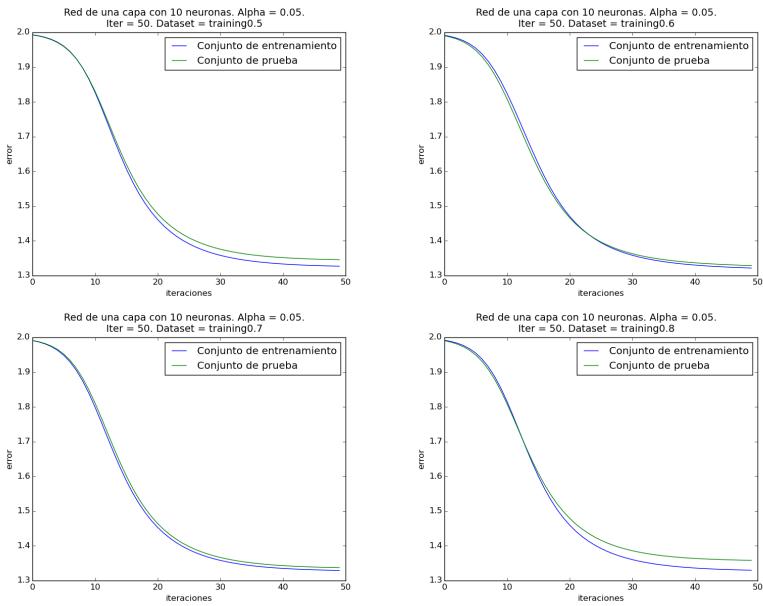


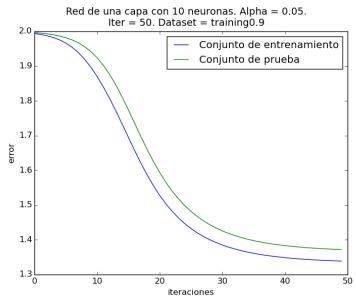
- Para 9 neuronas en 1 capas



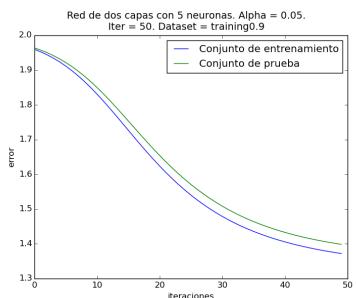
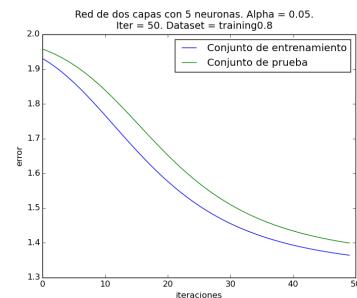
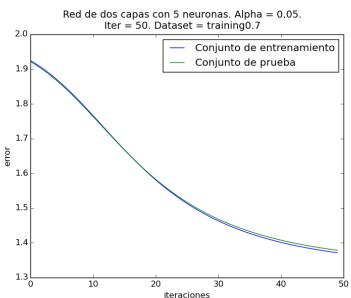
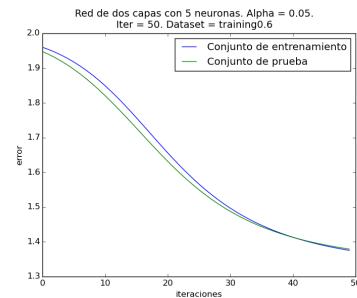
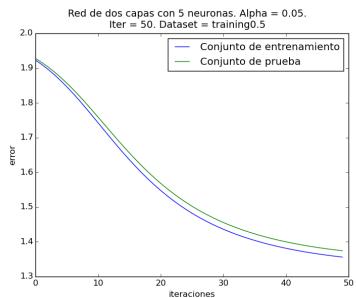


- Para 10 neuronas en 1 capas





- Para 5 neuronas en 2 capas



## Conclusiones

Como se puede apreciar en los resultados la red entrenada para los diferentes conjuntos de entrenamiento posee un error muy bajo con un alto grado de presicion las variaciones en el conjunto entre el numero de neuronas varia en las primeras iteraciones de manera abrupta pero a lo largo de las mismas tienden a valores similares, lo cual permite a cualquiera de las redes tener un resultado optimo para los valores que se intentan predecir.

En el dataset Iris se puede apreciar como los conjuntos se pueden clasificar de diferente forma y como algunos de los valores adoptan una nueva clasificacion cuando se toma en cuenta los 3 categorizaciones.

Se puede ver como en una red de mayor numero de capas el aprendizaje suele ser mas aletardado a lo largo de las iteraciones debido al numero de calculos que se deben realizar en cada una de las neuronas y como el hecho de no embeber la informacion de manera mas directa genera resultados que aun siendo menos eficientes son mas precisos.

La virtud de una red neuronal que pueda clasificar con un alto indice de precision permite que se pueda predecir resultados a futuro con calidad optima.