Sistemas de Inteligencia Artificial
TP Redes Neuronales

Integrantes:

- Maria Victoria Mesa Alcorta, legajo 49297
- Luciana Reznik, legajo 50466

Introducción

Enunciado:

Se debe implementar una red neuronal multicapa con aprendizaje supervisado para resolver el problema de predicción de una serie temporal cuyos datos fueron entregados en el archivo serieTemporal.txt. Se supone que $x(t)$ (el valor de la serie en el paso temporal t) es alguna función desconocida de los valores de la serie en pasos anteriores, es decir:

$$x(t) = f(x(t-1), x(t-2), x(t-3), ...)$$

La red neuronal multicapa que se implemente deberá poder aproximar la función f desconocida. A priori no se conoce cuantos pasos temporales previos hay que considerar como argumento de la función f. Se sugirió usar menos de cuatro.

Para la resolución del problema presentado, se implementó una red neuronal multicapa de tipo Feed Forward con aprendizaje supervisado. Para el entrenamiento de la red se desarrolló el algoritmo *Back Propagation* al cual se le realizaron distintas mejoras.

Implementación

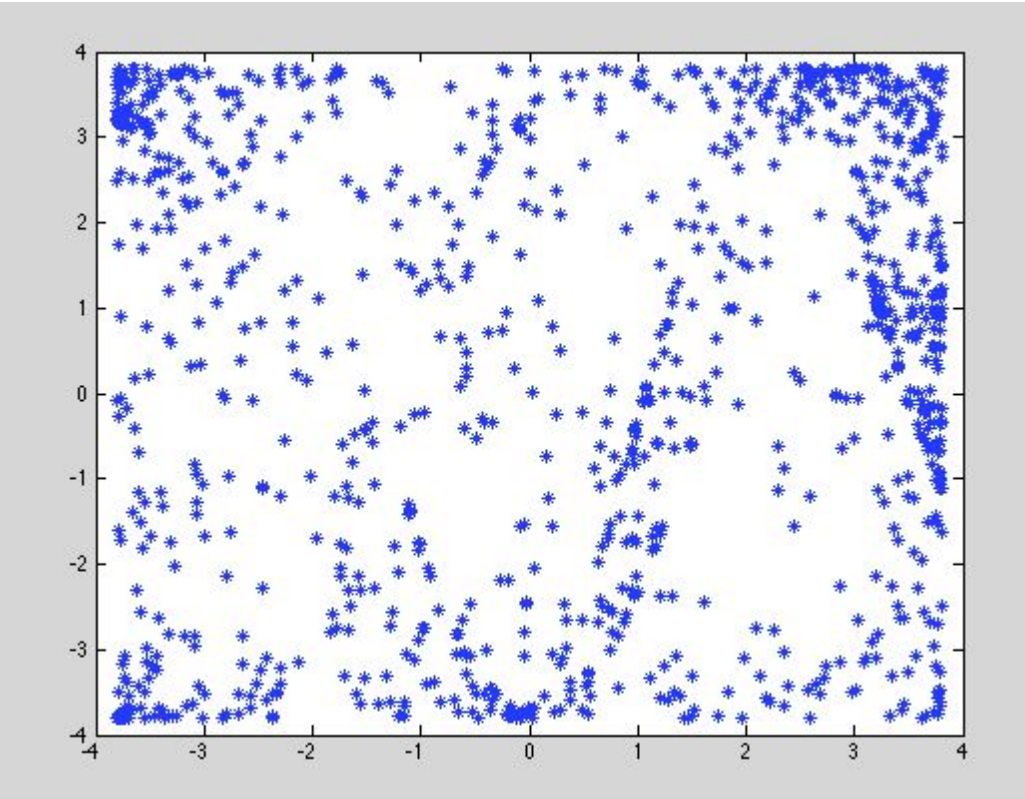
Análisis de los datos recibidos

Se dispuso de una cantidad de datos correspondientes a una serie temporal que era alguna función desconocida de los valores de la serie en pasos anteriores.

$$x(t) = f(x(t-1), x(t-2), x(t-3), ...)$$

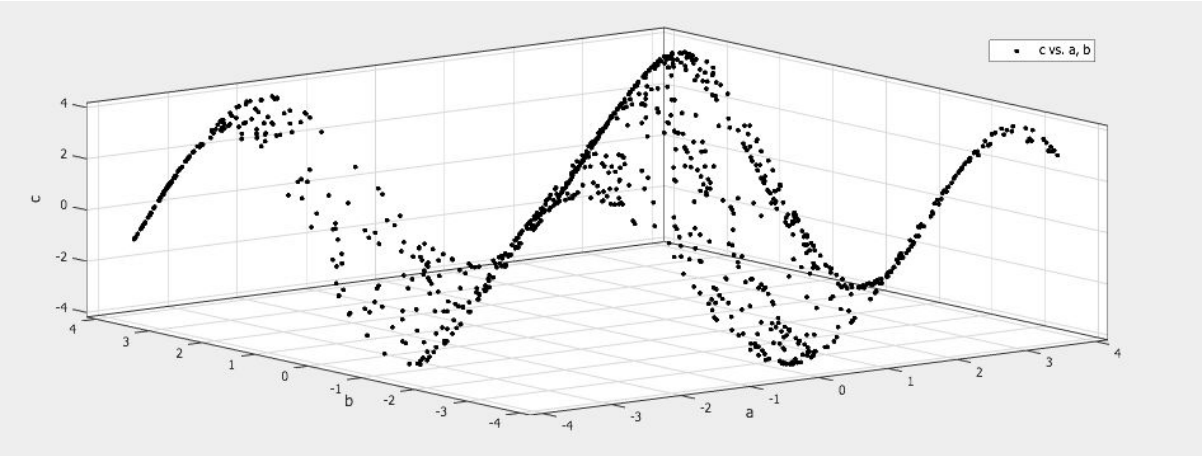
Dado que la cantidad de pasos anteriores se desconoce, se procedió a graficar $x(t)$ para uno y dos pasos anteriores (para tres pasos no es posible graficar la serie). A continuación se muestran los gráficos obtenidos:

1. Para la serie temporal $x(t) = f(x(t-1))$



Imágen 1: Gráfico de la serie temporal $x(t) = f(x(t-1))$

2. Para la serie temporal $x(t) = f(x(t-1), x(t-2))$



Imágen 2: Gráfico de la serie temporal $x(t) = f(x(t-1), x(t-2))$

En los gráficos previos se puede observar que cuando $x(t)$ depende de un paso anterior el gráfico no tiene un patrón fácil de detectar. En cambio cuando $x(t)$ depende de dos pasos anteriores se puede ver claramente una forma definida

Los gráficos previos dieron la intuición de que será más fácil para la red neuronal aprender para $x(t) = f(x(t-1), x(t-2))$. De todos modos se procedió a realizar pruebas para distintas arquitecturas elegidas y distintas funciones de activación comparando la cantidad de pasos. A su vez, se repitió cada prueba tres veces para poder medir el desvío y ver si es una muestra representativa. El error cuadrático medio calculado para cada prueba es el obtenido a partir de correr los patrones de testeo.

Para las siguientes corridas de 1000 épocas de la red se utilizó para el entrenamiento una muestra aleatoria del 70% de los datos y el 30% restante para el testeo[1]. Los pesos iniciales se inicializan en valores random pequeños (entre 0 y 0.5) y durante todo el informe se calculará un porcentaje de aciertos con un error máximo del 10% del valor esperado.

Por otro lado, se usaron dos funciones de activación diferentes, $g(x) = tanh(x)$ y $g(x) = 1/(1+e^{-x})$

[1] Más adelante se explica el porqué de esta decisión.

Se obtuvieron los siguientes resultados:

Función $x(t) = f(x(t-1))$

Arq.	$g(h)$	Eta η	ECM1	%OK	ECM2	%OK	ECM3	%OK	μ	σ
1, 40, 1	*tanh*	0.08	0.4648	0	0.4534	1.34	0.5282	0.67	0.4822	0.0403
1, 40, 1	*exp*	0.08	0.4121	0.67	0.4384	0.33	0.4485	1.67	0.4330	0.0188
1, 6, 9, 1	*tanh*	0.08	0.4479	4.0	0.3925	1.3	0.4679	1.4	0.4361	0.0391
1, 6, 9, 1	*exp*	0.08	0.4545	0.67	0.4190	0.33	0.446	1.00	0.4380	0.0179
1, 9, 13, 1	*tanh*	0.08	0.4007	4.0	0.4194	2.3	0.4007	4.0	0.4069	0.0108
1, 9, 13, 1	*exp*	0.08	0.5359	1.34	0.4454	1.67	0.4383	1.34	0.4732	0.0544

Tabla 1: Comparación de diferentes arquitecturas y funciones de activación cuando la serie depende de un paso anterior. El porcentaje de aciertos es con un error menor del 10% del valor esperado.

Función $x(t) = f(x(t-1), x(t-2))$

Arq.	$g(h)$	Eta η	ECM1	%OK	ECM2	%OK	ECM3	%OK	μ	σ
2, 40, 1	*tanh*	0.08	0.0057	60.7	0.0060	54.0	0.0050	55.7	0.0056	5.13e-04
2, 40, 1	*exp*	0.08	0.4563	1.34	0.4515	2.68	0.4584	0.34	0.4554	0.0035
2, 6, 9, 1	*tanh*	0.08	4.33e-04	90.27	8.87e-04	81.54	0.0018	74,50	0.0010	6.96e-04
2, 6, 9, 1	*exp*	0.08	0.4476	0.4476	0.4508	1.34	0.4195	0.39	0.4393	0.0171
2, 9, 13, 1	*tanh*	0.08	5.57e-04	88.93	5.68e-04	85.9	0.0011	85.9	7.42e-04	3.10e-04
2, 9, 13, 1	*exp*	0.08	0.2560	5,70	0.4218	1.34	0.3634	3.02	0.3484	0.0819

Tabla 2: Comparación de diferentes arquitecturas y funciones de activación cuando la serie depende de dos pasos anteriores. El porcentaje de aciertos es con un error menor del 10% del valor esperado.

Función $x(t) = f(x(t-1), x(t-2), x(t-3))$

Arq.	$g(h)$	Eta η	ECM1	%OK	ECM2	%OK	ECM3	%OK	μ	σ
3,40, 1	*tanh*	0.08	0.0070	52.2	0.0069	50.8	0.0069	50.9	0.0069	5.78e-05
3, 40, 1	*exp*	0.08	0.4525	1.01	0.4534	1.01	0.4650	1.68	0.4569	0.0070
3, 6, 9, 1	*tanh*	0.08	0.0011	75.08	0.0011	75.76	0.0049	73.06	0.0024	0.0022
3, 6, 9, 1	*exp*	0.08	0.4113	0.67	0.4613	0	0.4345	0.33	0.4357	0.0251
3, 9, 13, 1	*tanh*	0.08	5.07e-04	81.5	4.49e-04	86.2	0.0013	81.81	7.52e-04	4.75e-04
3, 9, 13, 1	*exp*	0.08	0.4191	2.69	0.4849	1.34	0.4448	0.33	0.4496	0.0332

Tabla 3: Comparación de diferentes arquitecturas y funciones de activación cuando la serie depende de tres paso anteriores. El porcentaje de aciertos es con un error menor del 10% del valor esperado.

Luego de estas pruebas se corroboró la hipótesis planteada anteriormente y se decidió elegir a $x(t)$ como una serie temporal que depende de dos pasos anteriores, es decir $x(t) = f(x(t-1), x(t-2))$. También se puede observar que la función de activación $tanh(h)$ es la función que mejor ayuda al aprendizaje por lo cual se realizarán todas las próximas pruebas con la misma.

Preparación de datos a utilizar

Es muy importante que los patrones elegidos para el entrenamiento de la red neuronal sean datos representativos de la serie, de esta forma la red podrá generalizar a partir de lo que aprendió. Es por ello que antes de comenzar el entrenamiento, fue necesario dividir los datos que se usarían para el mismo y los que servirían para testeo.
Para ello se crearon conjuntos de distintos tamaños con patrones elegidos aleatoriamente:

- 30% para entrenamiento y 70% para testeo:

Arq.	Eta η	ECM1	%OK	ECM2	%OK	ECM3	%OK	μ	σ
2, 9, 13, 1	0.08	0.0053	66.61	0.0026	72.34	0.0025	74.35	0.0035	0.0016
3, 9, 13, 1	0.08	0.0175	44.18	0.0057	55.81	0.0037	70.01	0.0090	0.0075

Tabla 4: Corridas con patrones de 30% para entrenamiento y 70% para testeo. El porcentaje de aciertos es con un error menor del 10% del valor esperado.

- 50% para entrenamiento y 50% para testeo:

Arq.	Eta η	ECM1	%OK	ECM2	%OK	ECM3	%OK	μ	σ
2, 9, 13, 1	0.08	4.39e-04	83.5	0.0015	81.9	0.0021	74.1	0.0013	8.41e-04
3, 9, 13, 1	0.08	0.0025	79.68	9.29e-04	83.90	0.0022	72.64	0.0019	8.34e-04

Tabla 5: Corridas con patrones de 50% para entrenamiento y 50% para testeo. El porcentaje de aciertos es con un error menor del 10% del valor esperado.

- 70% para entrenamiento y 30% para testeo:

Arq.	Eta η	ECM1	%OK	ECM2	%OK	ECM3	%OK	μ	σ
2, 9, 13, 1	0.08	0.0010	71.5	5.68e-04	85.9	0.0011	85.9	8.89e-04	2.83e-04
3, 9, 13, 1	0.08	5.07e-04	81.5	4.49e-04	86.2	0.0013	81.8	7.52e-04	4.75e-04

Tabla 6: Corridas con patrones de 70% para entrenamiento y 30% para testeo. El porcentaje de aciertos es con un error menor del 10% del valor esperado.

- 90% para entrenamiento y 10% para testeo:

Arq.	Eta η	ECM1	%OK	ECM2	%OK	ECM3	%OK	μ	σ
2, 9, 13, 1	0.08	0.0014	72.45	5.33e-04	81.6	3.59e-04	83.67	7.64e-04	5.57e-04
3, 9, 13, 1	0.08	9.45e-04	84.53	4.92e-04	84.5	2.95e-04	93.81	5.77e-04	5.76e-04

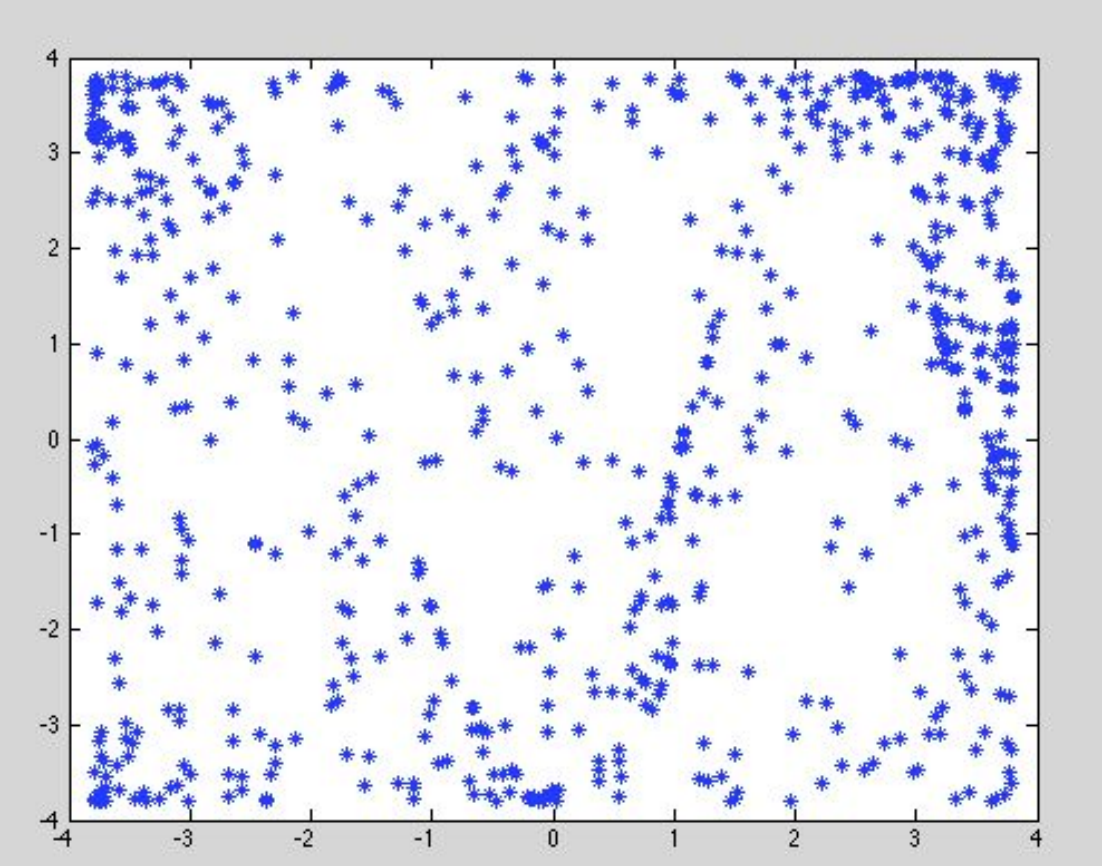
Tabla 7: Corridas con patrones de 90% para entrenamiento y 10% para testeo. El porcentaje de aciertos es con un error menor del 10% del valor esperado.

En la tabla 4 se puede ver que cuando se le da pocos patrones de entrenamiento a la red, si bien puede llegar a errores similares que con otras arquitecturas, no generaliza bien. Esto se debe a que con una cantidad tan baja de patrones no se puede armar un conjunto representativo de la serie. Por otro lado, las últimas tres tablas dan errores y porcentajes de acierto similares, se optó por usar un 70% para entrenamiento y 30% para testeo. Con esta elección de patrones se prioriza el entrenamiento sobre el testeo y a su vez la cantidad de valores de testeo no es tan baja como lo es en la opción de 90% para entrenamiento y 10% para testeo y se puede verificar que haya generalizado bien.

A su vez, para evitar que a medida que se corre el backpropagation los pesos se balanceen para un solo lado, en cada iteración del mismo, se hizo una mezcla al azar del orden en que se irán recibiendo los datos por parte de la red

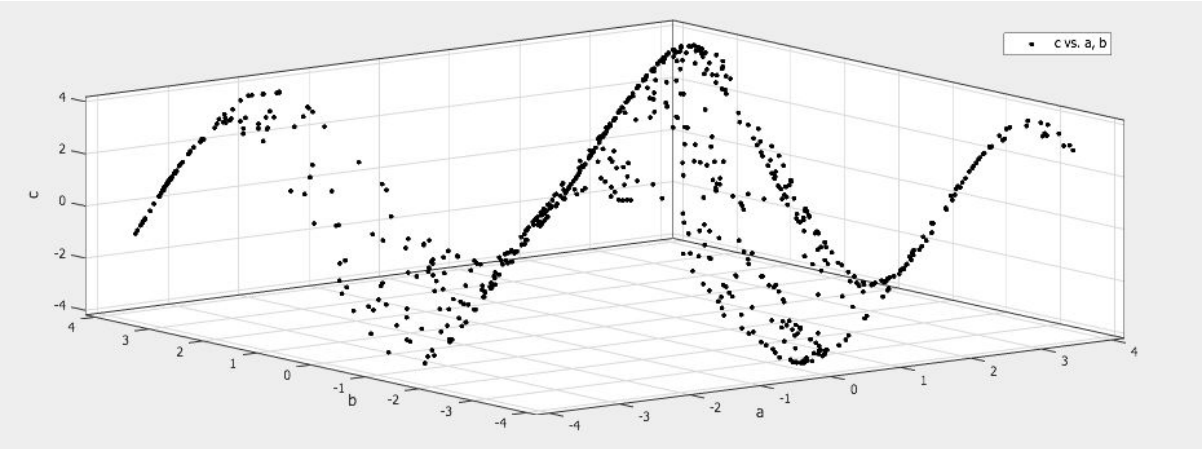
En los siguientes gráficos se pueden ver el 70% de los puntos elegidos aleatoriamente para entrenamiento, en una corrida dada. Se observa que la función mantiene una disposición de los puntos muy similar respecto de la original.

1. Para la serie temporal $x(t) = f(x(t-1))$



Imágen 3: Gráfico de la serie temporal $x(t) = f(x(t-1))$ con el 70% de los patrones elegidos aleatoriamente.

2. Para la serie temporal $x(t) = f(x(t-1), x(t-2))$



Imágen 4: Gráfico de la serie temporal $x(t) = f(x(t-1), x(t-2))$con el 70% de los patrones elegidos aleatoriamente.

Normalización

Los valores de la serie temporal están comprendidos en un rango de (-4,4). Se procedió a normalizar las entradas a el rango (-1,1). Dado que la función de activación Sigmoide $g(h) = 1/(1+e^{-h})$ da salidas en el rango (0,1), para obtener las salidas de la red neuronal en el rango (-1,1), se puso como función de activación en la capa de salida $g(h) = tanh(h)$. Para la función de activación $g(h) = tanh(h)$ eso no fué necesario ya que sus salidas ya están comprendidas en el rango (-1,1).

Mejoras implementadas

Para acelerar el aprendizaje de la red neuronal y evitar que se estanque en mínimos locales, se decidieron implementar distintas mejoras al algoritmo de backpropagation y ver cómo afectan al mismo.

Cálculo de regla delta

Para mantener los deltas distintos de cero y por ende evitar mínimos locales que se producen cuando la derivada es nula, se calculan los deltas de la siguiente manera:

$$\delta_i = [g'(h_i) + 0.1] (S_i - o_i)$$

A continuación se realizan algunas pruebas con la arquitectura [2, 9, 13, 1] utilizada previamente con y sin esta mejora para ver su incidencia.

Arq.	Mejora	**Eta** η	**ECM1**	**%OK**	**ECM2**	**%OK**	**ECM3**	**%OK**	μ	σ
2, 9, 13, 1	SI	0.08	0.0050	70.13	0.0086	56.71	0.0029	67.11	0.0055	0.0029
2, 9, 13, 1	NO	0.08	0.0144	63.08	0.0301	32.89	0.0061	63.42	0.0169	0.0122
2, 10, 6, 1	SI	0.08	0.0086	57.05	0.0058	60.40	0.0092	41.95	0.0079	0.0018
2, 10, 6, 1	NO	0.08	0.0077	53.20	0.0057	58.39	0.0057	55.37	0.0064	0.0012
2, 6, 9, 1	SI	0.08	0.0032	70.13	0.0016	78.52	0.0021	74.50	0.0023	0.0008
2, 6, 9, 1	NO	0.08	0.0047	64.77	0.0022	67.11	0.0041	67.11	0.0037	0.0013

Tabla 8: Corridas de 200 épocas con y sin la mejora para el Delta usando $g(h) = tanh(h)$. El porcentaje de aciertos es con un error menor del 10% del valor esperado.

En la tabla 8 se puede ver que si bien no se nota una mejora importante, en algunos casos no hacerlo implica un error cuadrático medio levemente más alto e incluso una mayor dispersión en las diferentes corridas. A su vez, como tampoco esto empeora el aprendizaje, se decidió continuar usándolo ya que esto prevendría quedar estancado en mínimos locales donde la derivada daría 0.

Momentum

Se agregó un término que pese el descenso promedio para suavizar oscilaciones:

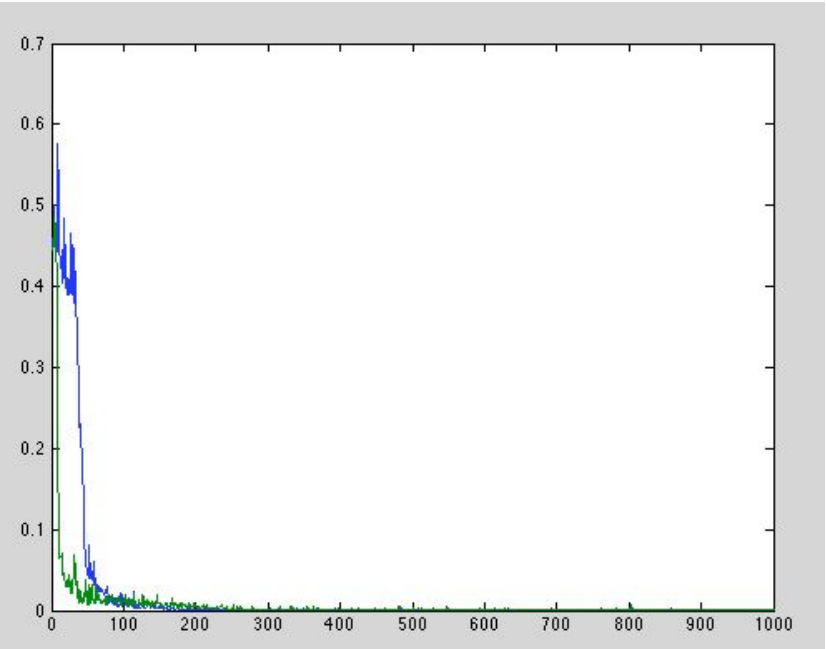
$$\Delta w_{pq}(t+1) = -\eta(\partial E/\partial w_{pq}) + \alpha \Delta w_{pq}(t) \text{ con } 0 < \alpha < 1$$

Arq.	Eta η	Alpha α	ECM1	%OK	ECM2	%OK	ECM3	%OK	μ	σ
2, 9, 13, 1	0.08	0	0.0011	76.51	3.35e-04	88.59	4.64e-04	83.89	6.33e-04	4.09e-04
2, 9, 13, 1	0.08	0.1	0.0120	56.04	5.91e-04	82.88	0.0034	61.74	0.0053	0.0059
2, 9, 13, 1	0.08	0.2	3.92e-04	87.58	0.0021	71.47	5.28e-04	86.91	0.0010	0.0009
2, 9, 13, 1	0.08	0.3	6.62e-04	87.24	2.28e-04	95.30	4.11e-04	85.23	4.34e-04	2.18e-04
2, 9, 13, 1	0.08	0.4	5.37e-04	86.91	2.98e-04	91.61	5.33e-04	88.25	4.56e-04	1.37e-04
2, 9, 13, 1	0.08	0.5	0.0012	75.50	3.46e-04	87.58	3.89e-04	85.23	6.45e-04	4.81e-04
2, 9, 13, 1	0.08	0.6	4.15e-04	88.25	2.44e-04	95.97	3.31e-04	86.91	3.30e-04	8.55e-05
2, 9, 13, 1	0.08	0.7	3.38e-04	92.28	7.70e-04	85.57	4.95e-04	86.24	5.34e-04	2.19e-04
2, 9, 13, 1	0.08	0.8	0.0020	68.79	5.82e-04	90.93	0.0015	78.18	0.0014	7.20e-04
2, 9, 13, 1	0.08	0.9	0.4157	13.42	0.51	16.10	1.0778	12.75	0.6678	0.3582

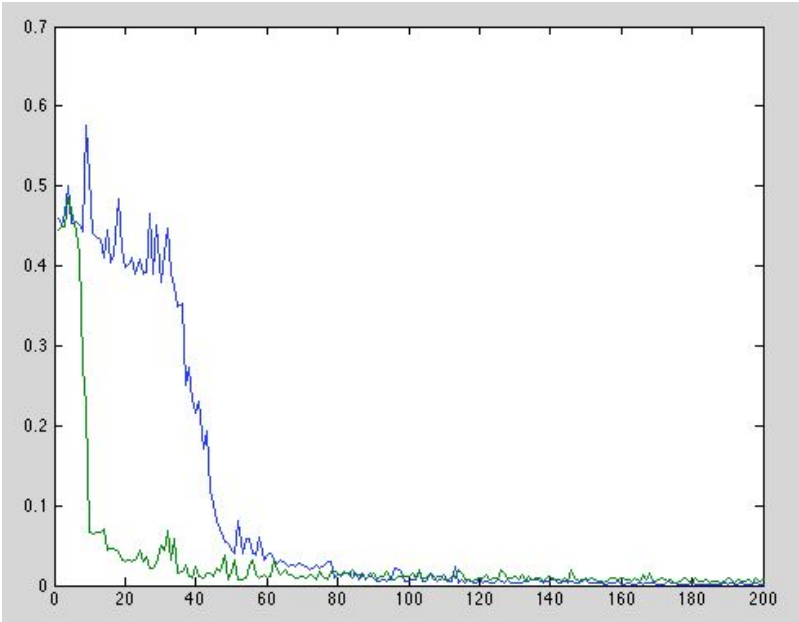
Tabla 9: Corridas de 1000 épocas con distintos valores de alpha. El porcentaje de aciertos es con un error menor del 10% del valor esperado.

En la tabla se puede ver que a las mejores generalizaciones se las logró utilizando la mejora del momentum. También se puede ver que para esos mejores resultados la dispersión del ECM es muy baja, por eso se decidió de ahora en más utilizar la mejora *momentum* del algoritmo.

En el siguiente gráfico se ve la comparación de la evolución del ECM en función de las épocas de una red que utiliza la mejora de momentum con un alpha = 0.6 (línea verde) respecto de otra que no la tiene (línea azul). Se puede ver claramente cómo la línea verde (con momentum) tiene oscilaciones más suaves y logra llegar a mejores errores más rápido que la azul.



Imágen 5: Gráfico la mejora de momentum con $\alpha = 0$ *(azul) y* $\alpha = 0.6$ *(verde)*



Imágen 6: ZOOM en las primeras 200 épocas: Gráfico la mejora de momentum con $\alpha = 0$ *(azul) y* $\alpha = 0.6$ *(verde)*

Eta adaptativo

Buenos valores de η al principio pueden ser malos al final o viceversa. Es por ello que si una actualización de los pesos de la red neuronal no reduce la función de costo, η debería decrementarse *geométricamente*. Luego de decrementar η, se deshace ese paso y se setea el *momentum* $\alpha = 0$ hasta encontrar un paso bueno.
Por lo contrario, si *k* iteraciones decrementaron la función de costo, entonces η debería crecer *en una constante* para acelerar el aprendizaje y no ir tan lento.

Cuando el eta es muy pequeño la red converge tan lentamente que termina siendo contraproducente. Cuando se eligen malos valores de incremento y decremento para parámetros adaptativos, el eta disminuye tanto que produce un estancamiento en el error. Es por eso que se decidió que cuando eta es más pequeño que 0.01 se lo resetea al valor inicial.

Arq.	Alpha α	M	Step	Inc	Dec	Épocas	ECM	%OK - 10%
2, 9, 13, 1	0.3	700	4	0.01	0.01	1000	0.011356	37.58
2, 9, 13, 1	0.3	700	4	0.01	0.01	1000	0.34956	1.68
2, 9, 13, 1	0.3	700	4	0.01	0.01	1000	0.4140	1.34

Tabla 10: Corridas de 1000 épocas con un eta inicial de 0.08.

Se realizaron más pruebas con otro incremento tratando de encontrar mejores resultados:

Arq.	Alpha α	M	Step	Inc	Dec	Épocas	ECM	%OK - 10%
2, 9, 13, 1	0.3	700	4	0.0001	0.01	1000	0.0023	76.17
2, 9, 13, 1	0.3	700	4	0.0001	0.01	1000	0.0096	43.62
2, 9, 13, 1	0.3	700	4	0.0001	0.01	1000	0.0230	39.26
2, 9, 13, 1	0.3	700	4	0.0001	0.01	1000	0.0106	41.61

Tabla 11: Corridas de 1000 épocas con un eta inicial de 0.08.

A continuación se repitieron las pruebas pero con la modificación de resetear el eta a un valor más pequeño de 0.02, ya que un valor alto hacía que el ECM oscile más cuando podría haber continuado con un eta pequeño. También se cambió el incremento.

Arq.	Alpha α	M	Step	Inc	Dec	Épocas	ECM	%OK - 10%
2, 9, 13, 1	0.3	700	4	0.001	0.01	1000	0.0042	55.37
2, 9, 13, 1	0.3	700	4	0.001	0.01	1000	0.0444	33.89
2, 9, 13, 1	0.3	700	4	0.001	0.01	1000	0.0132	44.97

Tabla 12: Corridas de 1000 épocas con un eta inicial de 0.08 reseteando el eta a 0.02 cuando alcanza su valor mínimo.

Se repitieron las pruebas pero ahora permitiendo que eta baje hasta un mínimo de 0.005 sin resetearlo.

Arq.	Alpha α	M	Step	Inc	Dec	Épocas	ECM	%OK - 10%
2, 9, 13, 1	0.3	700	4	0.001	0.01	1000	0.4227	1.34
2, 9, 13, 1	0.3	700	4	0.001	0.01	1000	0.0067	41.28
2, 9, 13, 1	0.3	700	4	0.001	0.01	1000	0.0172	34.90

Tabla 13: Corridas de 1000 épocas con un eta inicial de 0.08 permitiendo un eta mínimo de 0.005.

Como se puede ver en la Tabla 13, permitir un eta de 0.005 produjo una convergencia demasiado lenta y terminó siendo perjudicial al aprendizaje. El eta se mantuvo en una gran cantidad de iteraciones en 0.005 y tardó mucho en mejorar. Es por eso que se repitió la prueba permitiendo un eta mínimo de 0.01.

Arq.	Alpha α	M	Step	Inc	Dec	Épocas	ECM	%OK - 10%
2, 9, 13, 1	0.3	700	4	0.001	0.01	1000	0.0054	48.32
2, 9, 13, 1	0.3	700	4	0.001	0.01	1000	0.0387	29.19
2, 9, 13, 1	0.3	700	4	0.001	0.01	1000	0.0043	58.73

Tabla 14: Corridas de 1000 épocas con un eta inicial de 0.08 permitiendo un eta mínimo de 0.01.

Más pruebas con eta adaptativo y *tanh*:

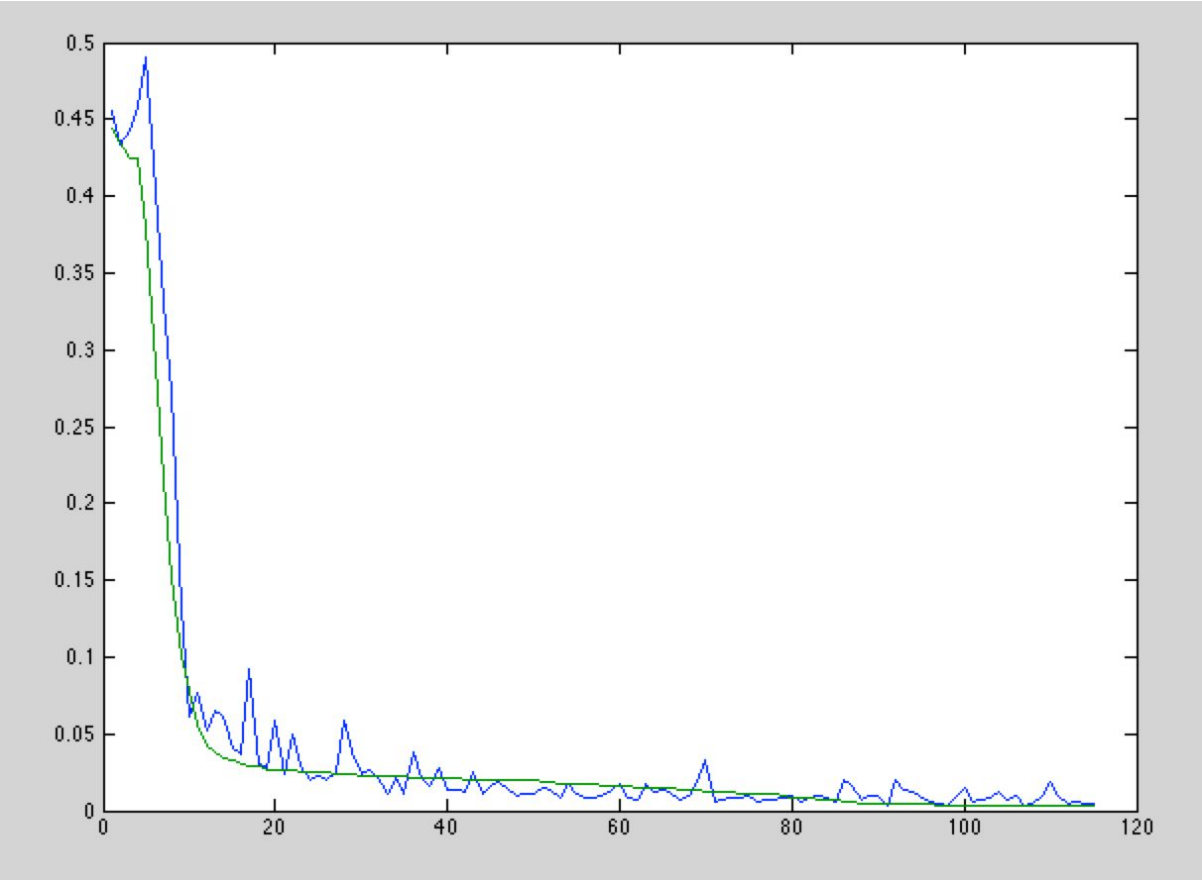
Todas las pruebas previas de parámetros adaptativos se realizaron con la misma arquitectura, es por eso que se procedió a realizar nuevas pruebas con dos arquitecturas más pero no se consiguieron mejores resultados.

Arq.	Alpha α	M	Step	Inc	Dec	Épocas	ECM	%OK - 10%
2, 10, 6, 1	0.9	200	5	0.05	0.0001	5000	0.0047	56.71
2, 10, 6, 1	0.9	300	5	0.005	0.001	1000	0.0061	53.69
2, 10, 6, 1	0.9	200	10	0.0005	0.08	1000	0.0076	49.33
2, 10, 6, 1	0.9	100	5	0.05	0.001	1000	0.0380	40.60
2, 4, 5, 1	0.9	100	2	0.09	0.0001	1000	0.0297	37.24
2, 4, 5, 1	0.9	200	2	0.01	0.0001	1000	0.0348	31.54
2, 4, 5, 1	0.9	500	2	0.05	0.001	1000	0.0201	66.10

Tabla 15: Corridas de distintas arquitecturas con distintos valores para eta adaptativo.

Comparación del error con eta adaptativo vs error sin eta adaptativo

En el siguientes gráfico podemos observar cómo varía el error utilizando o no la mejora de eta adaptativo. Se puede ver que si bien sin parámetros adaptativos el ECM oscila se llega a un error más bajo y no se tarda más que con parámetros adaptativos en conseguirlo.
Por eso se puede decir que la mejora de eta adaptativo no es fructífera para la red.



Imágen 7: Gráfico de misma configuración sin y con parámetros adaptativos.

Conclusiones

La primer conclusión a la que se llegó, es que no existe una configuración ideal de la red que aplique siempre. Las variables se fueron ajustando (arquitectura, cantidad de corridas, mejoras al algoritmo, entre otras) son dependientes entre ellas y con los parámetros de entrada. Eso quiere decir por ejemplo que si cierta arquitectura llega a mejores resultados con cierto valor de alpha, luego al probar con otra arquitectura no necesariamente esa ese alpha el indicado. Es por eso, que idealmente habría que hacer combinaciones de todo con todo para obtener el mejor resultado posible.

Para el trabajo presentado, y a fines prácticos, se fueron sacando ciertas conclusiones basadas en los resultados paso a paso cómo elegir una u otra función de activación, arquitectura, etc. pero se sabe que puede haber alguna combinación de factores que no se haya llegado a probar y que sea la que obtenga mejores resultados.

A partir de todas las pruebas mostradas durante el informe se procedió a elegir configuración la que se consideró mejor. Luego, se procedió a realizar pruebas con 5.000 épocas para tratar de llegar al ECM más bajo posible lo cual conlleva a un mayor porcentaje de aciertos (dado que se sabe que esa configuración logra una buena generalización).

A continuación se muestra una tabla con corridas para la mejor configuración encontrada para este problema con los 3 distintos valores de alpha que dieron mejores resultados.

Arq.	Alpha α	eta (No adapt.)	Épocas	ECM	%OK - 10%	%OK - 5%
2, 9, 13, 1	0	0.08	5.000	3.4387e-05	97.31	95.97
2, 9, 13, 1	0	0.08	5.000	3.3657e-04	91.61	78.85
2, 9, 13, 1	0	0.08	5.000	1.1067e-04	90.60	83.55

Tabla 16: Corridas de mejor configuración encontrada. ECM promedio: 1.6054e-04, desvío: 1.5714e-04.

Arq.	Alpha α	eta (No adapt.)	Épocas	ECM	%OK - 10%	%OK - 5%
2, 9, 13, 1	0.3	0.08	5.000	1.1353e-04	90.60	77.85
2, 9, 13, 1	0.3	0.08	5.000	1.8129e-05	96.64	94.29
2, 9, 13, 1	0.3	0.08	5.000	2.0144e-04	94.96	81.87

Tabla 17: Corridas de mejor configuración encontrada. ECM promedio: 1.1103e-04, desvío: 9.1681e-05.

Arq.	Alpha α	eta (No adapt.)	Épocas	ECM	%OK - 10%	%OK - 5%
2, 9, 13, 1	0.6	0.08	5.000	7.6156e-05	96.30	91.94
2, 9, 13, 1	0.6	0.08	5.000	3.4785e-04	86.24	77.85
2, 9, 13, 1	0.6	0.08	5.000	3.4387e-05	97.31	95.97

Tabla 18: Corridas de mejor configuración encontrada. ECM promedio: 1.5280e-04, desvío: 1.7021e-04.

Los pesos iniciales de las pruebas previas son valores aleatorios comprendidos entre 0 y 0.5. Es por eso que se repetirán las pruebas para $\alpha = 0$ con valores valores iniciales entre -0.5 y 0.5 a ver si se notan cambios importantes.

Arq.	Alpha α	eta (No adapt.)	Épocas	ECM	%OK - 10%	%OK - 5%
2, 9, 13, 1	0	0.08	5.000	1.6158e-04	93.29	83.22
2, 9, 13, 1	0	0.08	5.000	5.0526e-04	84.90	73.49
2, 9, 13, 1	0	0.08	5.000	4.4952e-04	94.63	84.23

Tabla 19: Corridas de mejor configuración encontrada con pesos iniciales entre -0.5 y 0.5. ECM promedio: 3.7212e-04, desvío:1.8445e-04.

Como se puede ver, no se notan grandes diferencias entre iniciar el aprendizaje con pesos entre (0, 0.5) y (-0.5, 0.5).

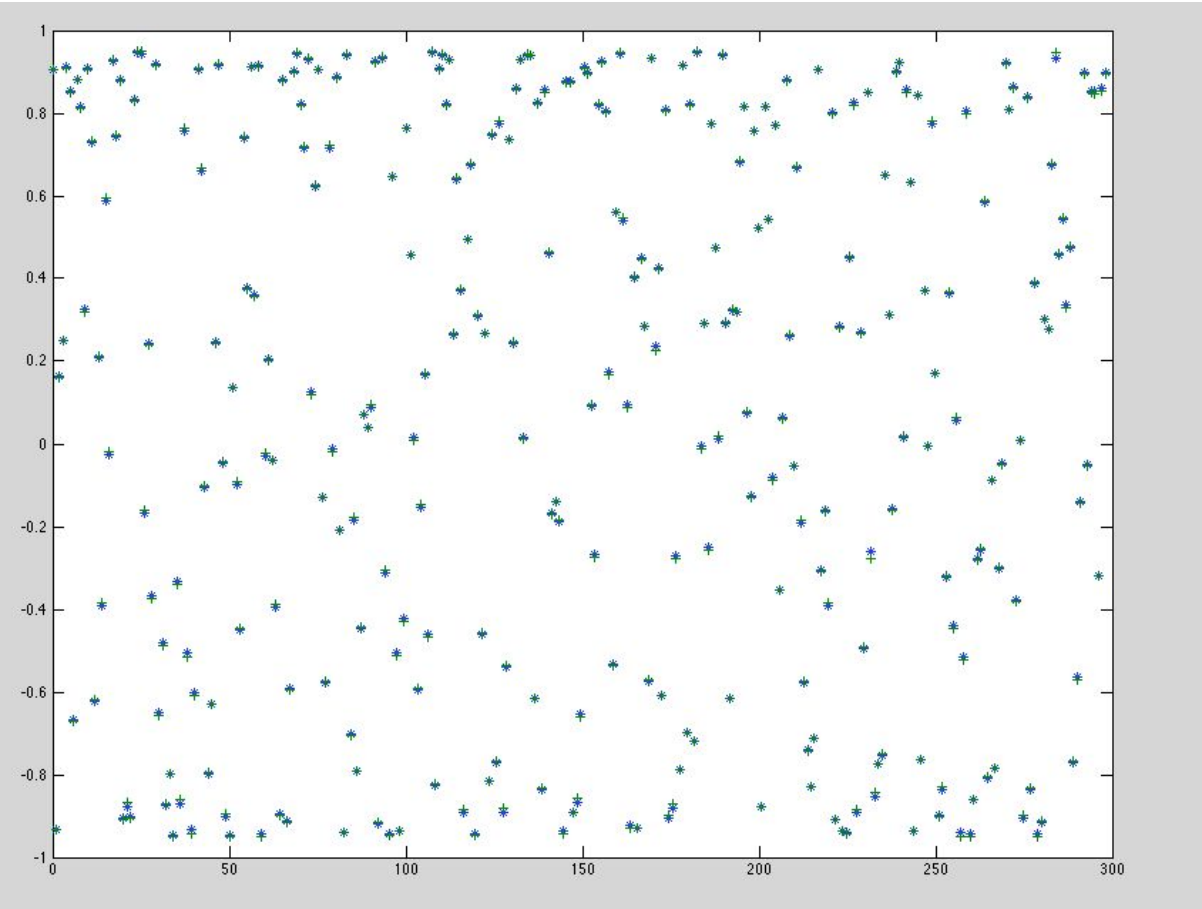
Finalmente se hizo una corrida de todos los valores de la serie con los pesos que dieron el porcentaje de aciertos más grande y se obtuvo:

ECM	%OK - 10%	%OK - 5%	%OK - 1%
0.0038141	96.991	92.837	69.771

Tabla 20: Corrida de todos los patrones de la serie con los pesos de la mejor configuración.

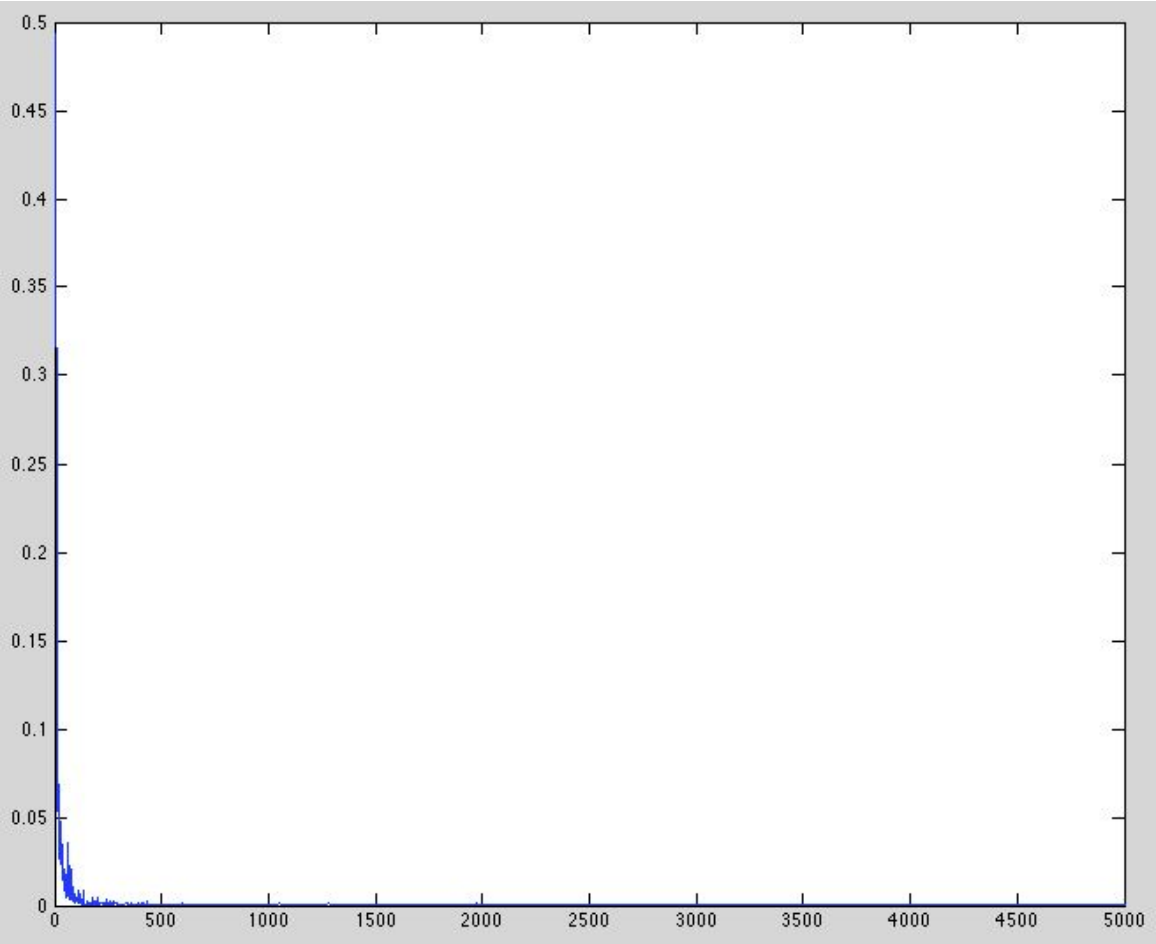
Gráficos de mejor configuración encontrada

El siguiente gráfico tiene la serie de puntos esperada y la serie de puntos obtenida en el testeo de la red para la configuración final elegida. El azul representa los puntos obtenidos y el verde los esperados.



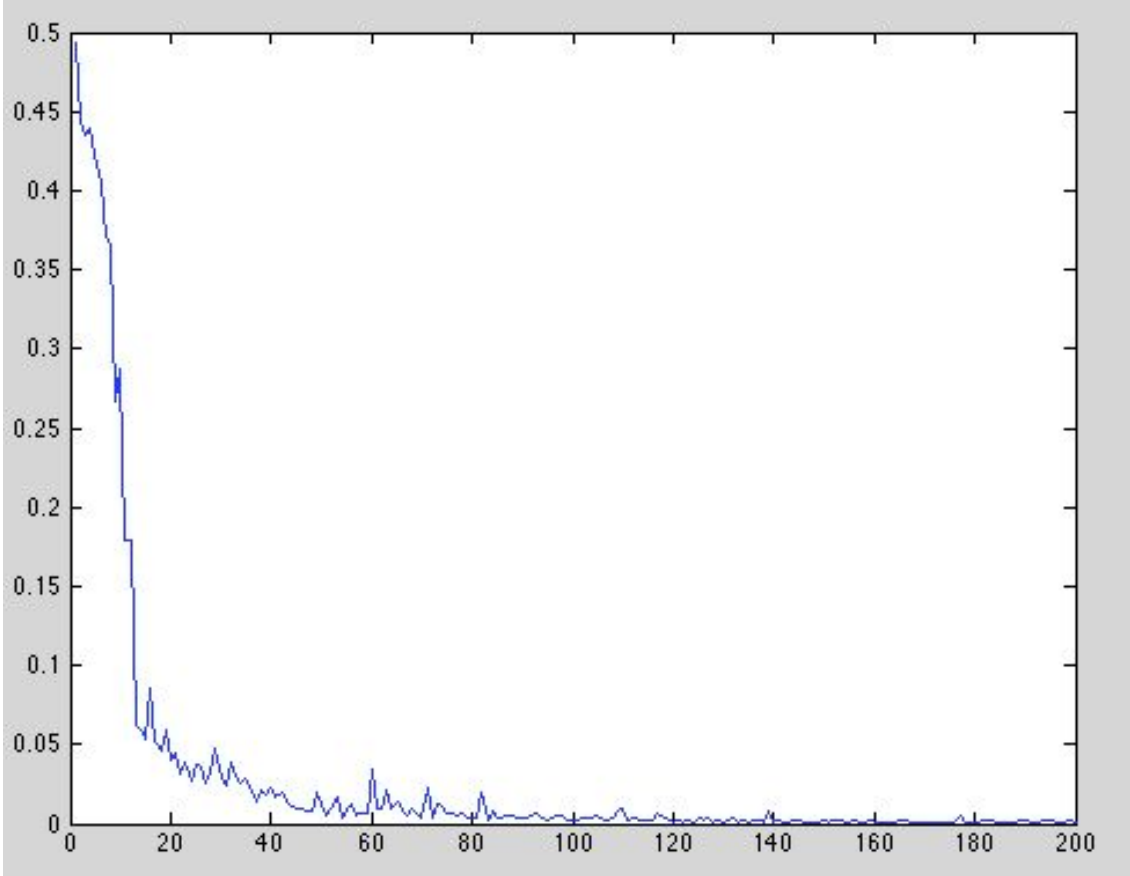
Imágen 8: Gráfico de los 300 valores de testeo obtenidos y esperados (normalizados) para la configuración final elegida.

El siguiente gráfico muestra la evolución del ECM para una de las corridas en función de la época.



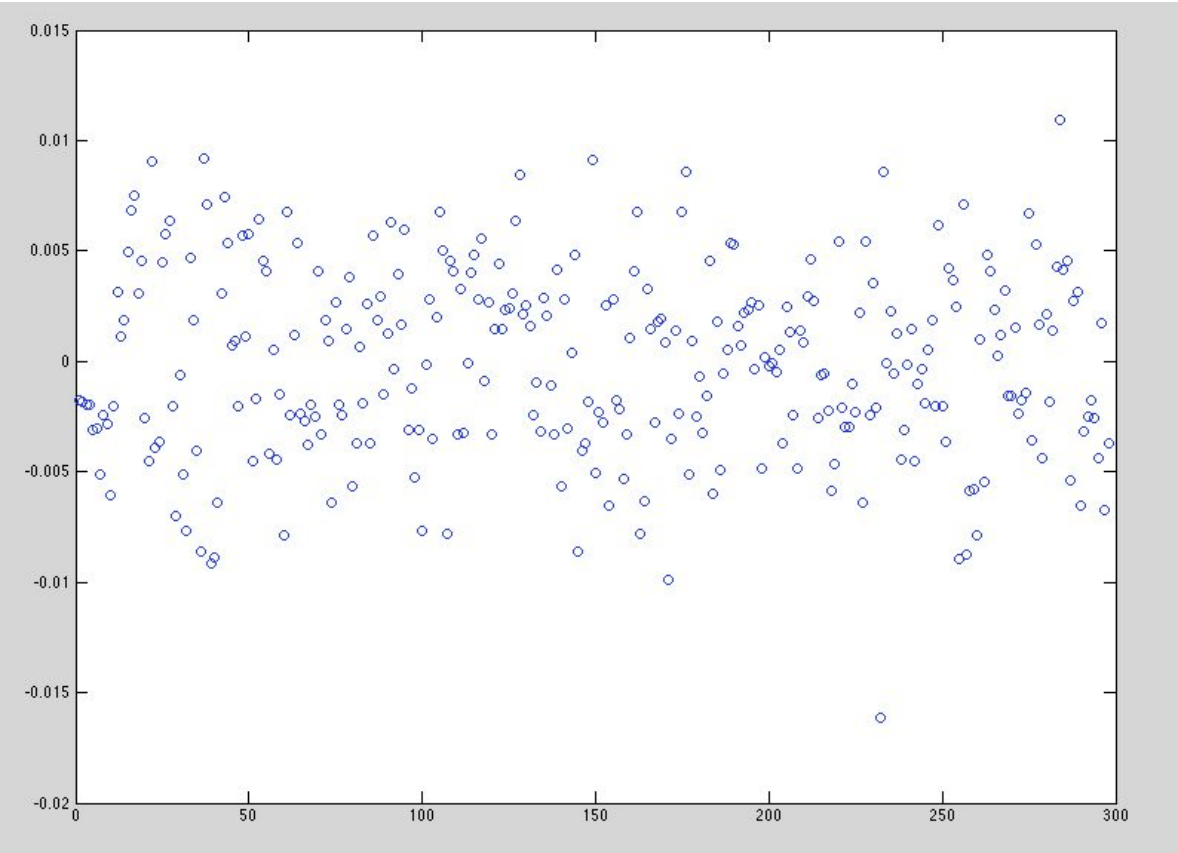
Imágen 9: Gráfico del ECM en función de la época.

Zoom en las primeras 200 épocas.



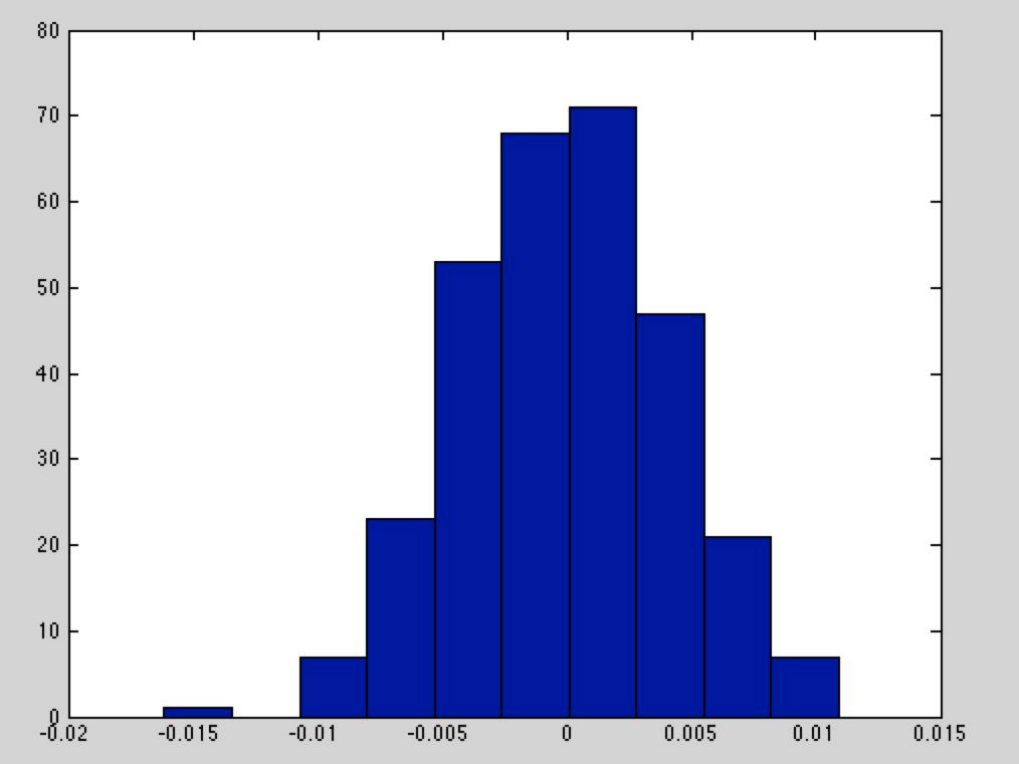
Imágen 10: ZOOM en las primeras 200 épocas: Gráfico del ECM en función de la época.

Dispersión de los errores en los 300 patrones de testeo.



Imágen 11: Dispersión de los errores en los patrones de testeo.

Histograma de errores en los patrones de testeo.



Imágen 12: Histograma de errores en los patrones de testeo.