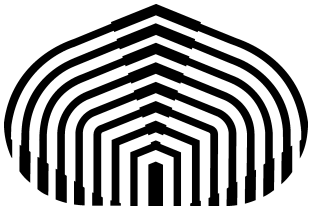


UNIVERSIDAD SIMÓN BOLÍVAR
INGENIERÍA DE LA COMPUTACIÓN
DEPT.COMPUTACIÓN Y TECNOLOGÍA DE LA INFORMACIÓN
CI5438: INTELIGENCIA ARTIFICIAL 2

Proyecto Redes Neuronales

PROFESOR:

Ivette Carolina Martinez

ALUMNOS:

Miguel C. Canedo R.	13-10214
José D. Bracuto D.	13-10173
Rafael A. Cisneros C.	13-11156

Índice

1. Resumen **3**

2. Detalles de implementación **3**
2.1. Ejecución 4
2.2. Función Sigmoidal 4
2.3. Red Neuronal 4
2.4. Elaboración de gráficas 5

3. Presentación y Discusión de Resultados **7**
3.1. Ejercicio 2 7
3.1.1. Escogencia de Mejor Tasa de Aprendizaje 7
3.1.2. Entrenando la red con 1000 datos 8
3.1.3. Entrenando la red con 500 datos 9
3.1.4. Usando de 2 a 10 Neuronas 11
3.1.5. Variando los radios 17
3.2. Ejercicio 3 21
3.2.1. Clasificador Binario 21
3.2.2. Clasificador Multiclase 34

4. Conclusiones **49**

5. Referencias **49**

Resumen

Para este proyecto se va a realizar una red neural feedforward multicapa que haga uso del algoritmo de backpropagation para realizarse varias pruebas usando dos tipos de dataset Primero se trabajara en un dataset que describe un circulo y la intención es probar con un rede de 8 neuronas usando 2000 entradas para escoger la mejor tasa la cual es este caso fue 0.001 Luego se procederá a probar con redes de 2 a 10 neuronas usando la tasa antes seleccionada para demostrar que a más neuronas se usa, más exacto se vuelve la red. Seguidamente se prueba a cambiar los radios del circulo que se está analizando, lo cual nos da como resultado que mientras más pequeño sea la zona de clasificación, la red va a necesitar más neuronas para poder clasificar bien los datos, por ende mientras más grande sea la zona de clasificación, más fácil es clasificar los datos. Ahora bien se pasara a trabajar con un dataset que describe tres tipos de pétalos según una serie de atributos. En este caso vamos a ver que tan eficiente puede ser una clasificación binaria (buscar un solo tipo de pétalo) en comparación a una clasificación muticlase (diferenciar los tres tipos de pétalos) Para esto primero se prueba con redes de 4 a 10 neuronas y variando en cada caso el porcentaje de data que se usa para entrenar la red. Al igual que usando el dataset de los círculos, se puede ver que mientras más neuronas tenga la red, más exacto serán los resultados, aunque en ciertos casos no se note una mejora significativa, siempre va a haber una mejora. A cuanto a la comparación de clasificación binaria y multiclase, se podrá ver que al momento de extraer un solo tipo de pétalo, la red es más exacta mientras que al diferencias los tres tipos tiene una menor exactitud pero sigue siendo viable, sobre todo si se usa una red con más neuronas. También es importante destacar que aunque al usarse una mayor cantidad de datos como entrenamiento mejor la rapidez de y exactitud de ciertos casos, también es cierto que en su contraparte, la red se vuelve más susceptible a sufrir un overfit, por lo que un rango confiable de data para ser usada como entrenamiento es entre 60 % y 70 % de dicha data para que la red sea eficiente sin correr riesgo de sufrir un overfit.

Detalles de implementación

Para la realización del proyecto se uso Python como lenguaje de programación puesto a que ya se tiene experiencia con el mismo, además de poseer tres librerías que destacan al momento de manejar los datasets, estas son:

- **Pandas:** nos ayuda con la lectura y manejo de los datos para poder manipularlos a conveniencia y poder analizarlos.
- **Numpy:** librería que facilita el manejo de matrices, además de contar con varias funciones matemáticas para las operaciones entre arreglos y cálculos necesarios.

- **Matplotlib:** se usa para crear los gráficos de los datos.
- **Sklearn:** Usada para separar los datos.

Ejecución

Para ejecutar los ejercicios se dividieron en un script principal que contiene la definición de la clase RedNeural y sus diferentes métodos, **redNeural.py**). Así mismo, se incluyen 2 scripts para la ejecución de los distintos ejercicios:

- **Ejercicio 2:** para ejecutarlo se tendrá que correr el siguiente comando:

```
$ python ejercicio2.py
```

- **Ejercicio 3:** para ejecutarlo se tendrá que correr el siguiente comando:

```
$ python ejercicio3.py
```

Función Sigmoidal

Es una función lógica que destaca por tener una curva de aprendizaje relativamente acelerada, sin importar el rango de las entradas, sus salidas están comprendidas entre el 0 y el 1, representando así a la salida 0.5 como punto de quiebre el cual se decidirá el ajuste que sera necesario según el valor dado. Esta función es comúnmente usada puesto que puede acomodar cualquier tipo de señal sin producir algún tipo de saturación, además de poseer una derivada sencilla que se comporta uniformemente.

Red Neuronal

Se creo una clase **RedNeural** la cual posee como atributos:

- **data:** es donde se guarda el dataset con el que se trabajara, al cual se le agrega un feature correspondiente al sesgo que sera igual a 1.
- **target:** los resultados esperados para la sección de entrenamiento.
- **clases:** las clases que clasificara la red.
- **num_feat/inter/out:** representa la cantidad de neuronas que tiene la capa de entrada, la capa intermedia y la capa de salida respectivamente.

- **w_feats/inter:** son arreglos que representan los pesos de cada capa, si se esta inicializando la red, estos pesos se escogerán de manera aleatoria.
- **activacion:** es donde se llevaran los cálculos matriciales del proceso de activación usando la función sigmoidal con los pesos como parámetros.

A esta clase se le agregaron las siguientes funciones:

- **_sigmoid:** función de activación.
- **_der_sigmoid:** derivada de la función sigmoidal que sirve para calcular los deltas, para luego ajustar los pesos.
- **propagate:** proceso en el que se activa la neurona y se calculan las salidas con la función sigmoidal.
- **backpropagate:** función en la que se calcula el error obtenido, se calculan los delta usando la derivada de la función sigmoidal y seguidamente se ajustan los pesos según la tasa de aprendizaje dada.
- **train:** iteración por épocas en donde se entrena a la red neuronal usando **propagate** y **backpropagate** y se va calculando como se van comportando los errores que se van obteniendo.
- **forward:** proceso en el que se avanza capa por capa activando las neuronas y calculando sus salidas.
- **save_w:** función que guarda los pesos en un archivo.

Elaboración de gráficas

Se utilizo matplotlib para graficar las convergencia de los errores

```
fig, ax = plt.subplots()
ax.plot(errores)
ax.set_ylim((0, 1))
plt.xlabel('Epocas')
plt.ylabel('Proporcion de elementos mal clasificados')
plt.title("Historial de Errores con %s Neuronas"%n)
plt.show()
```

Además se imprimen las métricas de evaluación de train y test para llevar un tracking con el que se pueda hacer un mejor análisis de la corridas

```
print("TRAIN")
print("-----------------------------------------")
acc, falsos_positivos, falsos_negativos = metricas_eval(target_train, RN.forward(data_train))
print("Accuracy: ", acc)
print("Falsos Positivos:", falsos_positivos)
print("Falsos Negativos:", falsos_negativos)

print("TEST")
print("-----------------------------------------")
acc, falsos_positivos, falsos_negativos = metricas_eval(target_test, RN.forward(data_test))
print("Accuracy: ", acc)
print("Falsos Positivos:", falsos_positivos)
print("Falsos Negativos:", falsos_negativos)
```

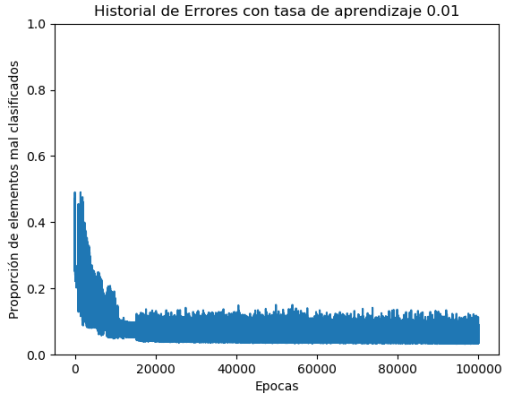
Presentación y Discusión de Resultados

Ejercicio 2

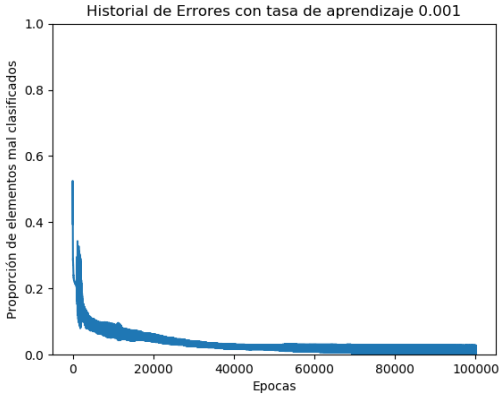
Escogencia de Mejor Tasa de Aprendizaje

Para escoger la mejor Tasa de Aprendizaje se construye una Red con una capa oculta de 8 neuronas y 2000 entradas, y se prueban 4 tasas distintas para ser comparadas entre ellas:

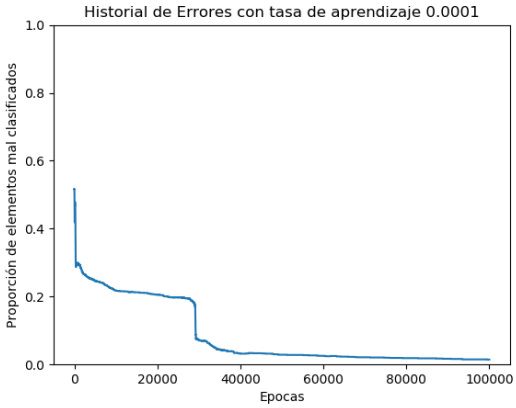
Curvas de convergencia (Error vs. Épocas) para 2000 entradas



Tasa 0.01



Tasa 0.001



Tasa 0.0001

Tasa 0.00001

Métricas de Evaluación

	Accuracy		Falsos Positivos		Falsos Negativos	
Tasas	**Train**	**Test**	**Train**	**Test**	**Train**	**Test**
0.01	0.91	0.895	0	0	144	42
0.001	0.997	0.988	4	4	1	1
0.0001	0.987	0.978	2	0	21	9
0.00001	0.904	0.92	55	8	98	24

Al estudiar la tabla anterior, se aprecia que las tasas 0.001 y 0.0001 son las mejores entre las 4 estudiadas ya que posee un Accuracy considerablemente alto. Pero al estudiar las gráficas de ambas, se aprecia que la Tasa 0.001 es mucho mas inestable mientras converge en comparación con la Tasa 0.0001. Se destaca que la tasa 0.01 es descartada por la notable inestabilidad en su convergencia y a su vez la tasa 0.0001 posee un Accuracy menor debido a que converge mas lento en comparación a las demás.

Por tanto, se determina que la Tasa de aprendizaje 0.0001 podría ser la mejor tasa para este estudio.

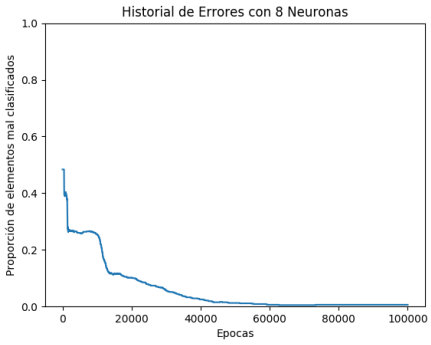
Entrenando la red con 1000 datos

Se realizaron 10 corridas del algoritmo usando como tasa de aprendizaje la obtenida en el apartado anterior (0.0001), y usando como datos los encontrados en el archivo "datos_P2_EM2019_N1500.txt"

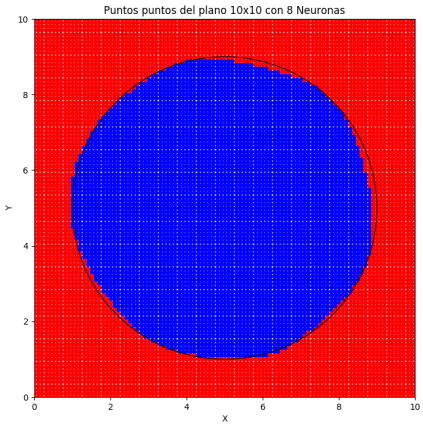
Métricas de Evaluación

	Accuracy		Falsos Positivos		Falsos Negativos	
Corrida	**Train**	**Test**	**Train**	**Test**	**Train**	**Test**
1	0.9825	0.97	1	1	13	5
2	0.98	0.99	2	0	14	2
3	0.98375	0.98	3	3	10	1
4	0.99	0.98	1	0	7	4
5	0.98875	0.97	1	2	8	4
6	0.99125	0.99	0	1	7	1
7	0.99	0.99	1	0	7	2
8	0.98625	0.97	1	1	10	5
9	0.99	0.985	0	1	8	2
10	0.99375	0.99	0	1	5	1
Promedio	0.987787	0.9815	1	1	8.9	2.7

Observamos que las corridas 2, 6, 7 y 10 obtuvieron la mejor precisión en el conjunto de prueba con 0.99, siendo de estos la corrida 10 la que obtuvo la mejor precisión en entrenamiento con 0.99375. En general, con esta cantidad de datos, se obtiene un muy buen modelo de predicción. A continuación mostramos la curva de convergencia para la corrida 10, así como la prueba realizada usando un nuevo conjunto de pruebas de un barrido 100x100 de la región cuadrada, observando el circulo de clasificación.



Curva de Convergencia



Circulo de prueba

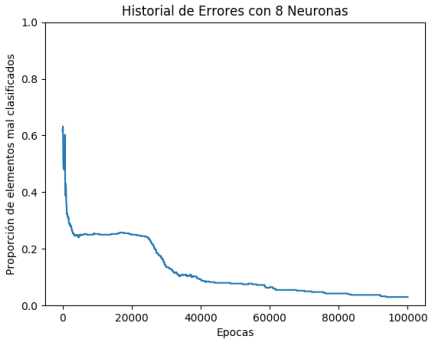
Entrenando la red con 500 datos

Se realizaron 10 corridas del algoritmo usando como tasa de aprendizaje la obtenida en el apartado anterior (0.0001), y usando como datos los encontrados en el archivo "datos_P2_EM2019_N500.txt"

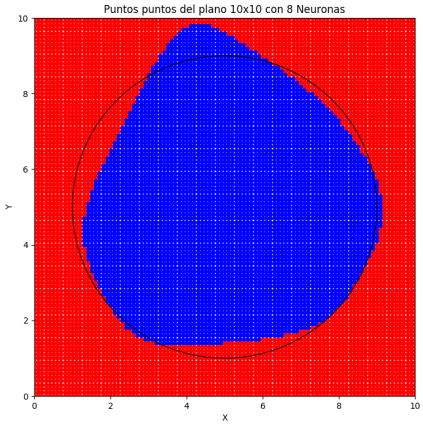
Métricas de Evaluación

	Accuracy		Falsos Positivos		Falsos Negativos	
Corrida	**Train**	**Test**	**Train**	**Test**	**Train**	**Test**
1	0.985	0.97	3	1	3	2
2	0.96	0.93	4	3	12	4
3	0.8125	0.82	65	13	10	5
4	0.9625	0.96	6	2	9	2
5	0.7975	0.78	69	19	12	3
6	0.8175	0.81	62	15	11	4
7	0.9175	0.9	17	4	16	6
8	0.8075	0.8	66	18	11	2
9	0.97	0.98	3	1	9	1
10	0.9525	0.92	8	1	11	5
Promedio	0.89225	0.887	30.3	7,7	10.4	3.4

Observamos que la 9 obtuvo la mejor precisión en el conjunto de prueba con 0.98, y en el conjunto de entrenamiento la mejor corrida fue la 1, sin embargo observamos un promedio menor al obtenido realizando el modelo con 1000 datos, lo que nos dice que usando un poco mas de datos podemos generar un mejor modelo, sin caer en overfitting. A continuación se muestra la curva de convergencia para la corrida 9, así como la prueba realizada usando un nuevo conjunto de pruebas de un barrido 100x100 de la región cuadrada, observando el circulo de clasificación:



Curva de Convergencia



Circulo de prueba

Usando de 2 a 10 Neuronas

Al analizar usando redes de 2 a 10 neuronas podemos ver que mientras mas neuronas se usan mas exacto se vuelve la red pero siempre se llega a un punto en donde la mejora empieza a no ser tan significativa, en este caso ese caso se evidencia con los resultados obtenidos a partir de usar 8 neuronas, todos los resultados posteriores a usar 8 neuronas van a mejorar de forma tan pequeña que no se pude notar la diferencia, por esto es que 8 neuronas es la cantidad óptima para este caso.

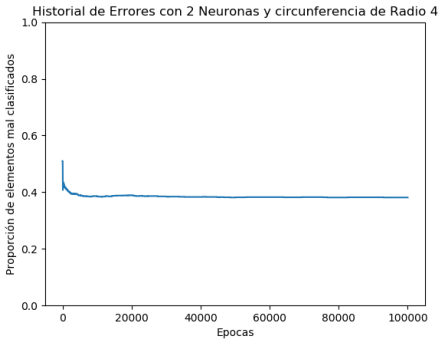
Métricas de Evaluación

	Accuracy		**Falsos Positivos**		**Falsos Negativos**	
N.º Neuronas	**Train**	**Test**	**Train**	**Test**	**Train**	**Test**
2	0.7390625	0.7125	367	102.9	50.5	12.1
3	0.842125	0.8665	181	35.2	71.6	18.2
4	0.8604	0.8615	161.6	39.6	61.8	15.8
5	0.9804	0.9767	8.33	7.67	23	7.67
6	0.98	0.9717	6.67	2.67	25.33	8.67
7	0.9777	0.9742	21	6.67	14.67	3.67
8	0.9869	0.985	9	3	12	3
9	0.9923	0.9933	4.33	1.33	8	1.33
10	0.9944	0.9908	1.33	2	7.67	1.67

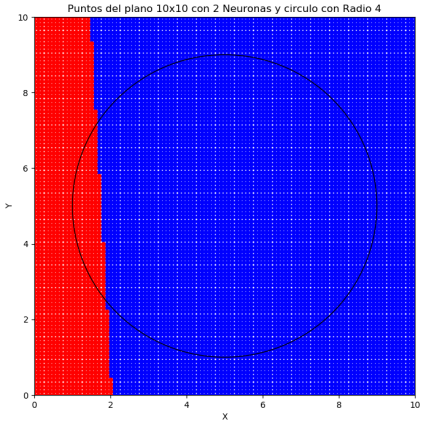
A continuación se observa las curvas de convergencia y las gráficas de prueba.

- **2 Neuronas**

  Se puede ver que con solo 2 neuronas el algoritmo no tiene la posibilidad de clasificar bien los datos, tiene un límite de convergencia el cual da como resultado que los datos sean clasificados en dos zonas divididas verticalmente como se puede observar en el gráfico.



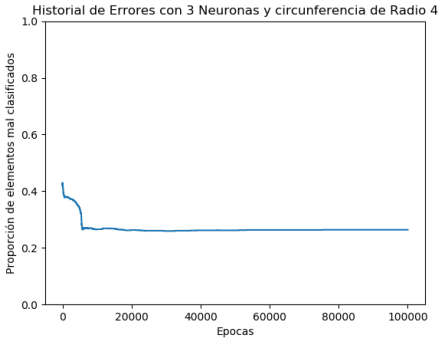
Curva de Convergencia



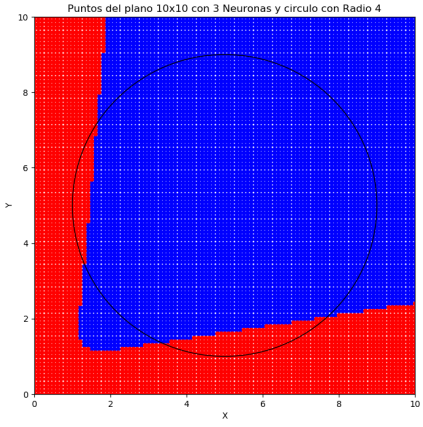
Circulo de prueba

- **3 Neuronas**

  Se puede ver que con 3 neuronas mejora la clasificación pero todavía no se puede delimitar el círculo puesto que el nivel de convergencia sigue siendo alto y separa las zonas aislando solo la zona superior derecha



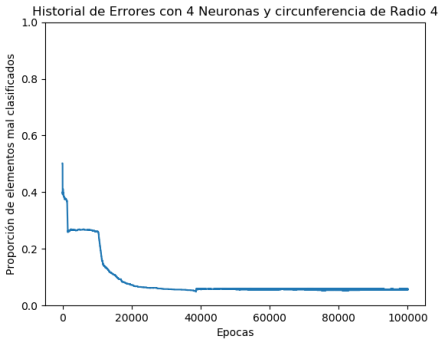
Curva de Convergencia



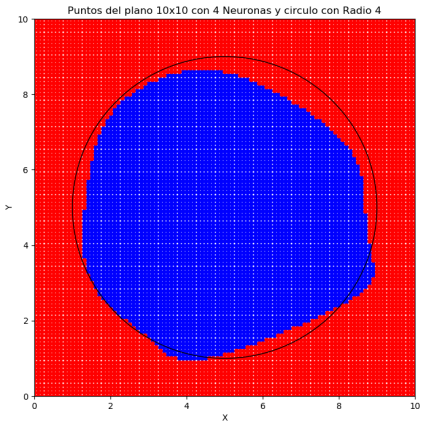
Circulo de prueba

- **4 Neuronas**

  A partir de 4 neuronas podemos ver que empieza a clasificar bien los datos que están en la región dentro del círculo, también se puede decir que es más probable que ocurra un overfitting en la red y por lo mismo sus resultados no son del todo correctos



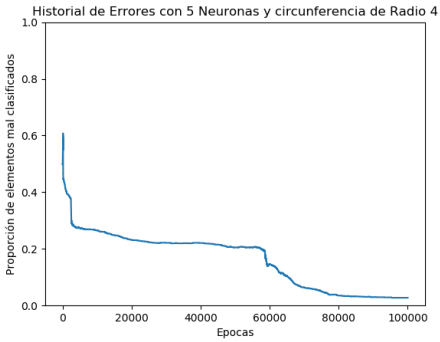
Curva de Convergencia



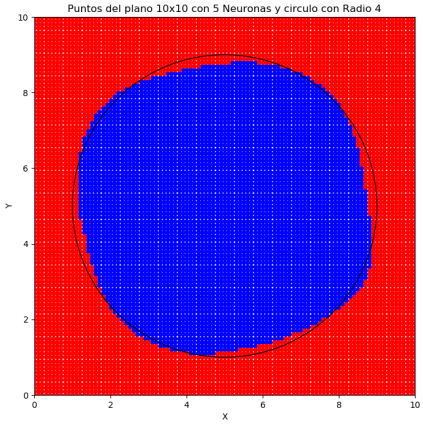
Circulo de prueba

- **5 Neuronas**

  Con 5 neuronas, el proceso de convergencia sigue una distribución irregular y llega un momento en que tarda en mejorar, si no es hasta las 60000 épocas que empieza a mejorar hasta converger



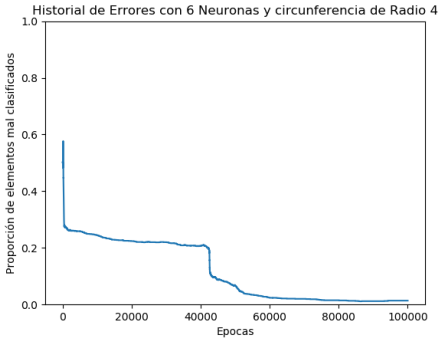
Curva de Convergencia



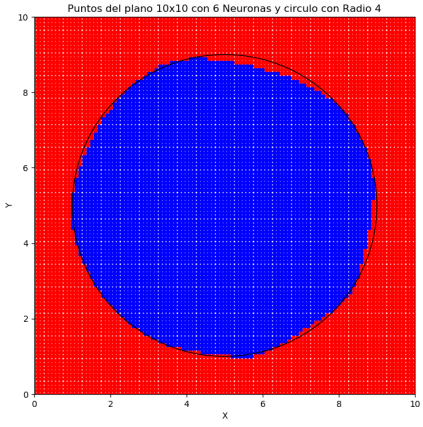
Circulo de prueba

- **6 Neuronas**

  Al igual que con 5 neuronas, en este caso la única mejor es que tarda menos en mejorar el error (en la época 40000 empieza a mejorar de forma regular) y por ende tarda menos en converger



Curva de Convergencia

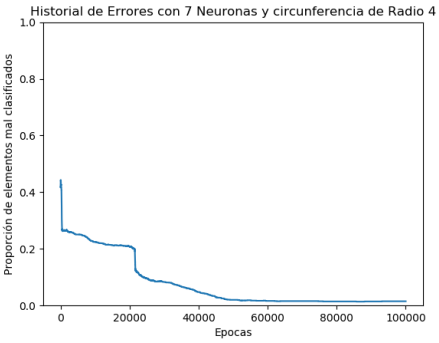


Circulo de prueba

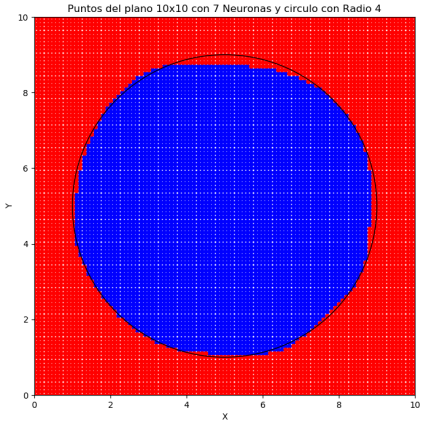
- **7 Neuronas**

  Con 7 neuronas tarda mucho menos en mejorar, en este caso a las 20000 épocas empieza el proceso

Curvas de convergencia (Error vs. Épocas) para 2000 entradas



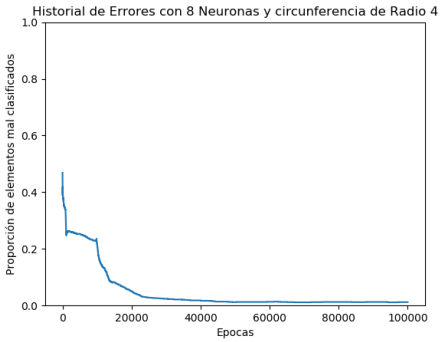
Curva de Convergencia



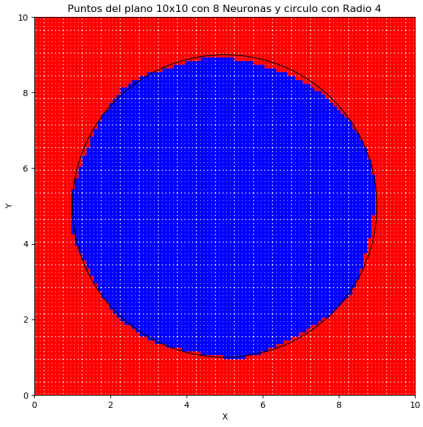
Circulo de prueba

- **8 Neuronas**

  Vemos como la red empieza a adaptarse más rápido a los datos pero también se puede ver qué ocurre ciertas irregularidades al momento de mejorar los errores que generan picos, puesto a que se encuentra con un valor el cual obliga a que el error aumente en vez de disminuir



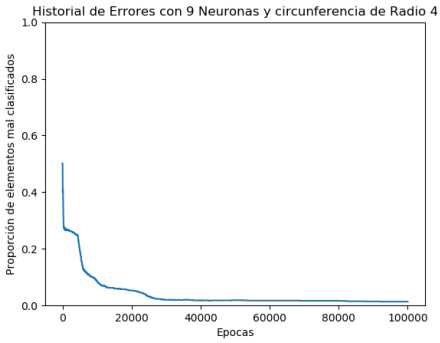
Curva de Convergencia



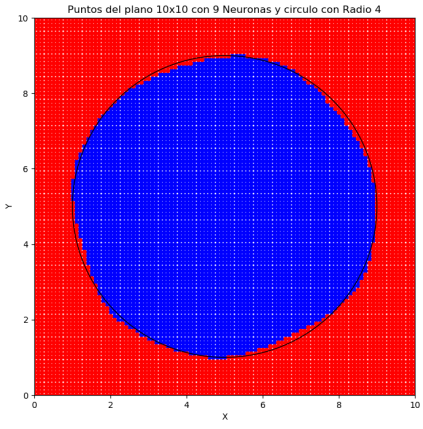
Circulo de prueba

- **9 Neuronas**

  Ya se puede ver que no se genera ningún tipo de irregularidad en cuanto a la convergencia, por lo que podemos decir que a partir de 9 neuronas la red se comporta de manera regular y tiene una buena tasa de convergencia, también podemos ver que la gráfica de la clasificación está bien delimitada.



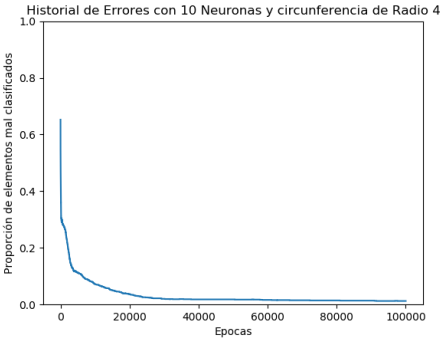
Curva de Convergencia



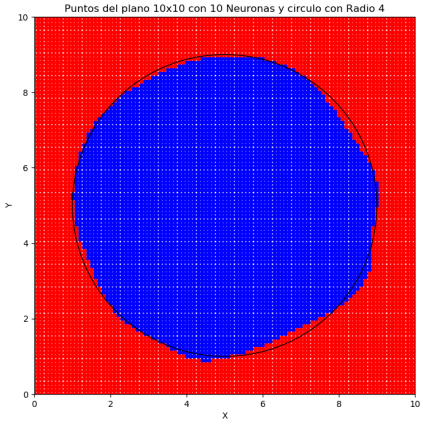
Circulo de prueba

- **10 Neuronas**

  Como se dijo anteriormente, a partir de 9 neuronas, el proceso de convergencia se da de manera regular sin ninguna falla. Así que con 10 neuronas podemos ver que lo que se va es perfeccionando las predicciones sin alterarse el proceso de convergencia, es decir, a partir de 9 neuronas es que convergen de manera confiable y mientras más neuronas se usen, más exacto y rápido será este proceso.



Curva de Convergencia



Circulo de prueba

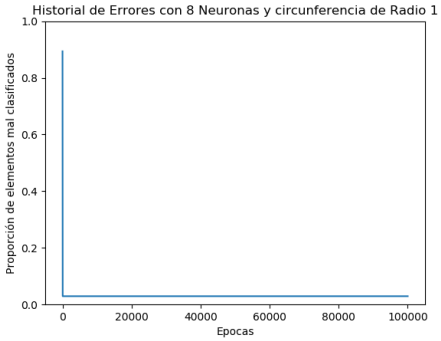
Variando los radios

- **Radio 1**

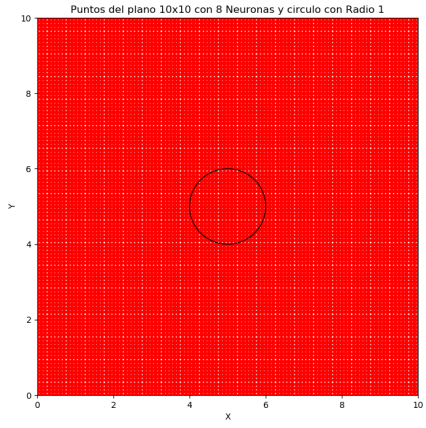
Métricas de Evaluación

	Accuracy		Falsos Positivos		Falsos Negativos	
Corrida	**Train**	**Test**	**Train**	**Test**	**Train**	**Test**
1	0.970625	0.95	0	0	47	20
2	0.970625	0.95	0	0	47	20
3	0.970625	0.95	0	0	47	20
4	0.970625	0.95	0	0	47	20
5	0.970625	0.95	0	0	47	20
6	0.970625	0.95	0	0	47	20
7	0.970625	0.95	0	0	47	20
8	0.970625	0.95	0	0	47	20
9	0.970625	0.95	0	0	47	20
10	0.970625	0.95	0	0	47	20
Promedio	0.970625	0.95	0	0	47	20

Mejor Corrida Cualquiera (?)



Curva de Convergencia



Circulo de prueba

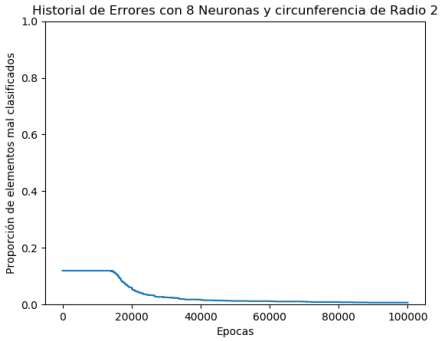
Se puede ver que aunque nos arroja un Accuracy de 0.97 y 0.95 en el entrenamiento y en las pruebas respectivamente, al memento de corroborar los resultados se evidencia que la mayoría son falsos negativos y en realidad no se clasifica bien con radio 1 puesto a que es una zona muy pequeña.

- **Radio 2**

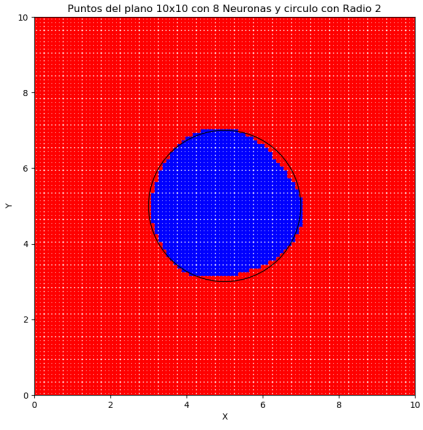
Métricas de Evaluación

	Accuracy		Falsos Positivos		Falsos Negativos	
Corrida	**Train**	**Test**	**Train**	**Test**	**Train**	**Test**
1	0.988125	0.99	3	2	16	2
2	0.990625	0.99	4	2	11	2
3	0.990625	0.9925	4	1	11	2
4	0.990625	0.99	4	2	11	2
5	0.99125	0.99	4	2	10	2
6	0.993125	0.9925	2	2	8	1
7	0.880625	0.8825	0	0	191	47
8	0.993125	0.9925	3	2	8	1
9	0.880625	0.8825	0	0	191	47
10	0.9925	0.99	3	2	9	2
Promedio	0.969125	0.96925	2.7	1,5	46.6	10.8

Mejor Corrida Cualquiera 6



Curva de Convergencia



Circulo de prueba

Se puede ver que al aumentar el radio se empieza a clasificar de forma correcta e igualmente el proceso de convergencia se da de forma regular a partir de alas 10000

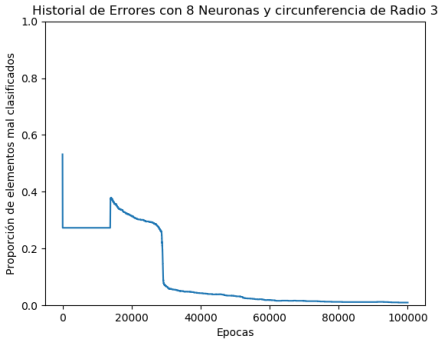
épocas. En cuanto a las corridas se puede decir que todas están con un accuracy mayor a 0.99 tanto en el entrenamiento como en las pruebas, a excepción de la corrida 7 y 9 en donde ocurrieron una cantidad elevada de falsos negativos, pero a pesar de esto se puede decir que por el promedio obtenido, con radio 2, los resultados son clasificados de forma correcta.

- **Radio 3**

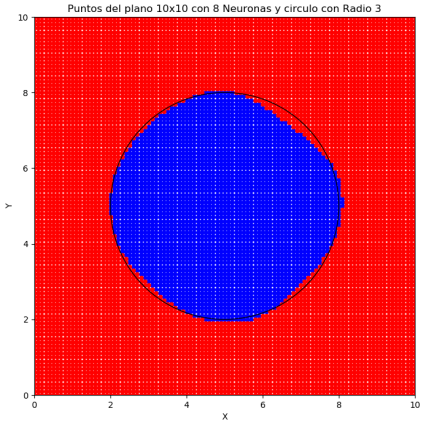
Métricas de Evaluación

	Accuracy		Falsos Positivos		Falsos Negativos	
Corrida	**Train**	**Test**	**Train**	**Test**	**Train**	**Test**
1	0.99	0.982	5	4	11	3
2	0.99	0.99	5	0	11	4
3	0.968125	0.9525	19	12	32	7
4	0.9925	0.98	2	4	10	4
5	0.9625	0.945	17	11	43	11
6	0.96875	0.975	12	2	38	8
7	0.995	0.985	3	4	5	2
8	0.99125	0.985	4	4	10	2
9	0.975	0.9575	13	7	27	10
10	0.98875	0.985	7	2	11	4
Promedio	0.9821	0.9737	8.7	5	19.8	8.5

Mejor Corrida Cualquiera 2



Curva de Convergencia



Circulo de prueba

Aunque se ve que se clasifican bien los datos, también se puede ver que el proceso de convergencia no se da de manera regular, como si en algún momento debido al radio propuesto, la red tuviera problemas en clasificar los datos propuestos, pero a partir de las 15000 épocas lograra encontrar un peso adecuado y así empezar el proceso de mejora de manera regular, por lo que a pesar del mal comienzo, termina clasificando de forma correcta, y esto se puede evidenciar tanto en la gráfica del circulo como en las tablas del promedio de falsos positivo y falsos negativos y tienen valores de 5 y 8.5 respectivamente, lo cual es un valor relativamente bajo que nos demuestra la correctitud de la rede creada.

Ejercicio 3

Para todas las ejecuciones pertinentes a este ejercicio, se entrenaron las redes con 60 épocas y con tasa de aprendizaje de 0.02.

Clasificador Binario

De forma general al analizar las tablas de las métricas podemos ver que al tratarse de un clasificador binario, la mínima exactitud que se obtiene es 0.98 sin presentar falsos positivos ni falsos negativos en cantidades significantes a excepción de cuando se usan 8 neuronas que próximamente se analizara dicho caso.

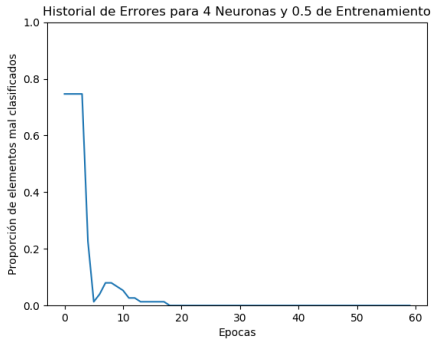
- **4 Neuronas**

  Se puede ver como el proceso de convergencia es más estable usando un 60 %, 70 % y 80 % como conjunto de entrenamiento aunque por usarse solo 4 neurona, al principio le cuesta hacer alguna mejora del error, siendo el caso de 70 % el mejor puesto que tiene una tasas de exactitud de 100 % y es la que converge más rápido

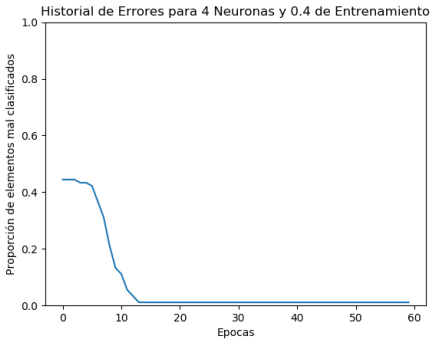
Métricas de Evaluación

	Accuracy		**Falsos Positivos**		**Falsos Negativos**	
% de Train	**Train**	**Test**	**Train**	**Test**	**Train**	**Test**
50 %	1.0	0.9333	0	0.4	0	0.1
60 %	1.0	0.9833	0	0	1	0.1
70 %	1.0	1.0	0	0	0	0
80 %	1.0	1.0	0	0	0	0
90 %	1.0	1.0	0	0	0	0

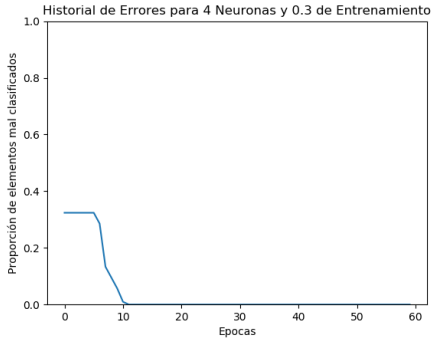
Curvas de convergencia (Error vs. Épocas) para 4 Neuronas



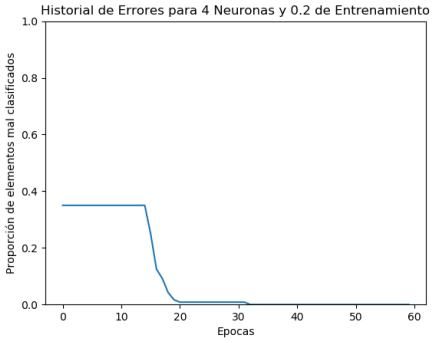
Conjunto de Entrenamiento 50 %



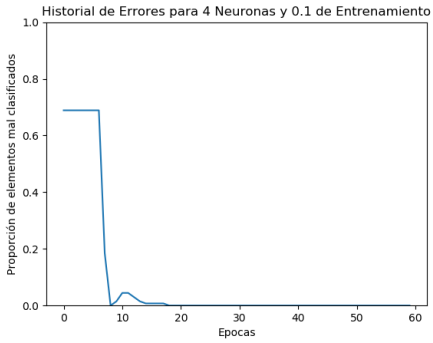
Conjunto de Entrenamiento 60 %



Conjunto de Entrenamiento 70 %



Conjunto de Entrenamiento 80 %



Conjunto de Entrenamiento 90 %

- **5 Neuronas**

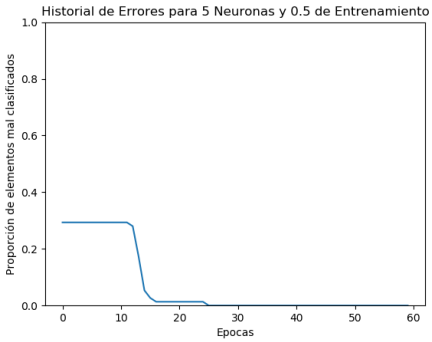
  A partir de usar 60 % de la data como entrenamiento se ve que los resultados son eficientes pero se nota más dicha eficiencia a partir de 80 % puesto a que la mejora del errar encada iteración es más sencilla y convergen rápidamente. Aunque

usando 90 % de la data como entrenamiento es la que tiene una convergencia más regular, se puede decir que usando 80 % de la data es mejor puesto a que llega a los mismos resultados en menos tiempo y con menos datos usados.

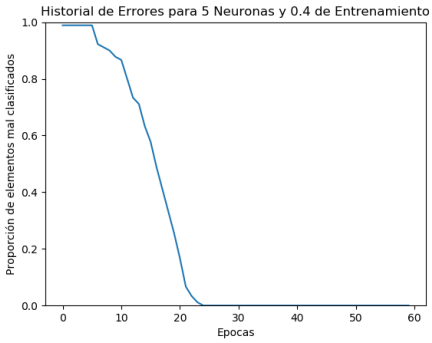
Métricas de Evaluación

	Accuracy		**Falsos Positivos**		**Falsos Negativos**	
% de Train	**Train**	**Test**	**Train**	**Test**	**Train**	**Test**
50 %	1.0	0.9987	0	0	0	0.1
60 %	1.0	1.0	0	0	0	0
70 %	1.0	1.0	0	0	0	0
80 %	1.0	1.0	0	0	0	0
90 %	1.0	1.0	0	0	0	0

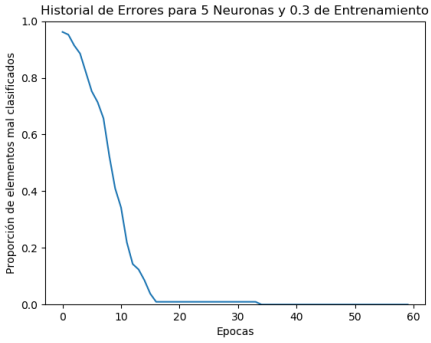
Curvas de convergencia (Error vs. Épocas) para 5 Neuronas



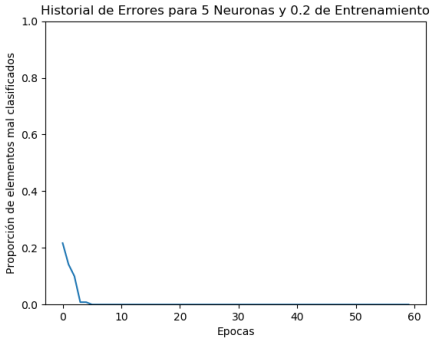
Conjunto de Entrenamiento 50 %



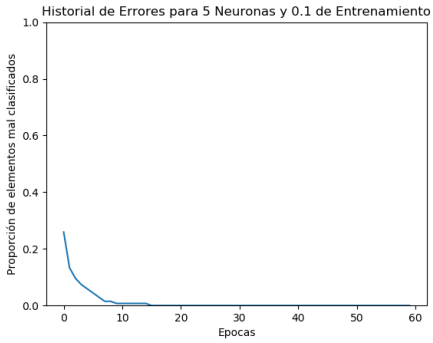
Conjunto de Entrenamiento 60 %



Conjunto de Entrenamiento 70 %



Conjunto de Entrenamiento 80 %



Conjunto de Entrenamiento 90 %

- **6 Neuronas**

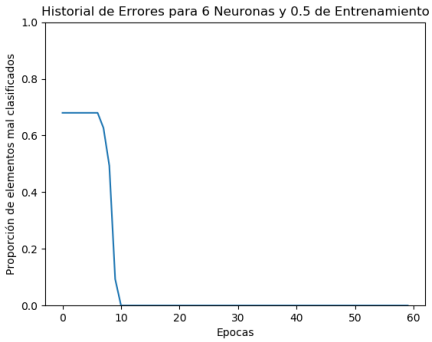
  A partir de 6 neuronas vemos como empieza a disminuir la exactitud de la red pero no es significativa. Además al observar las gráficas de convergencia se ve que al usar 70 % de la data es la mejor puesto a su distribución regular y rápida

convergencia. Representando en las demás una posibilidad elevada de sufrir un overfit si se trabajara con una mayor cantidad

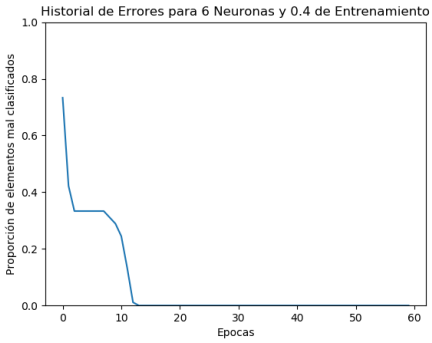
Métricas de Evaluación

	Accuracy		**Falsos Positivos**		**Falsos Negativos**	
% de Train	**Train**	**Test**	**Train**	**Test**	**Train**	**Test**
50 %	1.0	1.0	0	0	0	0
60 %	0.9989	1.0	0	0	0.1	0
70 %	1.0	0.9778	0	0	0	1
80 %	1.0	1.0	0	0	0	0
90 %	1.0	1.0	0	0	0	0

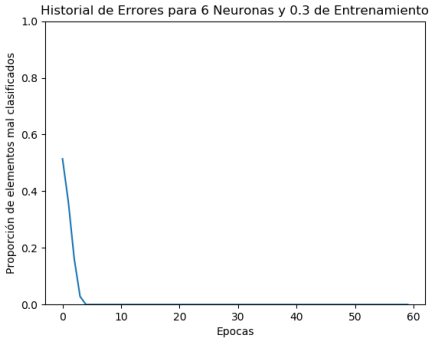
Curvas de convergencia (Error vs. Épocas) para 6 Neuronas



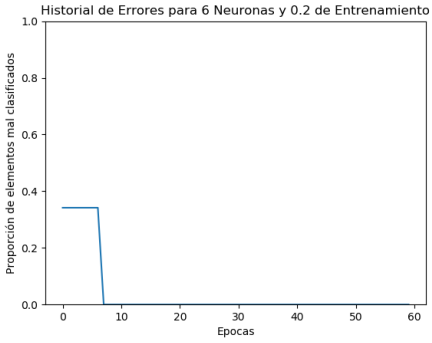
Conjunto de Entrenamiento 50 %



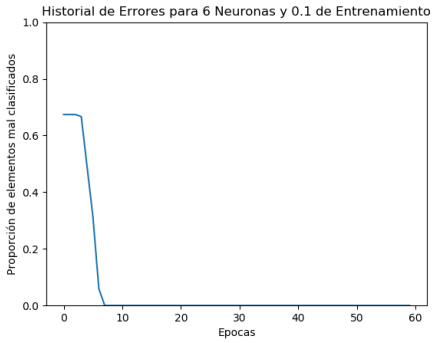
Conjunto de Entrenamiento 60 %



Conjunto de Entrenamiento 70 %



Conjunto de Entrenamiento 80 %



Conjunto de Entrenamiento 90 %

- **7 Neuronas**

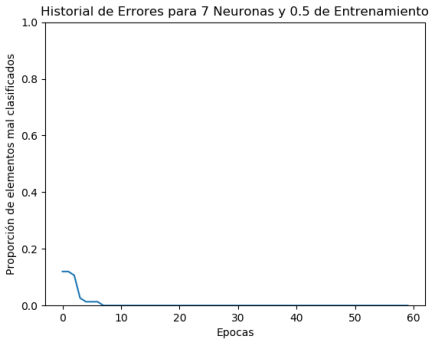
  Al igual que con 6 neuronas, este caso se comporta de la misma manera con la diferencia de que en este se logró una efectiva predicción usando 60 % de los datos como entrenamiento. Esto se debe a que al procesare con más neuronas esta arroja

los mismos resultados con menos necesidad de usar tanta cantidad de datos como entrenamiento

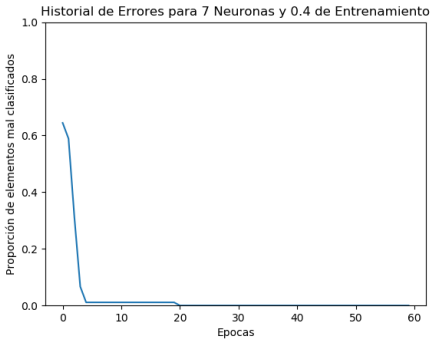
Métricas de Evaluación

	Accuracy		**Falsos Positivos**		**Falsos Negativos**	
% de Train	**Train**	**Test**	**Train**	**Test**	**Train**	**Test**
50 %	1.0	0.9973	0	0	0	0.2
60 %	1.0	1.0	0	0	0	0
70 %	1.0	0.9978	0	0	0	0.1
80 %	1.0	1.0	0	0	0	0
90 %	1.0	1.0	0	0	0	0

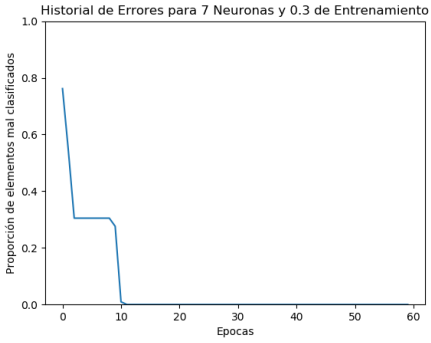
Curvas de convergencia (Error vs. Épocas) para 7 Neuronas



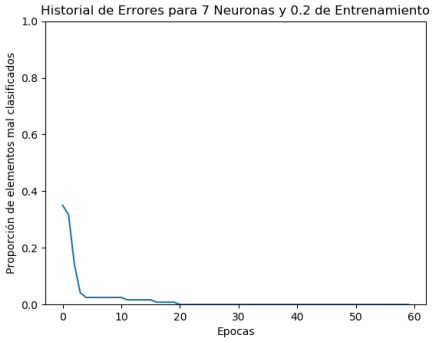
Conjunto de Entrenamiento 50 %



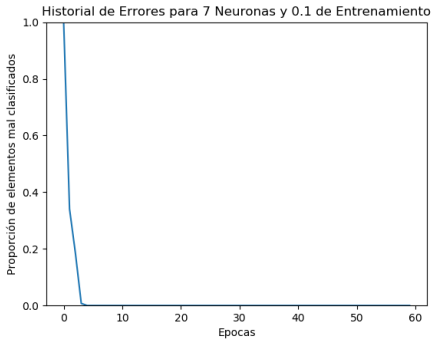
Conjunto de Entrenamiento 60 %



Conjunto de Entrenamiento 70 %



Conjunto de Entrenamiento 80 %



Conjunto de Entrenamiento 90 %

- **8 Neuronas**

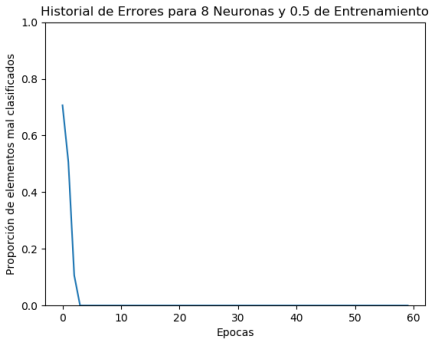
  Este es un caso especial puesto a que tanto la tasas de exactitud como las gráficas de convergencia están más alteradas que las demás, esto puede ser causado por la data que se escogió para ser entrenada y al tener más neuronas, la red sufre mas

variaciones antes de llegar a un resultado final preciso, por lo que a partir de este momento mientras más datos se usen para entrenar la red, más efectivo y rápido será la convergencia.

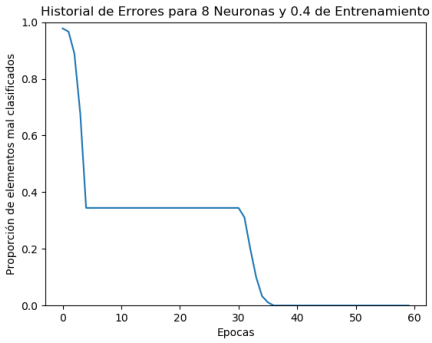
Métricas de Evaluación

	Accuracy		**Falsos Positivos**		**Falsos Negativos**	
% de Train	**Train**	**Test**	**Train**	**Test**	**Train**	**Test**
50 %	0.9853	0.9853	0	0	1.1	1.1
60 %	0.9744	0.955	0	1	2.3	2.7
70 %	1.0	0.9978	0	0	0	0.1
80 %	0.965	0.97	0	0	4.2	0.9
90 %	1.0	1.0	0	0	0	0

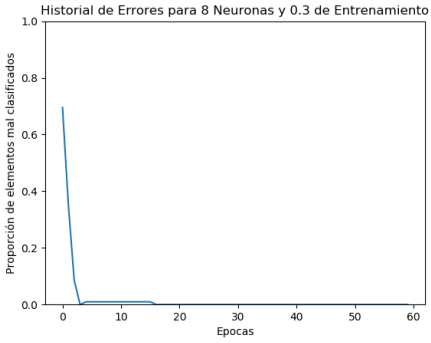
Curvas de convergencia (Error vs. Épocas) para 8 Neuronas



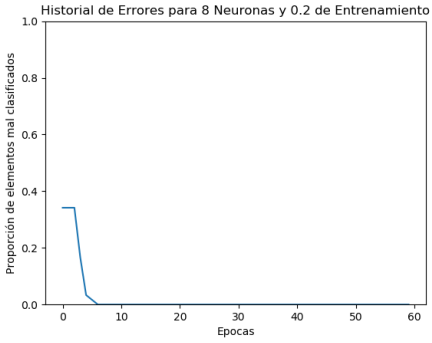
Conjunto de Entrenamiento 50 %



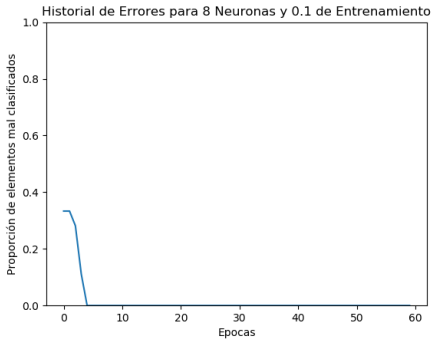
Conjunto de Entrenamiento 60 %



Conjunto de Entrenamiento 70 %



Conjunto de Entrenamiento 80 %



Conjunto de Entrenamiento 90 %

- **9 Neuronas**

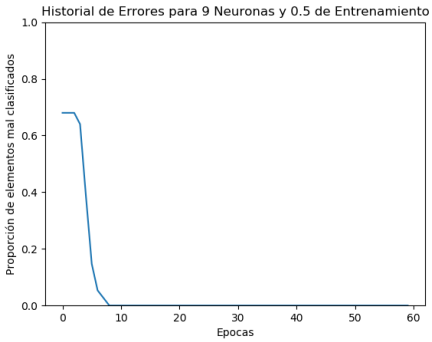
  Se puede ver que con 9 neuronas la red se comporta estable a excepción de cuando se usa el 90 % de los casos, pero ese caso se debe a que en la época 5 se obtuvo un error significativamente alto pero la red vuelve a recuperar la normalidad

rápidamente a pesar de dicho error. También se puede ver que cuando cualquier cantidad de data la red clasifica eficientemente, siendo 60 % la cantidad que causa mejor desempeño.

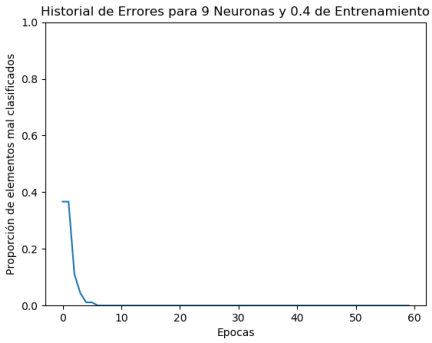
Métricas de Evaluación

	Accuracy		**Falsos Positivos**		**Falsos Negativos**	
% de Train	**Train**	**Test**	**Train**	**Test**	**Train**	**Test**
50 %	1.0	0.9986	0	0	0	0.1
60 %	1.0	1.0	0	0	0	0
70 %	0.9686	0.9667	0	0	3.3	1.5
80 %	1.0	1.0	0	0	0	0
90 %	1.0	1.0	0	0	0	0

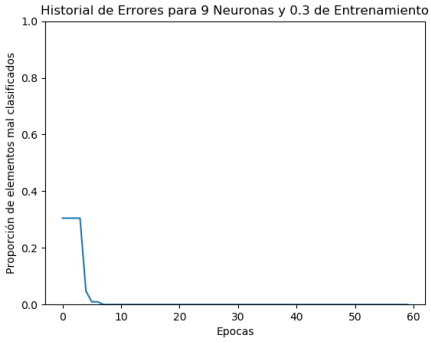
Curvas de convergencia (Error vs. Épocas) para 9 Neuronas



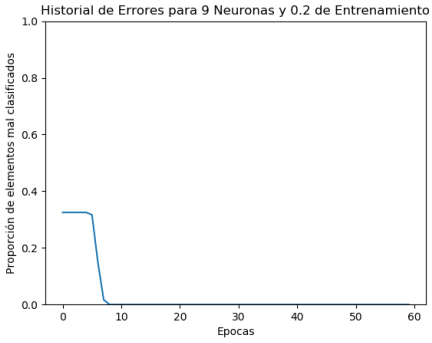
Conjunto de Entrenamiento 50 %



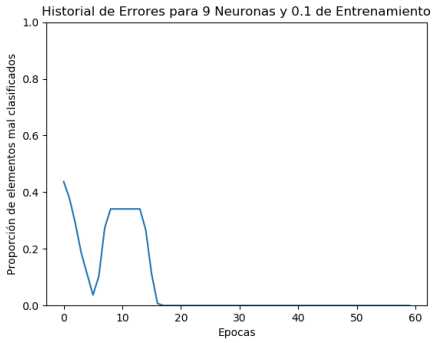
Conjunto de Entrenamiento 60 %



Conjunto de Entrenamiento 70 %



Conjunto de Entrenamiento 80 %



Conjunto de Entrenamiento 90 %

- **10 Neuronas**

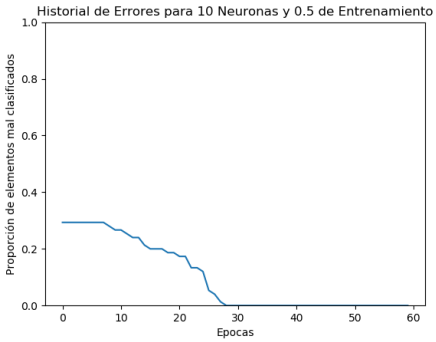
  Al igual que usando 9 neuronas, este caso es eficiente a partir de usar 60 % de la data como entrenamiento y conforme mas data se use mejor va a ser la predicción, aunque la red con 10 neuronas sigue siendo susceptible a errores atípicos, esta tiene

una mayor capacidad de mejora por lo que no importa mucho los posibles errores atípicos que salgan puesto a que la red aprenderá relativamente rápido y llegara a los mismos resultados.

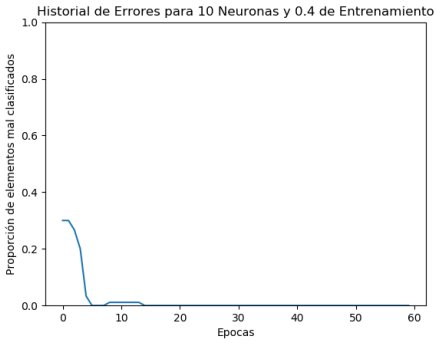
Métricas de Evaluación

	Accuracy		**Falsos Positivos**		**Falsos Negativos**	
% de Train	**Train**	**Test**	**Train**	**Test**	**Train**	**Test**
50 %	1.0	1.0	0	0	0	0
60 %	0.9689	0.96	0	0.2	2.8	2.2
70 %	1.0	1.0	0	0	0	0
80 %	1.0	1.0	0	0	0	0
90 %	1.0	1.0	0	0	0	0

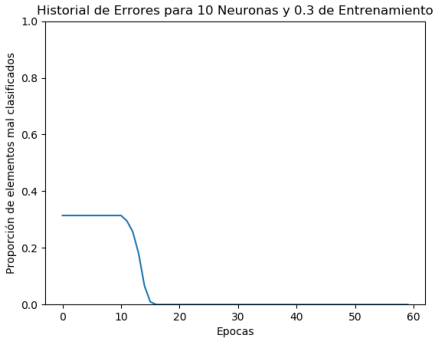
Curvas de convergencia (Error vs. Épocas) para 10 Neuronas



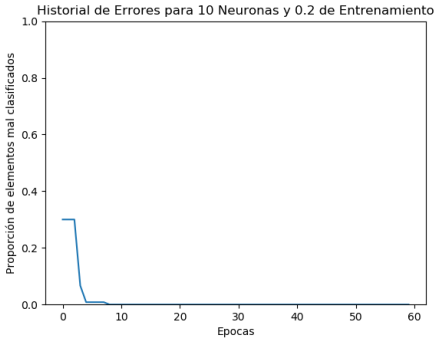
Conjunto de Entrenamiento 50 %



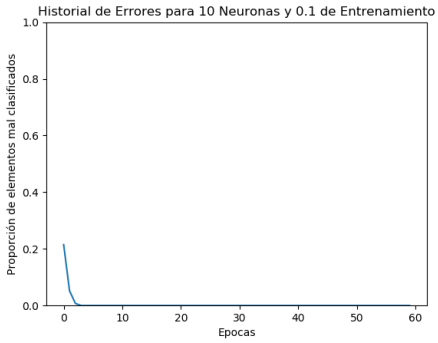
Conjunto de Entrenamiento 60 %



Conjunto de Entrenamiento 70 %



Conjunto de Entrenamiento 80 %



Conjunto de Entrenamiento 90 %

Clasificador Multiclase

A diferencia del clasificador binario, el clasificador multiclase presenta una menor exactitud, pero el proceso de convergencia sigue siendo confiable mientras más neuronas

son usadas, y los únicos problemas que presenta son las variaciones del error con los distintos conjuntos de datos que se usan para entrenar la red.

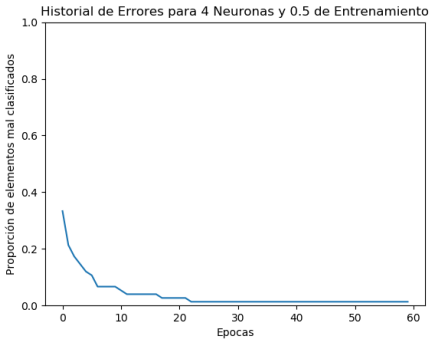
- **4 Neuronas**

  Se puede ver que la menor exactitud es de 0.82 y 0.77 para el entrenamiento y la prueba respectivamente, además se muestra que la predicción más confiable tomando en cuenta los resultados del test y el train es al momento de usar 70 % de la data como entrenamiento, que aunque tarda un aprender, sigue arrojando un resultado confiable y exacto.

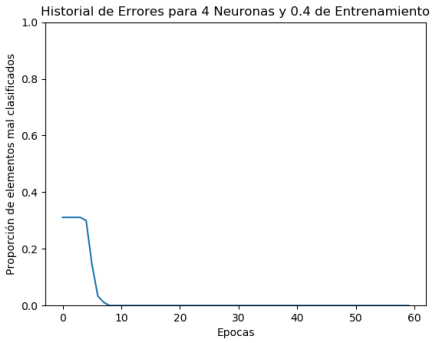
Métricas de Evaluación

	Accuracy	
% de Train	**Train**	**Test**
50 %	0.848	0.8533
60 %	0.8289	0.7733
70 %	0.8591	0.8644
80 %	0.8492	0.8533
90 %	0.8607	0.8267

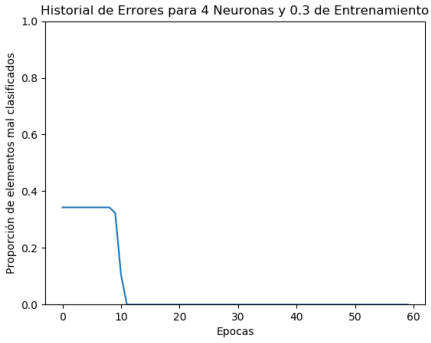
Curvas de convergencia (Error vs. Épocas) para 4 Neuronas



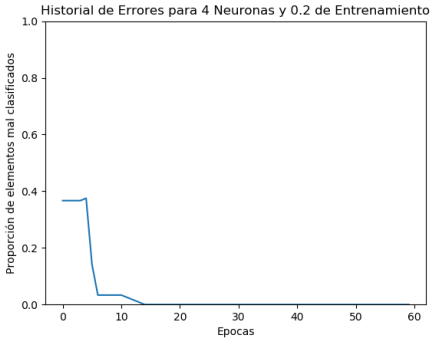
Conjunto de Entrenamiento 50 %



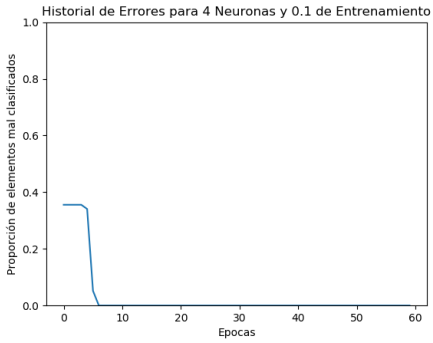
Conjunto de Entrenamiento 60 %



Conjunto de Entrenamiento 70 %



Conjunto de Entrenamiento 80 %



Conjunto de Entrenamiento 90 %

- **5 Neuronas**

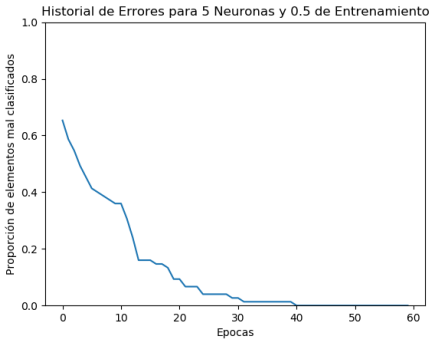
  En este caso podemos decir que usando 60 % y 90 % de la data como entrenamiento se obtienen los resultados con mayor exactitud tanto en el entrenamiento como

en la prueba pero definitivamente mejor es usando 90 % puesto a que converge más rápido.

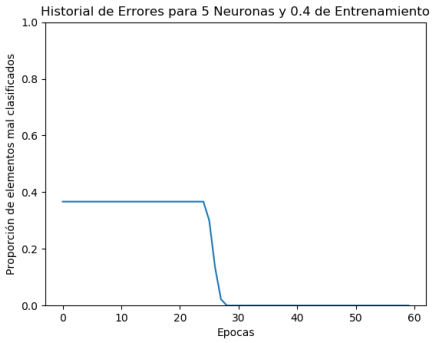
Métricas de Evaluación

	Accuracy	
% de Train	**Train**	**Test**
50 %	0.808	0.7693
60 %	0.8744	0.895
70 %	0.8305	0.7689
80 %	0.865	0.8267
90 %	0.8793	0.9133

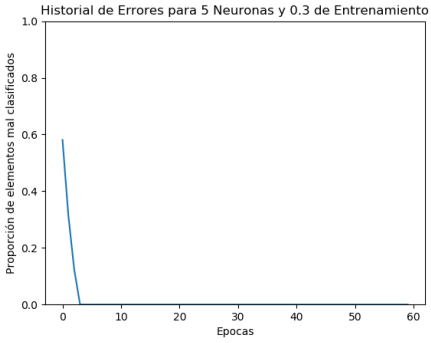
Curvas de convergencia (Error vs. Épocas) para 5 Neuronas



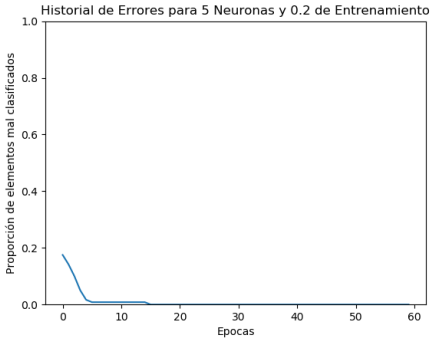
Conjunto de Entrenamiento 50 %



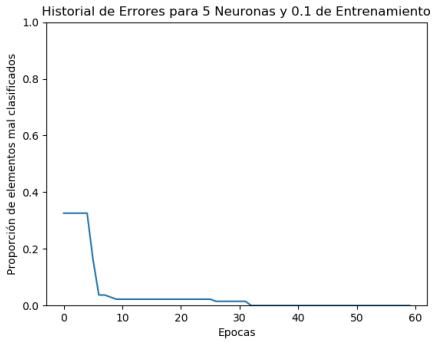
Conjunto de Entrenamiento 60 %



Conjunto de Entrenamiento 70 %



Conjunto de Entrenamiento 80 %



Conjunto de Entrenamiento 90 %

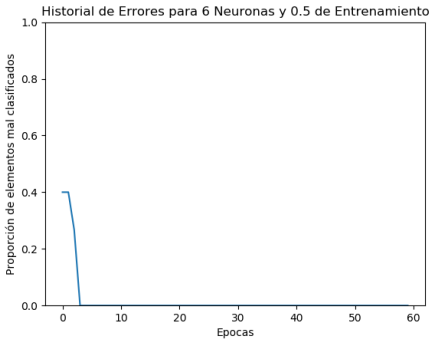
- **6 Neuronas**

  Aquí vemos como la convergencia varia de formas distintas dependiendo de la data que se usa como entrenamiento, a más data más varia el proceso pero siguen siendo exactos.

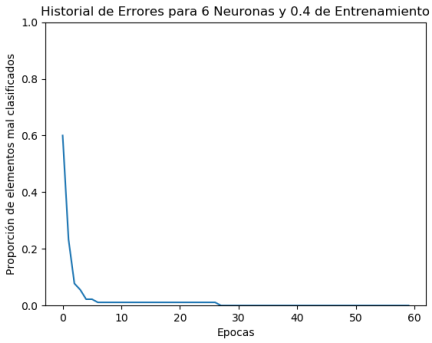
Métricas de Evaluación

	Accuracy	
% de Train	**Train**	**Test**
50 %	0.772	0.772
60 %	0.8678	0.8517
70 %	0.8743	0.8556
80 %	0.8958	0.87
90 %	0.8622	0.8667

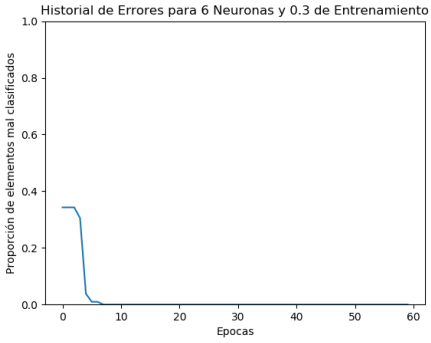
Curvas de convergencia (Error vs. Épocas) para 6 Neuronas



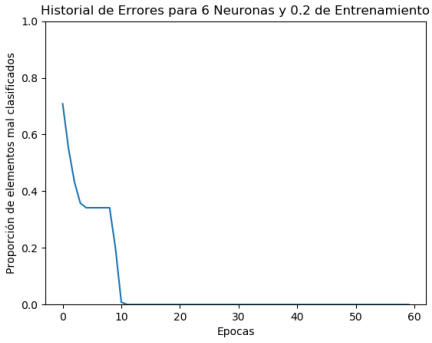
Conjunto de Entrenamiento 50 %



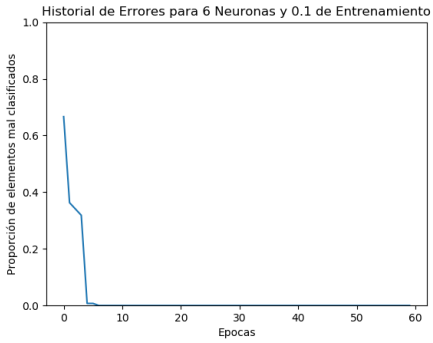
Conjunto de Entrenamiento 60 %



Conjunto de Entrenamiento 70 %



Conjunto de Entrenamiento 80 %



Conjunto de Entrenamiento 90 %

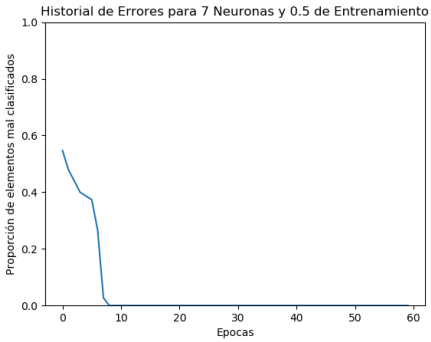
- **7 Neuronas**

  A partir de 7 neuronas se puede ver que la mejor cantidad de data para usarse como entrenamiento es 70 % puesto que si tener que usarse tanta data, la red es capaz de converger de la misma forma que usando 90 %

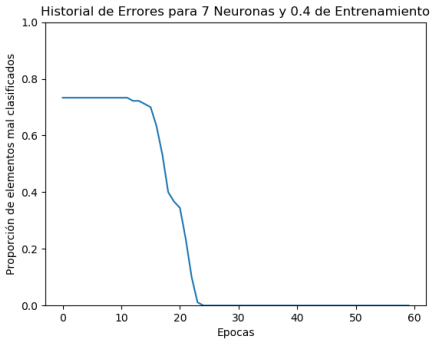
Métricas de Evaluación

	Accuracy	
% de Train	**Train**	**Test**
50 %	0.876	0.856
60 %	0.8567	0.8467
70 %	0.9086	0.8711
80 %	0.8967	0.9033
90 %	0.8926	0.9133

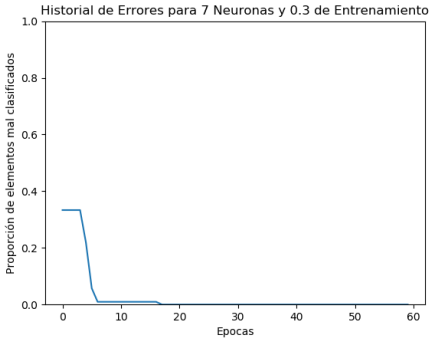
Curvas de convergencia (Error vs. Épocas) para 7 Neuronas



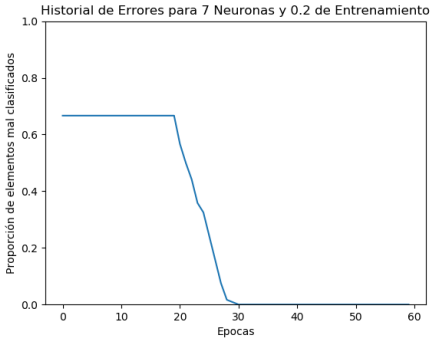
Conjunto de Entrenamiento 50 %



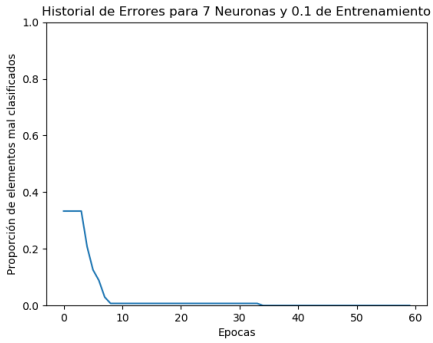
Conjunto de Entrenamiento 60 %



Conjunto de Entrenamiento 70 %



Conjunto de Entrenamiento 80 %



Conjunto de Entrenamiento 90 %

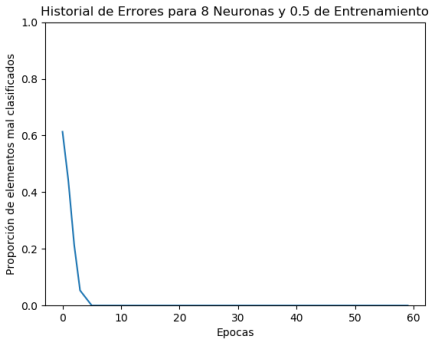
- **8 Neuronas**

  Ya con 8 capaz no importa tanto cuanto porcentaje de data se use como entrenamiento puesto que todas converger de forma regular, lo único que varía será la rapidez con la que convergen

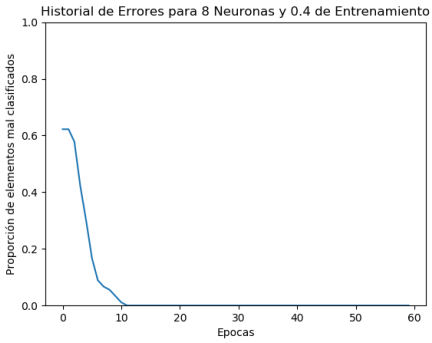
Métricas de Evaluación

	Accuracy	
% de Train	**Train**	**Test**
50 %	0.8587	0.8373
60 %	0.8944	0.8667
70 %	0.8781	0.8711
80 %	0.9058	0.8866
90 %	0.9	0.88

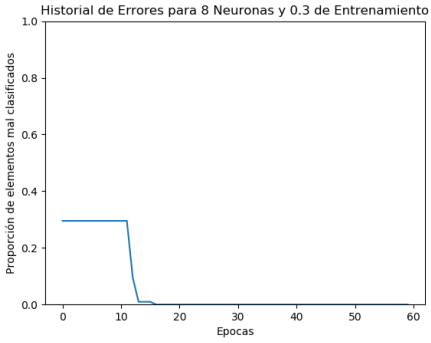
Curvas de convergencia (Error vs. Épocas) para 8 Neuronas



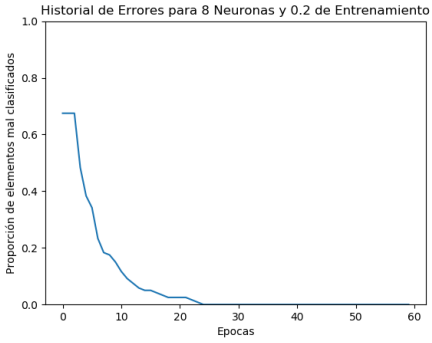
Conjunto de Entrenamiento 50 %



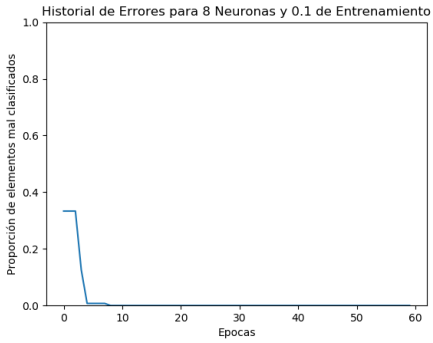
Conjunto de Entrenamiento 60 %



Conjunto de Entrenamiento 70 %



Conjunto de Entrenamiento 80 %



Conjunto de Entrenamiento 90 %

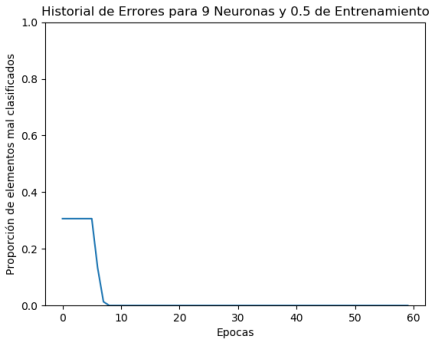
- **9 Neuronas**

  A partir de 9 neuronas, al igual que en el caso de clasificación binaria, se puede ver que mientras más data se suministra para entrenarse, el error converge mucho más rápido.

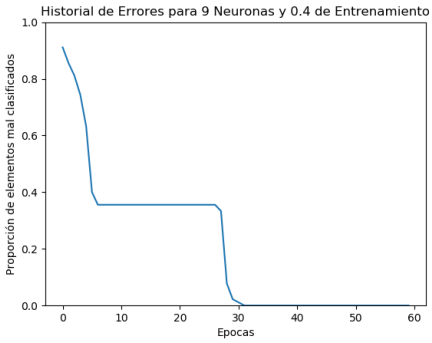
Métricas de Evaluación

	Accuracy	
% de Train	**Train**	**Test**
50 %	0.8667	0.86
60 %	0.8433	0.835
70 %	0.8990	0.8667
80 %	0.9092	0.87
90 %	0.8911	0.86

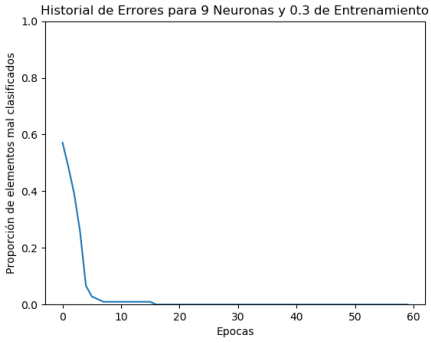
Curvas de convergencia (Error vs. Épocas) para 9 Neuronas



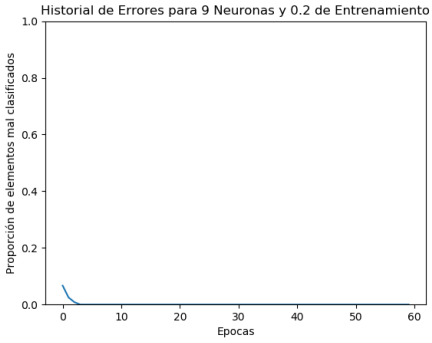
Conjunto de Entrenamiento 50 %



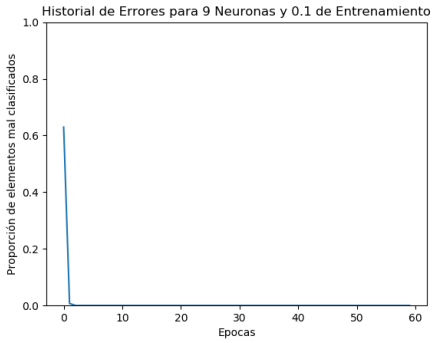
Conjunto de Entrenamiento 60 %



Conjunto de Entrenamiento 70 %



Conjunto de Entrenamiento 80 %



Conjunto de Entrenamiento 90 %

- **10 Neuronas**

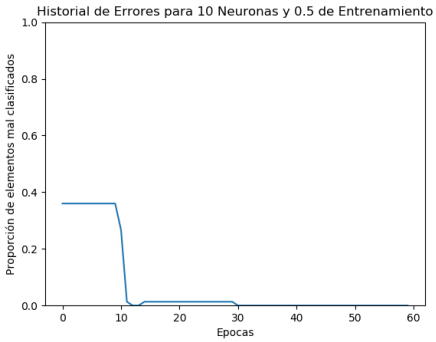
  Como se dijo en el caso de 9 neuronas, la red es mucho más exacta y converge aún más rápido mientras más datos se usan como entrenamiento pero igualmente se va a dar mucho peso a dichos datos puesto que la red va a aprender conforme

los datos sean presentados y dependiendo de los mismo, la red es más susceptible a producir un overfit.

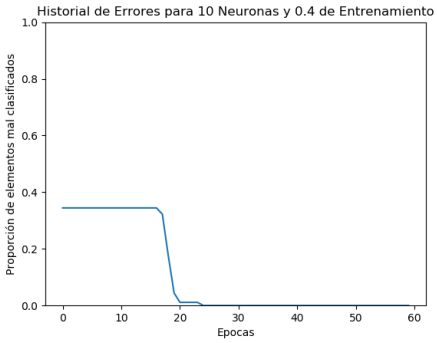
Métricas de Evaluación

	Accuracy	
% de Train	**Train**	**Test**
50 %	0.908	0.8573
60 %	0.8689	0.8833
70 %	0.8714	0.8178
80 %	0.8958	0.8933
90 %	0.9133	0.8467

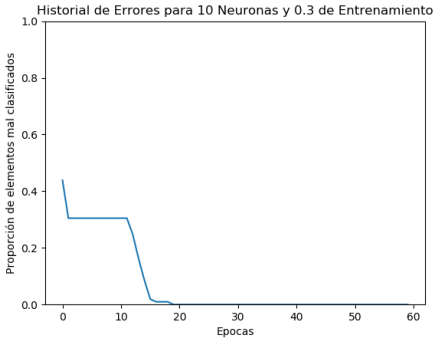
Curvas de convergencia (Error vs. Épocas) para 10 Neuronas



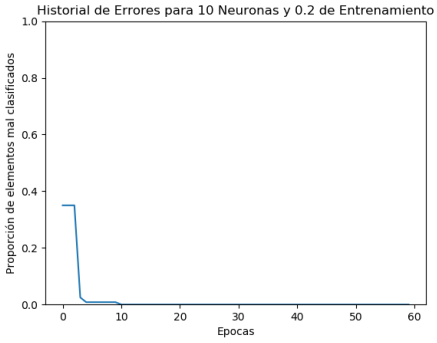
Conjunto de Entrenamiento 50 %



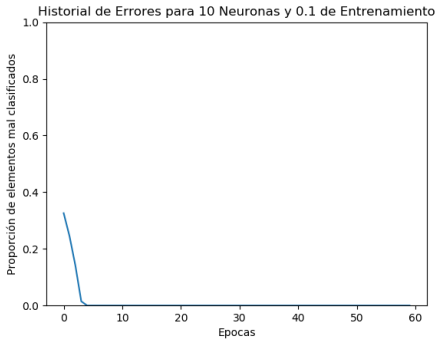
Conjunto de Entrenamiento 60 %



Conjunto de Entrenamiento 70 %



Conjunto de Entrenamiento 80 %



Conjunto de Entrenamiento 90 %

Conclusiones

Después de haber realizado y analizado las distintas pruebas se puede decir que:

- Con respecto a la clasificación del círculo, mientras las zonas que se elijan sean más grandes, la red va a clasificar mejor mientras que a menos rango, menos exactos serán los resultados
- En cuanto al iris data set, la clasificación binaria va a ser mucho más exacta que la clasificación multiclase y en ambos casos la mejor cantidad de data que debe ser usada para entrenar la red es entre 60 % y 70 % de dicha data para evitar un overfitting y por ende una posible mal predicción al momento de la prueba

Es importante destacar que al analizar los resultados de la red multiclase, algo muy importante del proceso es que el mismo varia dependiendo de el orden en que se suministran los datos y por lo mismo es que se puede ver variaciones el las gráficas de de convergencia.

Finalmente se puede decir que sin importar cual caso sea, siempre es mejor usar más neuronas, aunque esto implica que en el proceso de convergencia el error varia más mientras más neuronas se usen, el resultado final va a ser más exacto mientras más neuronas se usen. También es posible que a cierta cantidad de neuronas, la red experimente una mejora significativa pero igualmente sigue habían una mejoría.

Referencias

- **Tipos de Normalización:** http://es.coursera.org/lecture/data-analysis-with-python/data-normalization-in-python-pqNBS.
- **Documentación de la libreria Pandas:** http://pandas.pydata.org/pandas-docs/stable/reference.
- **Documentación de la libreria NumPy:** https://docs.scipy.org/doc/numpy/
- Laminas de las clases.