Redes Neuronales

Sistemas de Inteligencia Artificial

Trabajo Práctico nº2 – 1er Cuatrimestre 2016

Alumnos

- Agopian, Michel -54325-
- Rossi, Melisa Anabella -54265-
- Zannini, Franco Michel -54080-

Grupo nº 4

Fecha de entrega 23 mayo de 2016

Índice

Introducción

El objetivo del trabajo es la implementación de una red neuronal multicapa, la cual, tenga la capacidad de aproximar los puntos de un terreno dado por la cátedra.

Se analizarán los errores obtenidos variando la estructura de la red, su configuración y método de aprendizaje, intentando llegar a un caso óptimo.

Análisis del terreno

Antes de comenzar, se graficaron los puntos del archivo "terrain8.txt" para tener una noción de como es el terreno elegido para el trabajo.

Como se observa en las figuras 1 y 2 (ver anexo), el terreno no posee simetrías así como tampoco zonas planas. Sin embargo, se puede encontrar cierto patrón al que responden los puntos. Si se observan las figuras, podemos observar que todos los puntos que responden a un mismo valor para x, cumplen con una función senoidal particular, que posee determinada amplitud. En los valores de x más cercanos al centro de nuestro dominio, podemos observar que la función senoidal a la que responde cada uno de ellos cambia su signo con respecto a los valores de x más cercanos a los extremos. También, se observa que cada una de estas funciones senoidales varía en su amplitud.

Conjunto de entrenamiento

Para el entrenamiento de la red neuronal se formaron tres conjuntos diferentes con el fin de determinar diferencias tanto en los errores obtenidos durante el entrenamiento, así como también en la capacidad de generalización que tiene la misma.

En primer lugar, para poder pasar a la creación de estos tres conjuntos, se comenzó por ordenar los puntos ascendentemente por su x, para luego poder ordenarlos, para cada x, por su y (también de manera ascendente). De esta forma, se tiene los puntos del terreno de forma ordenada.

Para formar el primer conjunto de puntos de entrenamiento, se recorrió la lista ordenada de puntos y se tomó uno de cada dos. Es decir, se eligieron aquellos puntos que se encuentran en una posición que sea múltiplo de dos.

Para el segundo conjunto se realizó un procedimiento similar al anterior, pero esta vez eligiendo uno de cada cuatro puntos, tomando los puntos que se encuentran en una posición múltiplo de cuatro. Esto permitiría seguir contando con una distribución uniforme, pero con muchos menos puntos, para poder observar si después de entrenar a la red contamos con un error de generalización distinto al del primer conjunto.

Como última implementación, para el último conjunto de puntos de entrenamiento, se decidió seleccionar los puntos de a "tiras de x". Basándonos en lo previamente mencionado en el análisis del terreno, contamos con distintas "tiras de x", cada una de la cual responde a una función senoidal. Estas distintas funciones senoidales van variando en amplitud de manera progresiva, por lo que se decidió tomar una de cada dos "tiras de x" y observar si las "tiras faltantes" eran aprendidas de manera correcta.

Tanto el primero como el tercer conjunto de puntos, utilizan la mitad de la totalidad de los puntos conocidos para el entrenamiento, mientras que el segundo conjunto utiliza tan solo un cuarto de la totalidad de puntos.

Luego de sucesivas pruebas para los distintos conjuntos, se pudo observar que una vez obtenido el error máximo deseado para dicho conjunto de entrenamiento, el error de generalización arrojado era similar para los tres conjuntos. Por este motivo, se decidió operar con el segundo conjunto de entrenamiento, el cual cuenta con mucho menos puntos que los otros dos conjuntos (en particular la mitad), lo que hace mucho más rápido la ejecución de algunos algoritmos implementados.

Método de entrada

Para correr los algoritmos se deberá llamar a la función terrain_training_test de la siguiente manera

terrain_training_test(net_structure, error, g, g_der, n, betha, learningType, algorithm, graphics, alpha, a, b, K),

donde :

- net_structure : array de enteros indicando para cada capa la cantidad de neuronas que posee.
- error : el error máximo que debe tener el entrenamiento.
- g : función de activación.
- g_der : derivada de la función de activación.
- n : valor de etha.
- betha : parámetro utilizado en la función de activación.
- learningType : 1 (batch) o 2 (incremental).
- algorithm : 1 (original), 2 (momentum) o 3 (adaptative etha).
- graphics : valor booleano indicando si graficar el error a medida que avanza el algoritmo.
- alpha : valor utilizado para modificación de momentum.
- a : valor en el que se aumenta etha en modificación de adaptative etha.
- b : porcentaje en el que se disminuye etha en modificación de adaptative etha.
- K : cantidad de pasos positivos consecutivos antes de modificar el etha en modificación de adaptative etha.

Dicha función retornará 2 parámetros. En primer lugar, una matriz que representa la generalización de los puntos realizada luego del entrenamiento y, en segundo lugar, las distintas subredes, con sus respectivos pesos, que se obtuvieron en el último paso del algoritmo.

Ejemplo de prueba :

[generalization, nets] = terrain_training_test([2 17 1], 0.0005, @tanh_ft, @tanh_ft_der, 0.2, 0.2, 1, 2, false, 0.9)

Implementación

Con respecto a la implementación de la red, es importante destacar que se definió como aceptable, un error menor a 1E-3.

A la hora de elegir la estructura, experimentalmente se logró obtener redes de una única capa oculta que logren obtener un error de generalización aceptable.

Modificaciones al algoritmo original

Como primera modificación, en el método de aprendizaje incremental, los patrones a analizar se tomarán de manera aleatoria para que el entrenamiento de la red no se de siempre en el mismo orden. Para el método de aprendizaje batch, esta modificación no se realizó ya que no tiene sentido debido a que los pesos de la matriz que hay entre las distintas capas se modifican una vez que se entrenó una época completa.

Por otro lado, tanto para el método batch, como para el incremental, se implementaron las variaciones a back propagation de momentum y etha adaptativo.

Algoritmos con momentum

Para la mejora de momentum se realizó una implementación que recibe como parámetro adicional un valor alpha.

El objetivo principal de esta mejora es que, luego de cada iteración completa por una época en batch, o luego de iterar en cada patrón en incremental, las subredes que existen entre las distintas capas no se vean solamente afectadas por este paso del algoritmo, sino que se vean también influenciadas por la variación sufrida en el paso anterior. Para cumplir con este objetivo, a la variación de cada subred se le sumará la variación que sufrió en el paso anterior multiplicado por el parámetro alpha. De esta manera, se intenta que las variaciones sigan una tendencia.

Algoritmos con Etha adaptativo

Para la mejora de back propagation con etha adaptativo se decidió realizar una implementación que recibe como parámetro 4 valores importantes:

- K : número de pasos positivos consecutivos que deben suceder antes de aumentar el etha.
- a : valor con el cual se aumentará el etha de la manera `etha + a`.
- b : valor con el que se disminuye el etha de la manera `etha - b * etha`.
- alpha : parámetro del momentum.

En esta mejora, cada vez que se finaliza una época, se compara el nuevo error calculado con el obtenido por haber ejecutado el algoritmo en la época anterior. Si el error actual es menor al anterior, se aumentará un contador. Si el contador se aumenta K veces consecutivas, entonces se aumenta el etha en un valor a.

Sin embargo, podría pasar que al calcular el error de la época actual, el mismo sea más grande que el error obtenido en la época anterior. En ese caso, se realizara un paso de prueba (iterar en una época el algoritmo) para comparar el error obtenido en dicho paso con el actual. Como se sabe que el paso actual no fue un paso bueno, se setea alpha en 0 para que la próxima época no se vea afectada por el paso malo. Si el error obtenido en el paso de prueba es menor al obtenido 2 pasos anteriores, es decir, menor al paso anterior al paso malo, se continúa con el algoritmo normalmente. Si el error obtenido en el paso de prueba es mayor, entonces se volverá al último paso bueno y se decrementara etha en b*etha.

Exponencial vs Tangente Hiperbólica

Al analizar entre ambas funciones cual resulta más beneficiosa a la hora del aprendizaje, luego de varias pruebas para diferentes algoritmos notamos que, si bien el error de exponencial era muchas veces menor al obtenido con la tangente hiperbólica, la comparación de ambos errores carecía de sentido.

Para errores iguales, en los casos de tangente hiperbólica la red aprendía de manera correcta mientras que para exponencial los puntos se mostraban como planos con muy pocas variaciones. Esto se debe a que al normalizar, el dominio de la exponencial era menor (exactamente la mitad) generando que la distancia entre valores normalizados no sea tan significante como lo es en la tangente hiperbólica, produciendo así errores mas pequeños pero no necesariamente mejores.

Teniendo en cuenta que la tangente hiperbólica nos daba un nivel mayor de precisión en los errores y, que luego de varias pruebas experimentales no se logró obtener una mejora de aprendizaje utilizando la exponencial, se decidió optar por la tangente hiperbólica como función de activación.

Resultados

Para el análisis de los errores arrojados por los seis algoritmos, se decidió ejecutar cada uno de los mismos durante 20000 épocas, variando el número de neuronas en la capa oculta, de 1 hasta 20 inclusive, tomando nota de los errores de entrenamiento y de generalización que se tenían hasta dicho punto.

Por otra parte, se decidió repetir este proceso para cuatro configuraciones distintas, las cuales fueron seleccionadas a partir de pruebas experimentales, ya que permitieron obtener errores aceptables en las ejecuciones de los algoritmos. Las mismas fueron:

A. etha: 0.3, betha: 0.2, alpha: 0.9, a: 0.2, b: 0.1, K: 5;
B. etha: 0.5, betha: 0.4, alpha: 0.9, a: 0.2, b: 0.05, K: 11;
C. etha: 0.3, betha: 0.5, alpha: 0.9, a: 0.2, b: 0.1, K: 5; y
D. etha: 0.2, betha: 0.2, alpha: 0.9, a: 0.2, b: 0.1, K: 5.

Habiendo corrido el proceso para las cuatro configuraciones, basándonos en los cuadros de resultados 1, 2, 3 y 4 (ver anexo), obtenemos las siguientes conclusiones luego de 20000 etapas:

El error mínimo se obtuvo con la mejora de MOMENTUM para el método BATCH, utilizando 17 neuronas en la capa oculta, con la configuración mencionada como d.

Error: 4.7754e-5 | 5.6950e-5. (error en conjunto de entrenamiento | error en generalización).

En segundo lugar, se posiciona el resultado arrojado por el algoritmo INCREMENTAL original, utilizando 11 neuronas en la capa oculta, con la configuración mencionada como c.

Error: 1.1521e-04 | 1.3232e-04. (error en conjunto de entrenamiento | error en generalización).

Luego, nos encontramos con el resultado arrojado por el algoritmo BATCH original, utilizando 7 neuronas en la capa oculta, con la configuración mencionada como d.

Error: 6.1401e-4 | 5.1996e-4. (error en conjunto de entrenamiento | error en generalización).

En último lugar se encuentra la modificación de ETHA ADAPTATIVO que se hizo en el algoritmo de BATCH, utilizando 12 neuronas en la capa oculta con la configuración mencionada como b.

Error: 7.5033e-4 | 6.3869e-4. (error en conjunto de entrenamiento | error en generalización).

Cabe destacar que mientras se ejecutaba el proceso que luego serviría para realizar el análisis de los resultados, se detectó un error en la implementación de las modificaciones de etha adaptativo y momentum del algoritmo incremental. Una vez corregido dicho error, se volvió a correr el proceso para estos algoritmos, variando tanto las neuronas en la capa oculta así como también las distintas configuraciones. Al evaluar los resultados obtenidos, el error al que se llegaba era muy alto comparado al de los demás algoritmos (ver anexo tablas a, b, c y d), por lo que se decidió encontrar una nueva configuración capaz de encontrar un error similar al de los demás al cabo de 20000 etapas.

Dicha configuración se eligió, al igual que las anteriores, por buenos resultados en lo experimental. La misma cuenta con los valores de etha: 0.02, betha: 0.5, alpha: 0.9, a: 0.02, b: 0.1, K: 5.

El error mínimo para la modificación de MOMENTUM en el algoritmo INCREMENTAL, utilizando 11 neuronas en la capa oculta, fue de 1.414e-4 | 1.542 e-4 (error en conjunto de entrenamiento | error en generalización).

En el caso de la modificación de ETHA ADAPTATIVO también se obtuvo con 11 neuronas en la capa oculta, encontrando un error de: 7.8636e-04| 6.4465e-04 (error en conjunto de entrenamiento | error en generalización).

De esta manera, podemos ahora ubicar en tercera posición a la modificación momentum del algoritmo incremental y, en último lugar, a la modificación de etha adaptativo del algoritmo incremental.

Conclusiones

Como una primera conclusión, si observamos las figuras 4 y 11, que muestran la evolución del error a lo largo de las 20000 épocas para los algoritmos de batch e incremental originales, podemos observar que se obtienen errores similares al finalizar el proceso. Sin embargo, se puede apreciar que el algoritmo incremental obtiene un error mucho más bajo al poco tiempo de iniciado el proceso, mientras que el de batch consigue un error bajo recién después de recorrer, al menos, 5000 épocas. En las figuras 3 y 10, se comparan cada uno de los puntos obtenidos por los algoritmos (batch e incremental, respectivamente) con los puntos originales en color blanco, mostrando la similitud en el terreno obtenido.

Al analizar la mejora de momentum para estos algoritmos, encontramos que la tendencia obtenida para el error es la misma. Si bien la comparación entre incremental original e incremental con momentum no muestra más que una leve mejora en el error y en cuando comienza a descender el mismo (ver figuras 10 y 12), si se compara la implementación de batch original con la de batch con momentum, podemos apreciar una gran mejoría en cuanto al descenso del error, asi como tambien en el error final obtenido (ver figuras 4 y 6). Una vez más, en las figuras 5 y 12, se podrán comparar los puntos obtenidos por los algoritmos comparados con los puntos originales en color blanco.

Con respecto a las modificaciones de etha adaptativo, podemos observar que debido a la implementación realizada, tanto para batch como para incremental, se observan notorias fluctuaciones (ver figuras 8 y 15). Si bien la tendencia que sigue el error es similar a la de los demás algoritmos, las fluctuaciones hacen más lento el proceso, obteniendo los errores más altos al finalizar las corridas por 20000 épocas completas. Aquí también se podrán comparar los puntos obtenidos por los algoritmos comparados con los puntos originales en color blanco (ver figuras 7 y 14).

Como se mencionó previamente a lo largo del trabajo, el terreno se lo puede ver cómo “tiras de x”, donde cada “tira” representa una función senoidal. Existen dos de estas “tiras” que poseen una amplitud mínima y se ven más reflejadas como una recta casi perfecta. Si el error que se utiliza para aproximar el terreno no es lo suficientemente exigente, dichas “tiras” que se ven como rectas, serán representadas como funciones senoidales de amplitud muy pequeña, en lugar de verse como “casi rectas” (ver figura 16).

Por último, una particularidad del terreno es, que debido a su configuración y disposición de los puntos, el error de generalización sigue la misma tendencia que el error para el conjunto de entrenamiento (ver figura 17), lo que nos obligo a encontrar un punto de corte distinto que, en nuestro caso, fue 20000 épocas.

Anexo

Figuras

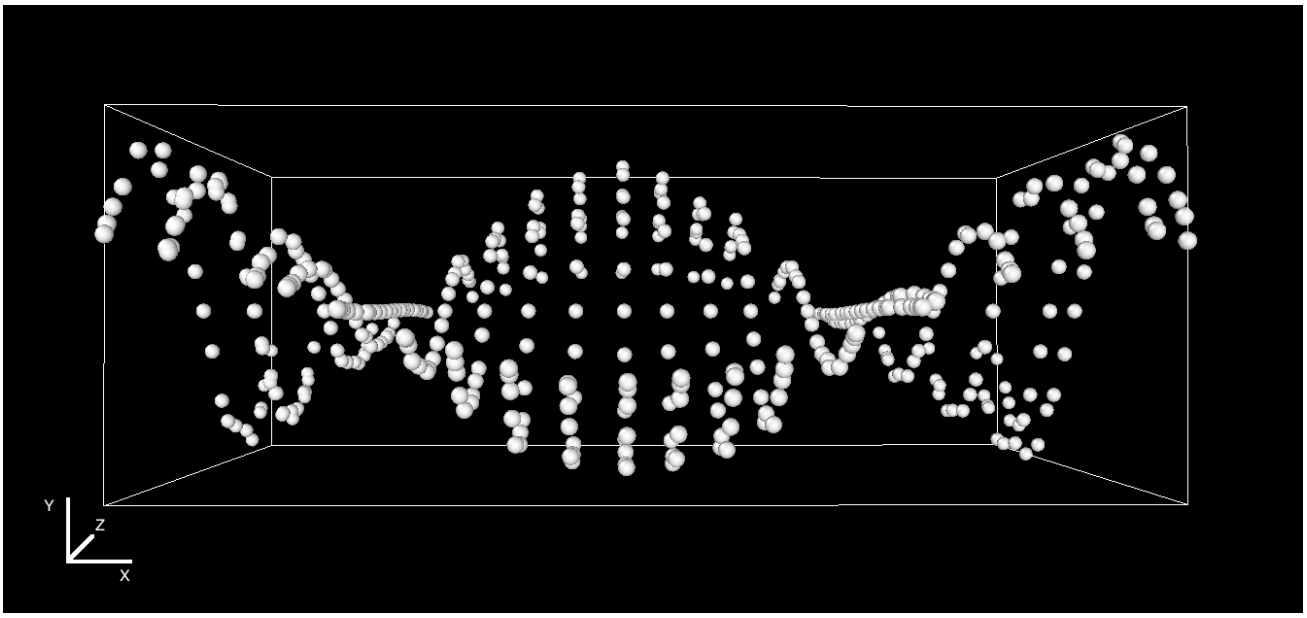

Figura 1 – terrain8.txt

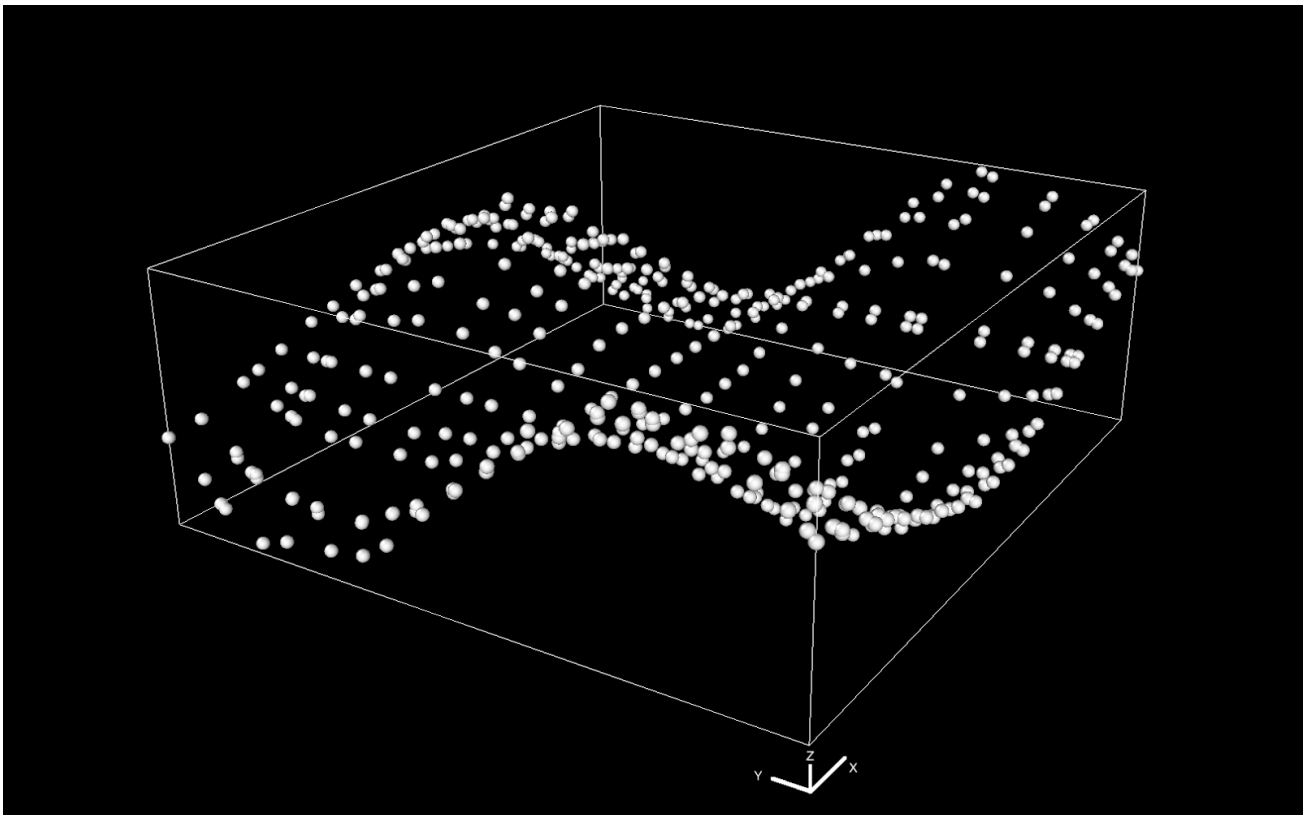

Figura 2 – terrain8.txt

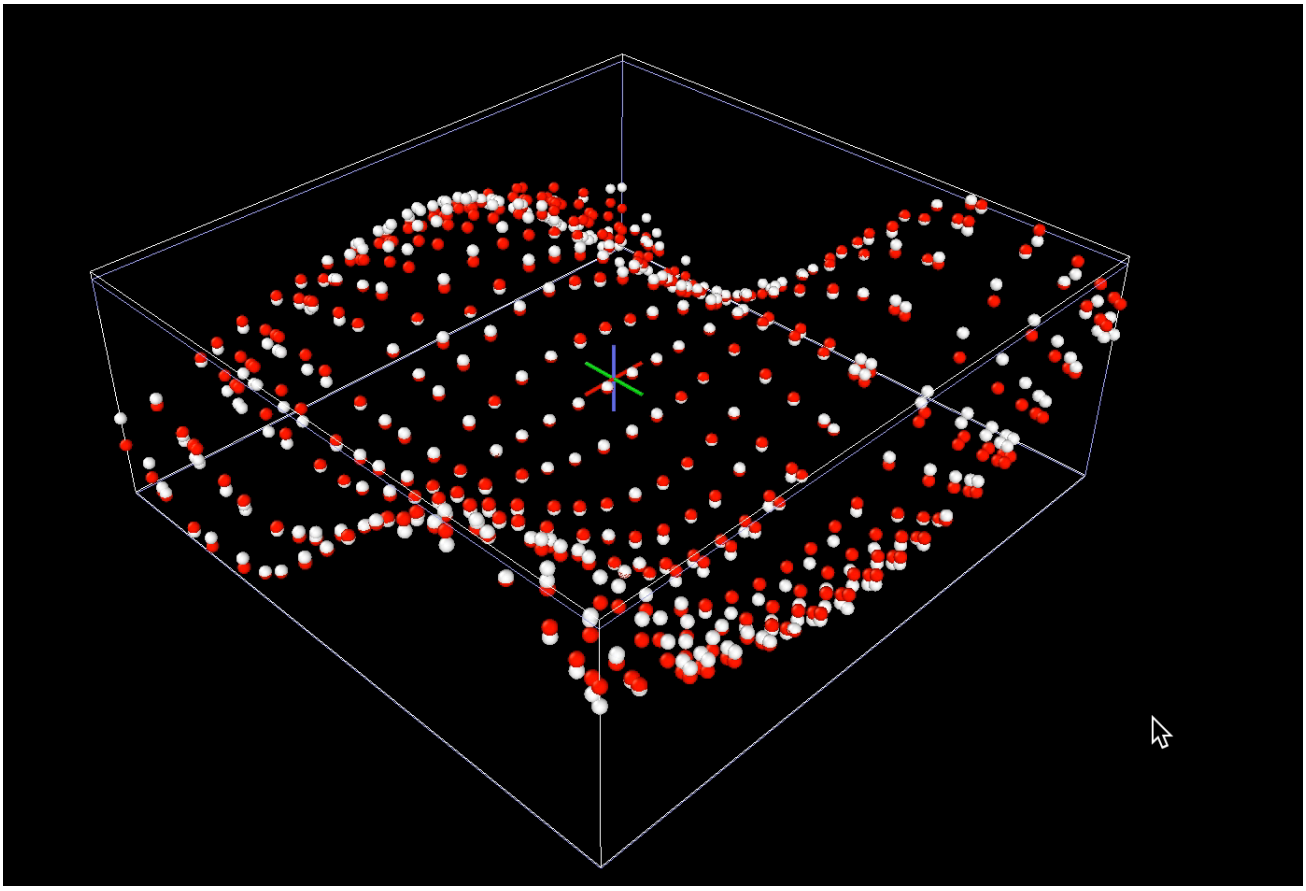

Figura 3 – aprendizaje con batch

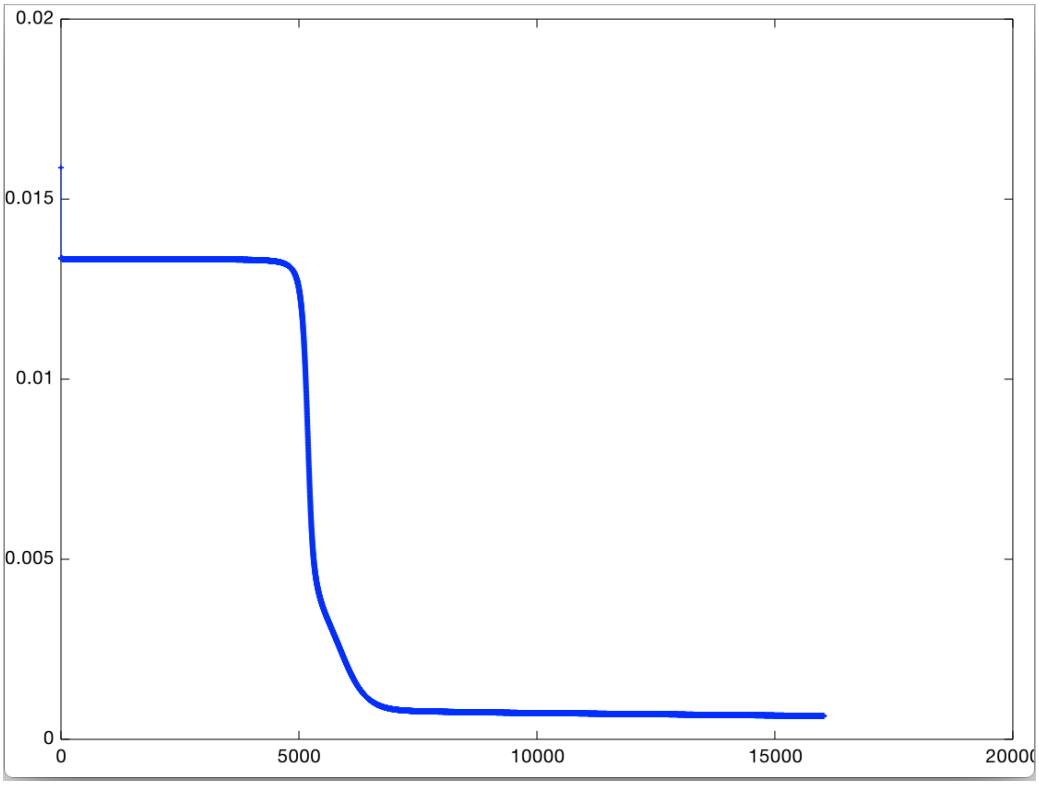

Figura 4 – error de batch

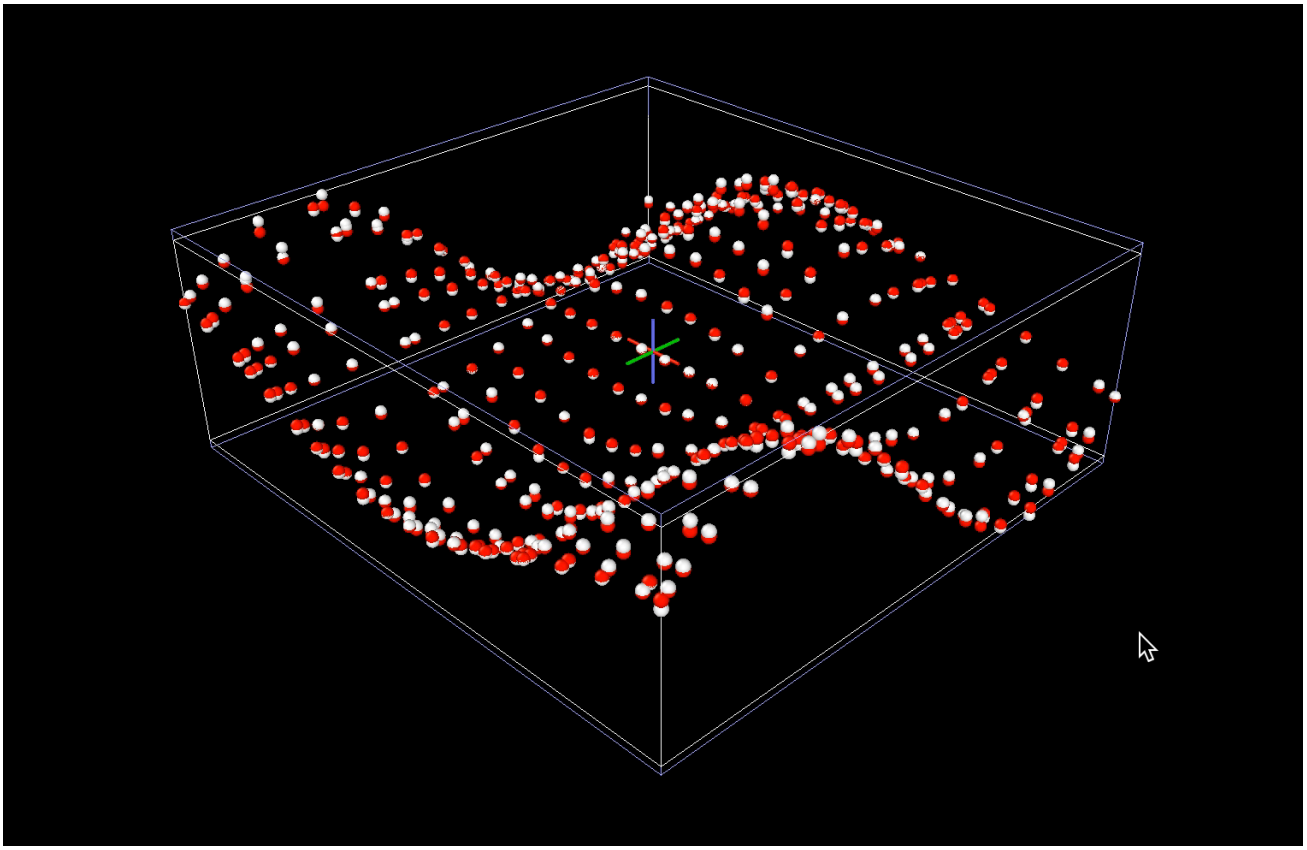

Figura 5 – aprendizaje con batch momentum

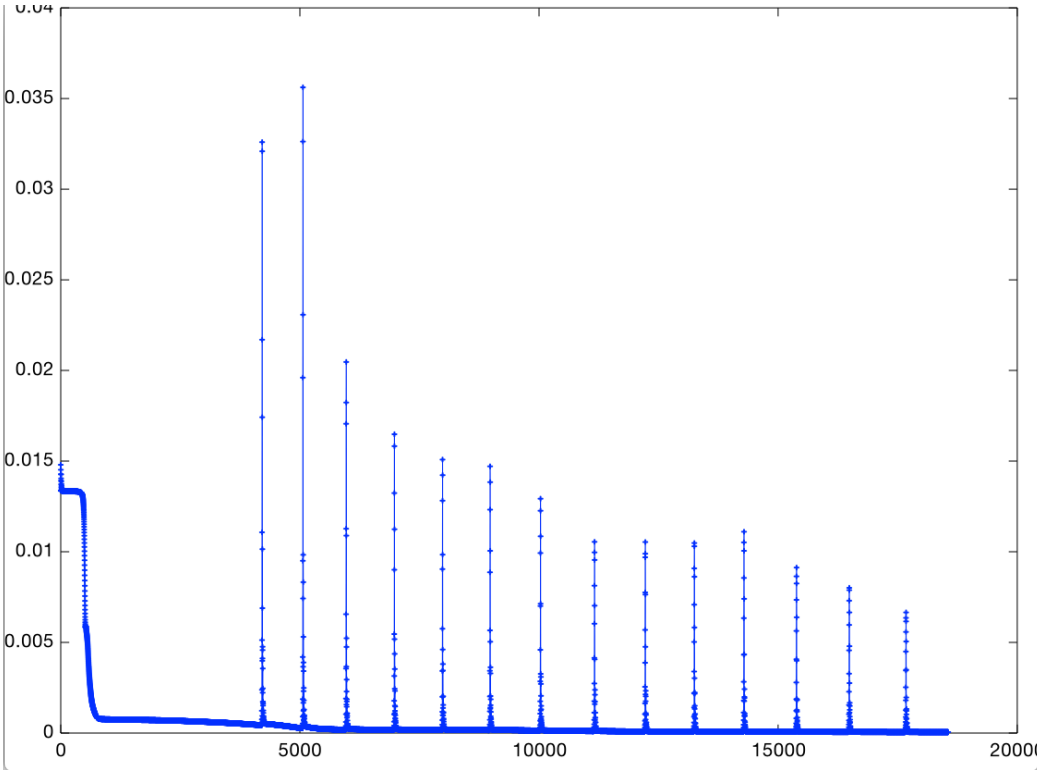

Figura 6 – error con batch momentum

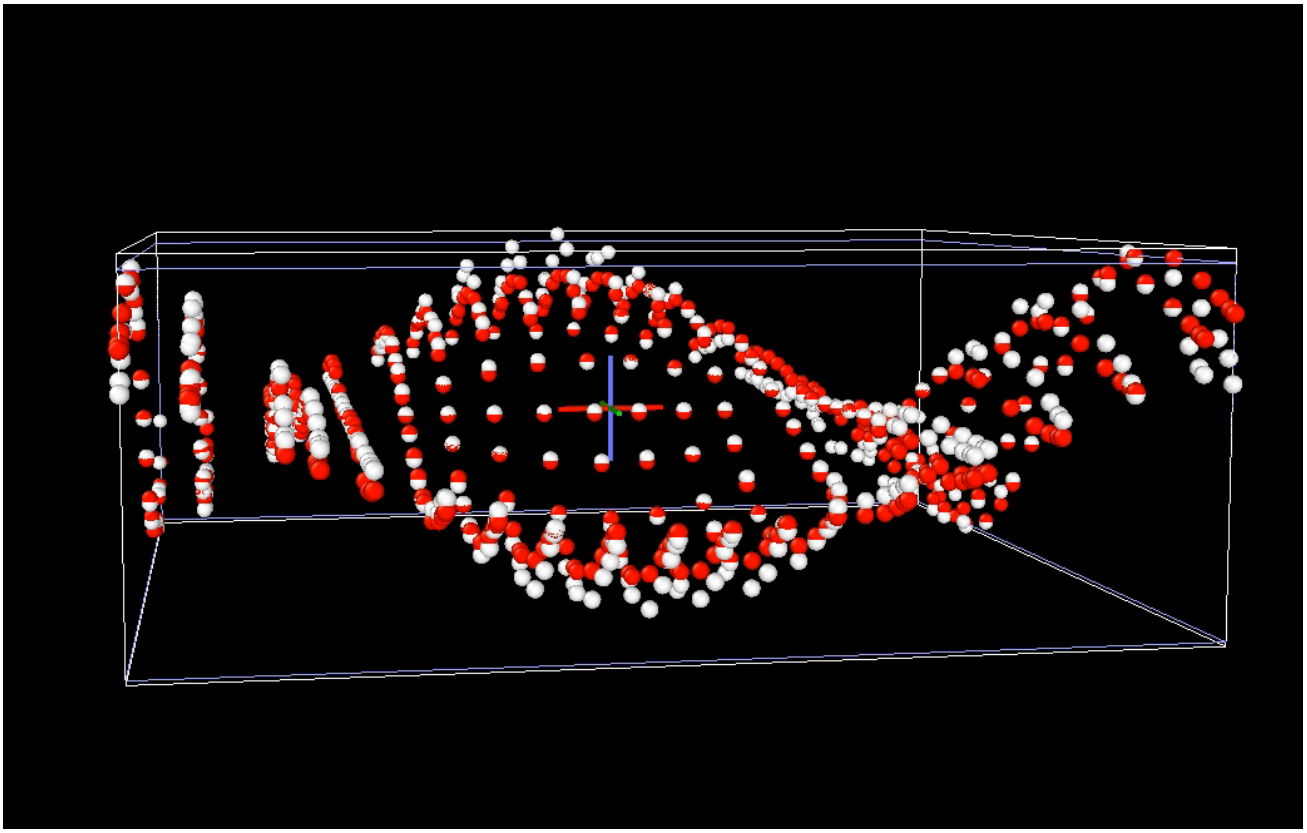

Figura 7 – aprendizaje batch con etha adaptativo

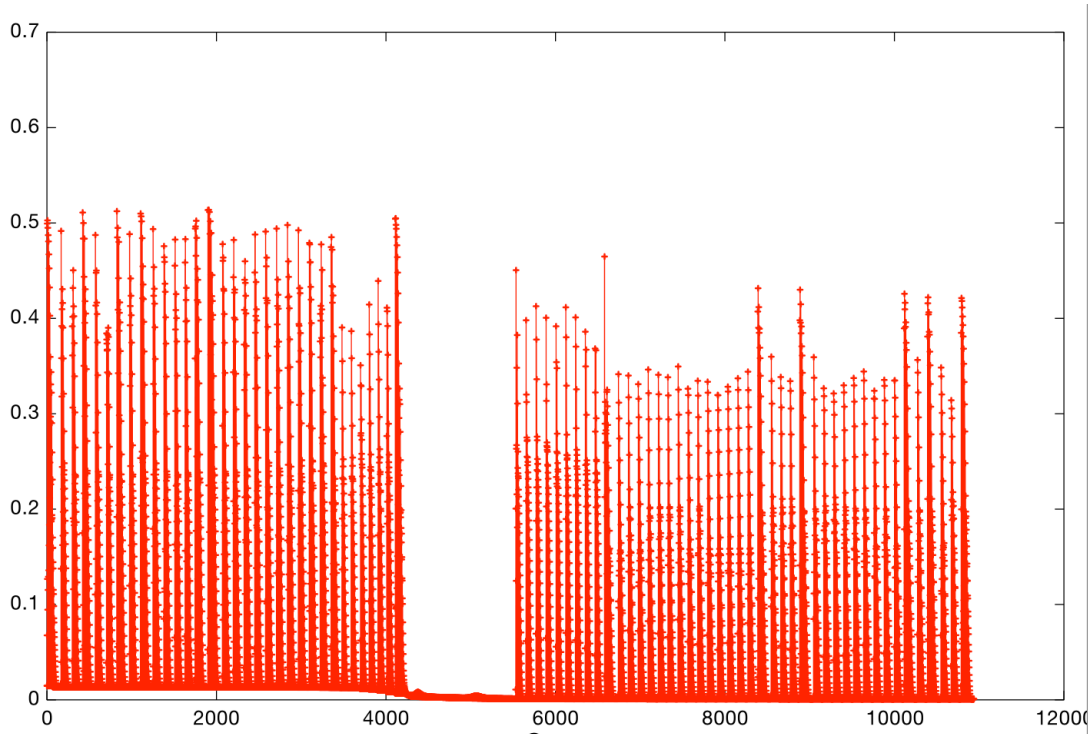

Figura 8 –error batch con etha adaptativo

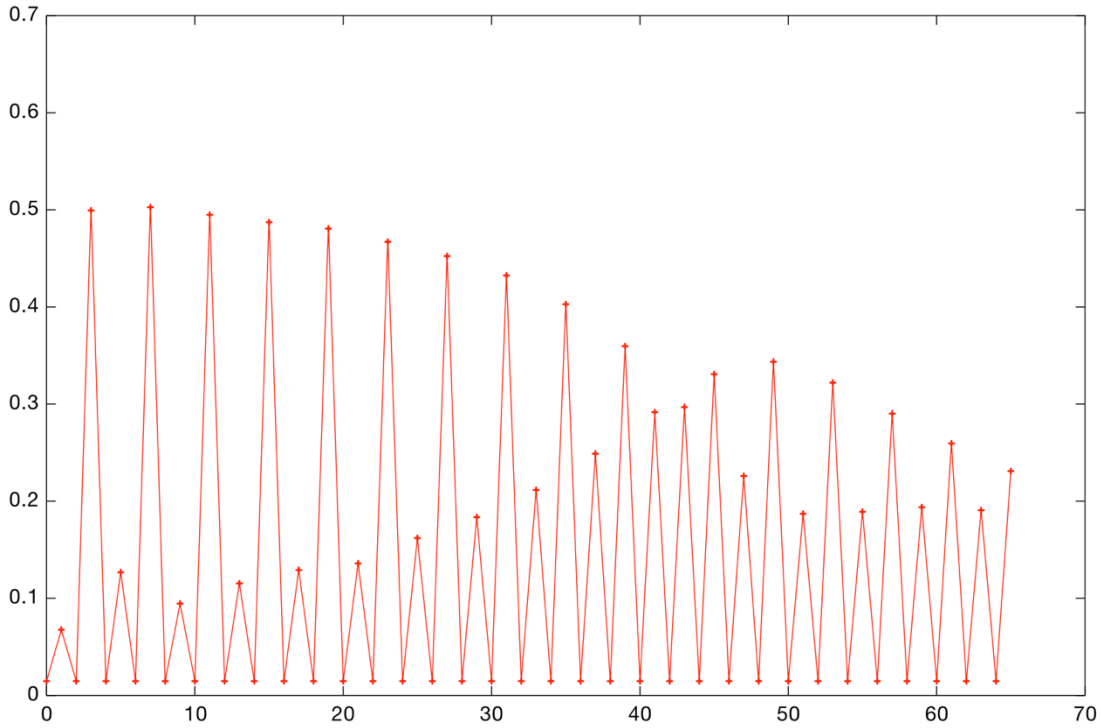

Figura 9 – zoom error batch con etha adaptativo

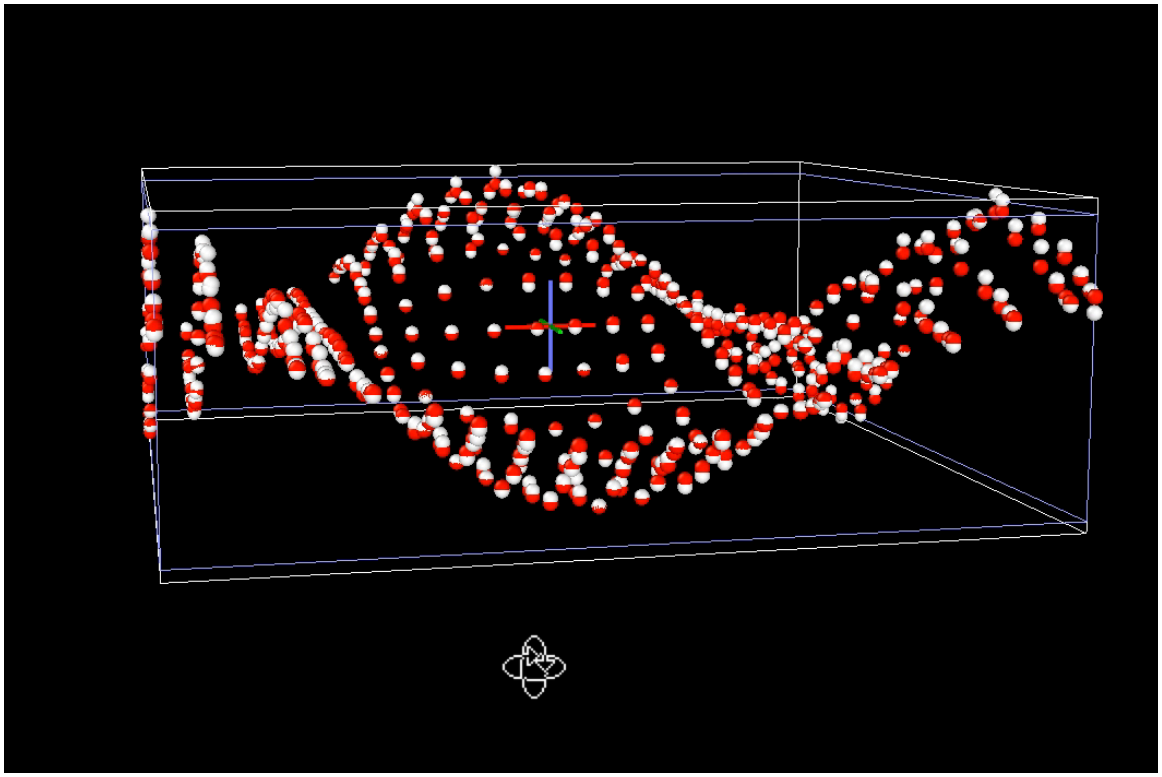

Figura 10 – aprendizaje incremental

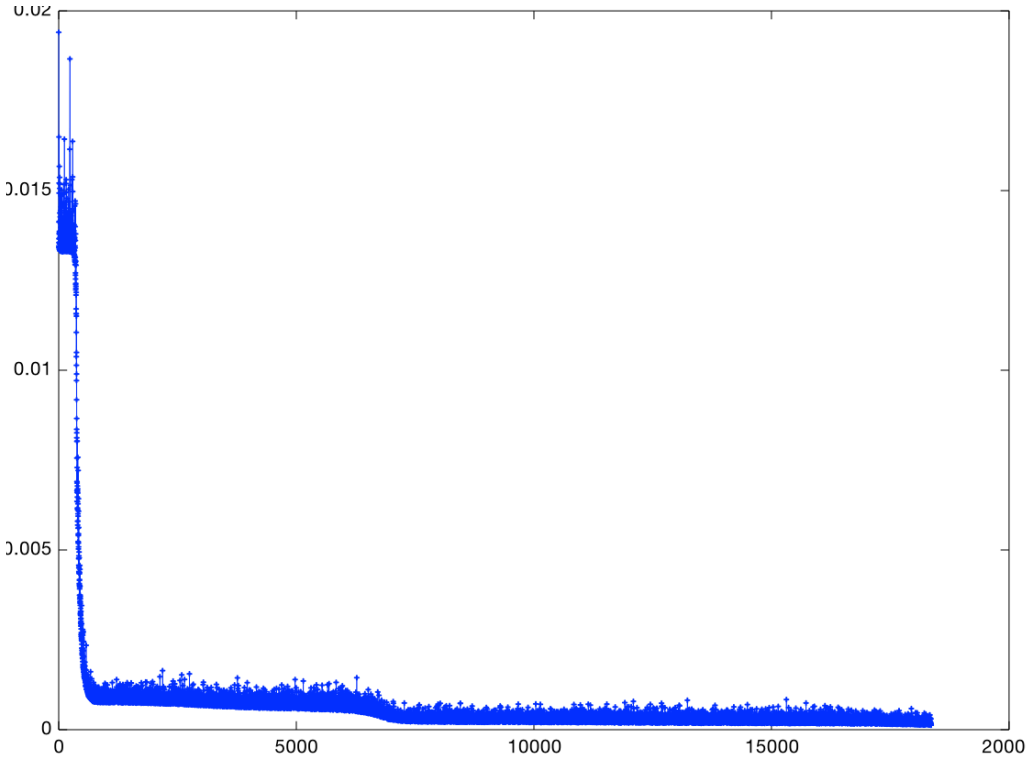

Figura 11 – error incremental

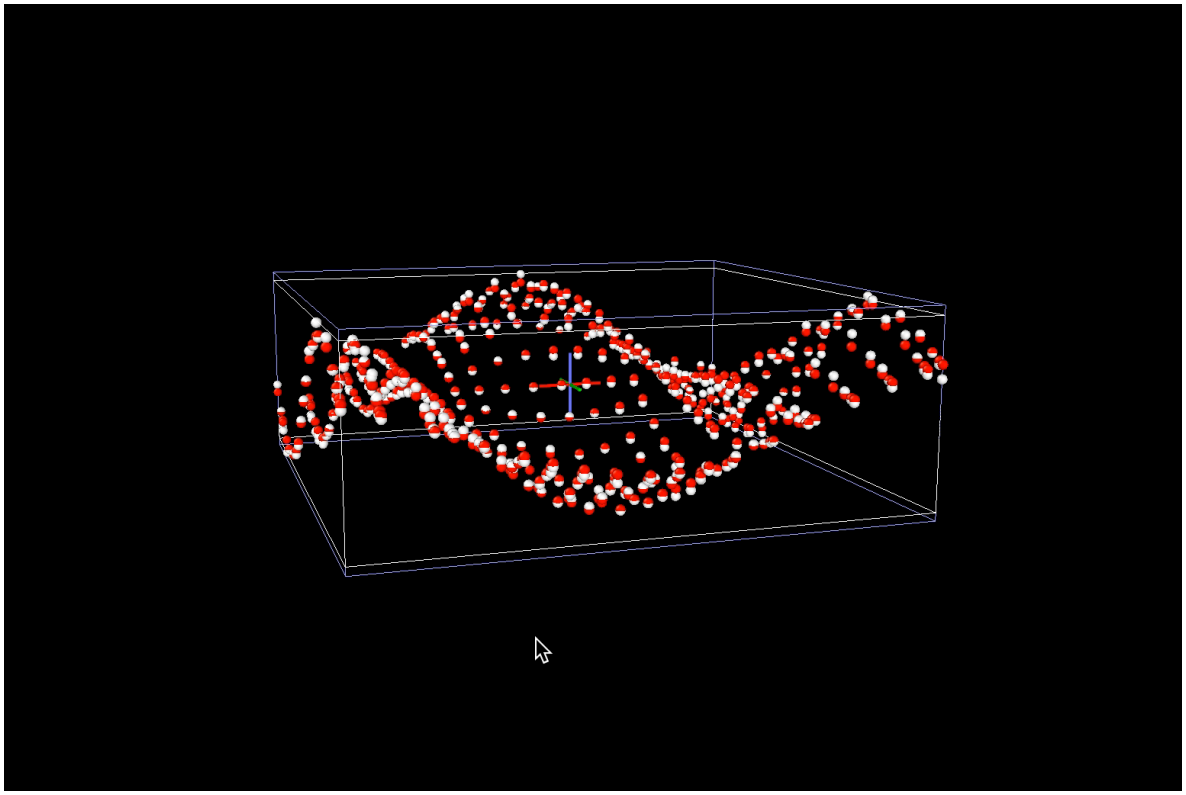

Figura 12 – aprendizaje incremental momentum

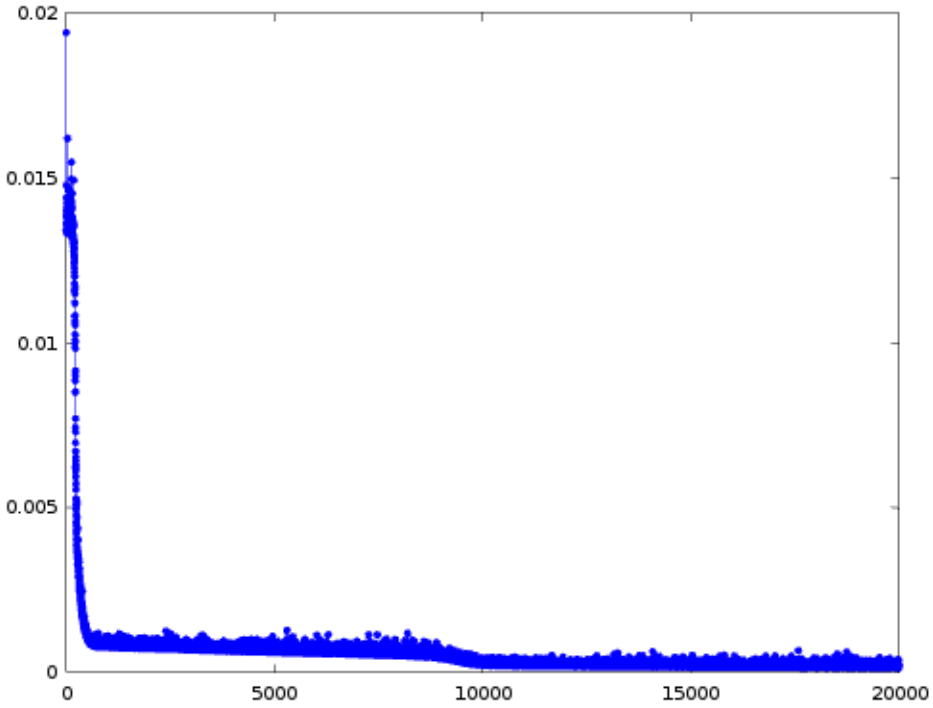

Figura 13 – error incremental momentum

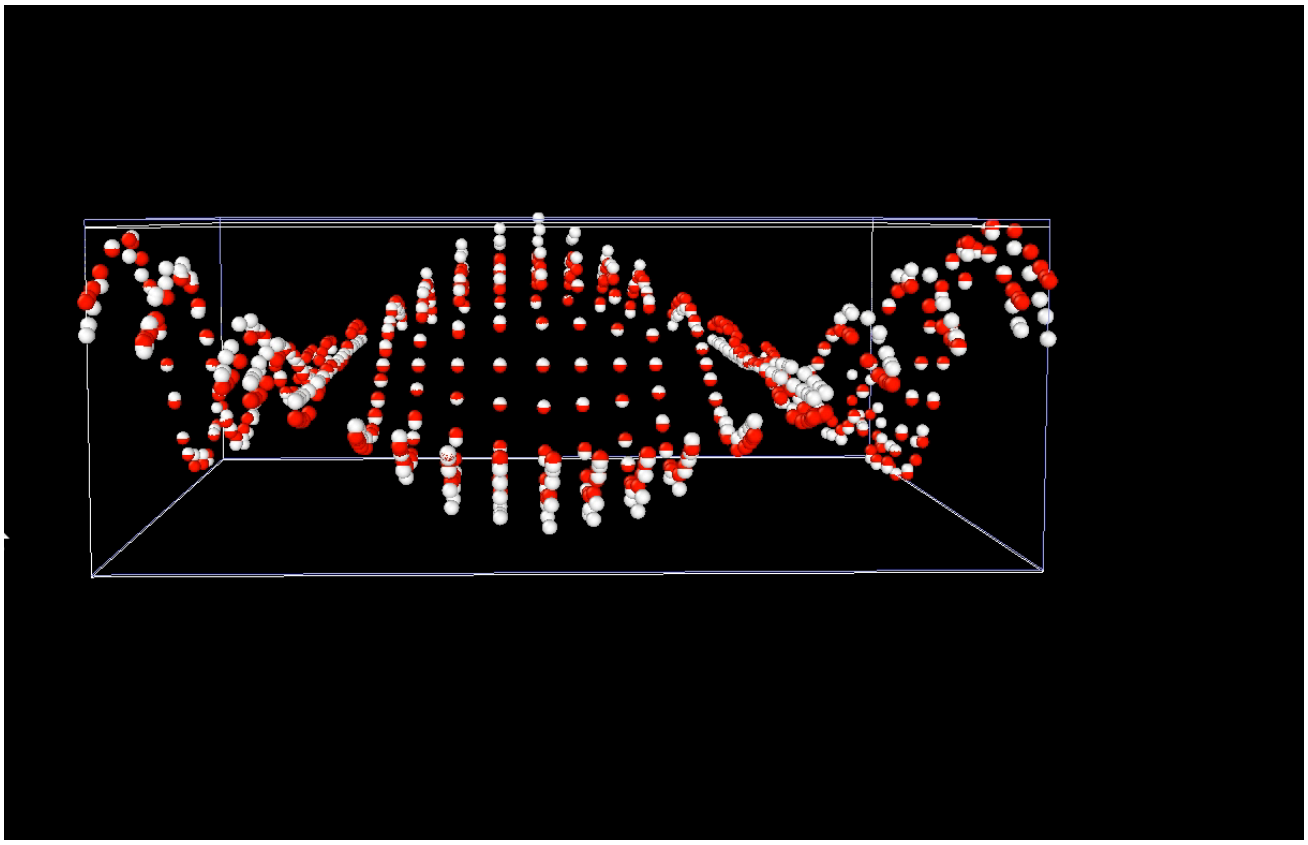

Figura 14 – aprendizaje incremental con etha adaptativo

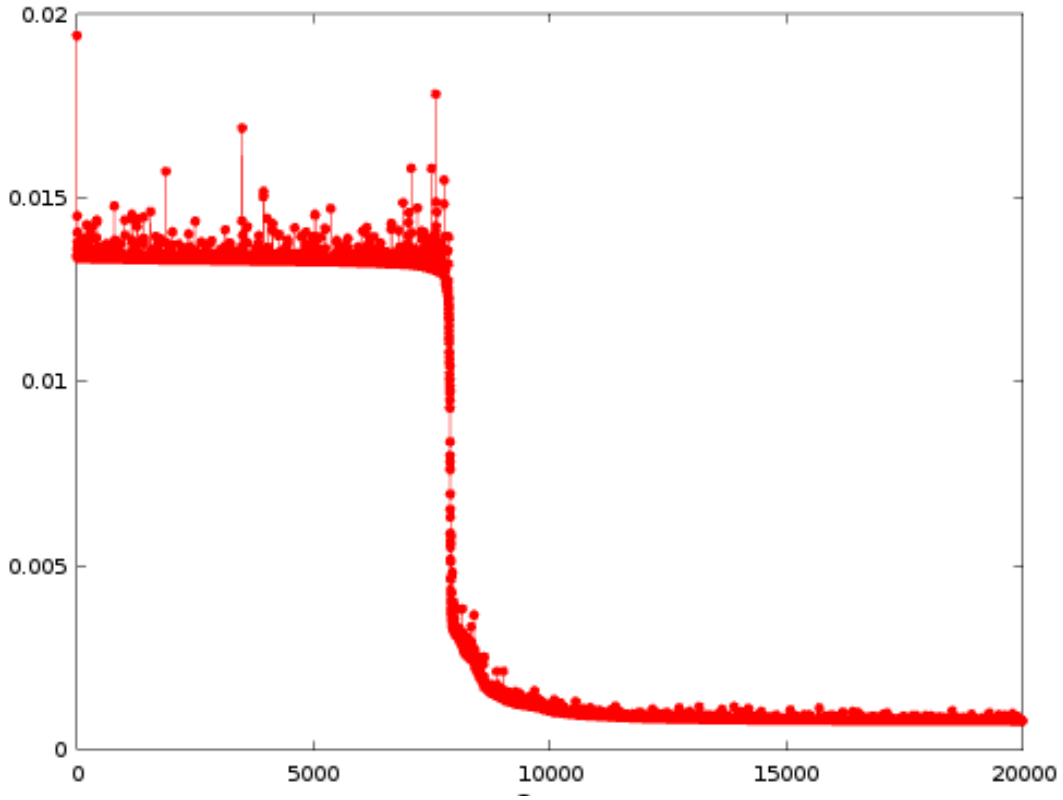

Figura 15 – error incremental con etha adaptativo

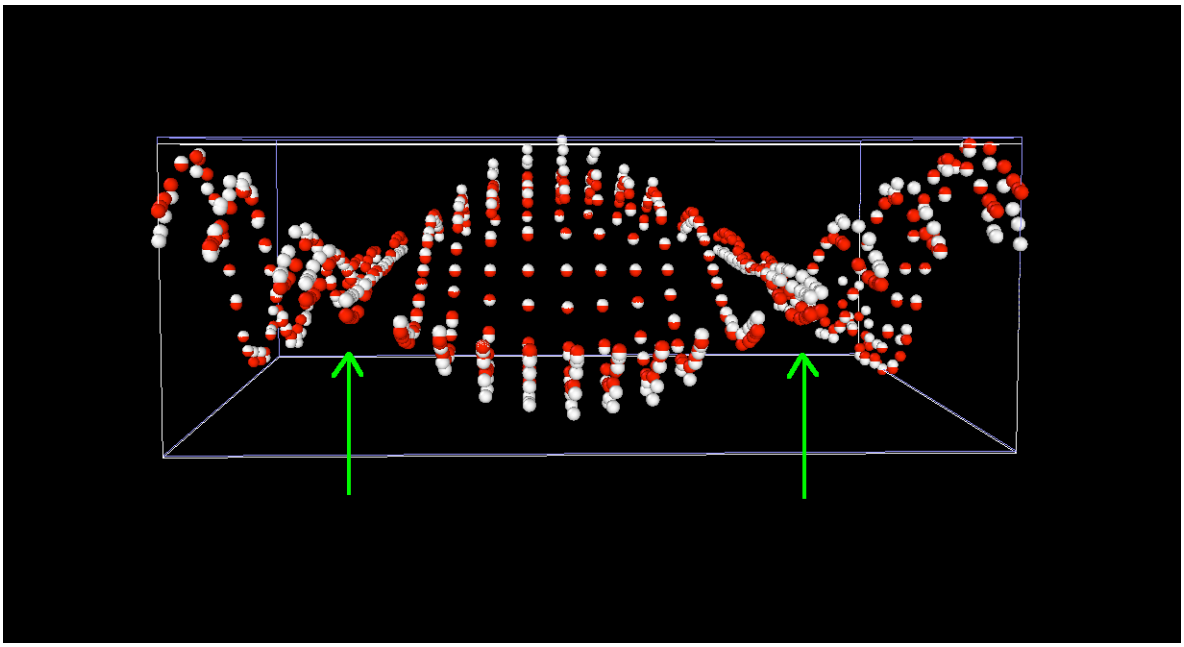

Figura 16 - comparación de generalización con puntos originales con error de 0.001

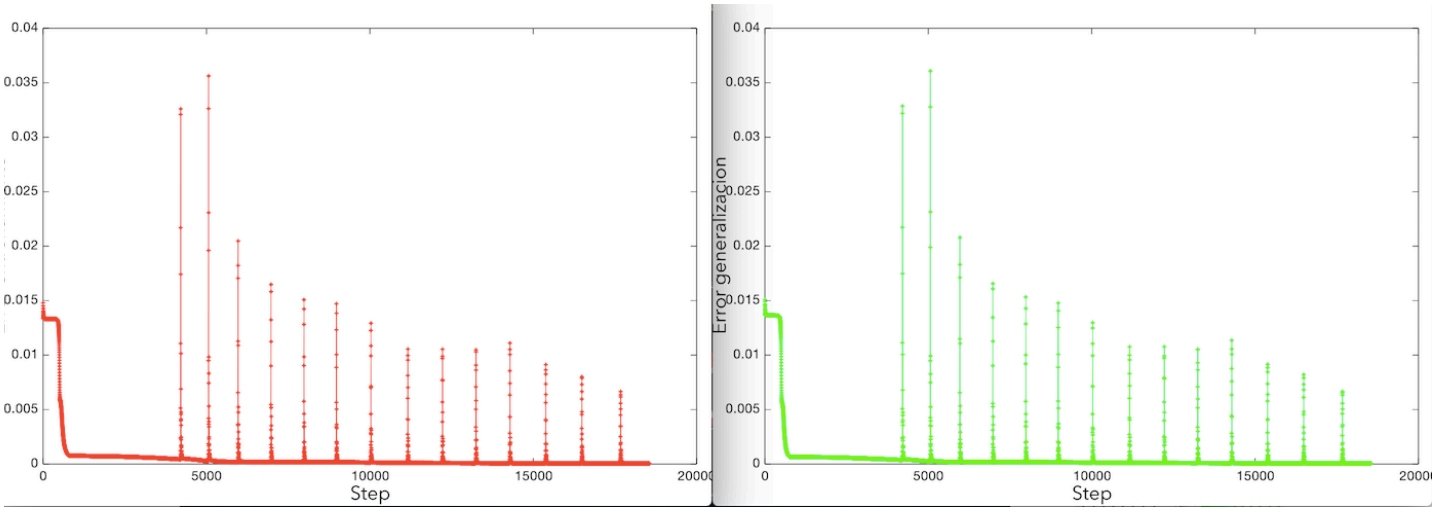

Figura 17 – Comparación de la evolución de el error de entrenamiento con el de generalización en batch momentum.

Tablas

Tabla A

Neuronas capa oculta	Incremental original	Incremental con momentum	Incremental con Etha adaptativo	Batch original	Batch con Momentum	Batch con Etha adaptativo
1	0.013335 | 0.013653	0.013474 | 0.013767	0.013333 | 0.013651	0.013332 | 0.013653	0.013332 | 0.013653	0.013332 | 0.013653
2	0.01336 | 0.013654	0.013339 | 0.013645	0.013334 | 0.013648	0.0069515 | 0.0071892	0.013332 | 0.013653	0.013332 | 0.013654
3	0.013361 | 0.013689	0.015617 | 0.015991	0.013333 | 0.013652	0.0017567 | 0.0016936	7.4881 e-4 | 6.4668 e-4	0.013329 | 0.013650
4	0.013332 | 0.013653	0.013428 | 0.013724	0.013333 | 0.013654	0.0011837 | 0.0010864	7.6031e-4 | 6.8106 e-04	0.0050506 | 0.0050477
5	0.013332 | 0.013652	0.013917 | 0.014257	0.013607 | 0.013947	0.0011208 | 9.7537 e-4	6.2571e-4 | 5.3720e-4	0.22688 | 0.22876
6	0.013344 | 0.013669	0.014104 | 0.014453	0.013333 | 0.013653	0.0010224 | 8.376 e-4	2.6926 e-4 | 2.5535 e-4	0.0067412 | 0.0066581
7	0.013338 | 0.013662	0.013401 | 0.013729	0.013338 | 0.013662	0.0016785 | 0.0016332	1.3291e-04 | 1.3569e-04	8.7751e-04 | 7.0327e-04
8	0.013336 | 0.013659	0.0036960 | 0.0045620	0.0020916 | 0.0034399	0.0010450 | 0.0010233	2.0051e-04 | 1.9918e-04	0.011520 | 0.011877
9	0.013332 | 0.013653	0.0026769 | 0.0040430	0.0020117 | 0.0027770	0.0012568 | 0.0011323	1.3506e-04 | 1.3687e-04	0.0057010 | 0.0054726
10	0.013333 | 0.013652	0.0047558 | 0.0058945	0.0013220 | 0.0019983	0.0012392 | 0.0010092	1.9358e-04 | 1.9534e-04	0.24433 | 0.24526
11	0.013335 | 0.013653	0.013335 | 0.013644	0.013333 | 0.013654	0.0012219 | 0.0010800	1.2809e-04 | 1.3074e-04	0.28937 | 0.29025
12	6.6639e-04 | 5.6821e-04	0.013338 | 0.013643	0.013331 | 0.013652	0.0015281 | 0.0014578	0.0023939 | 0.0023996	7.8699e-04 | 6.5334e-04
13	6.3979e-04 | 5.3780e-04	0.013530 | 0.013825	0.013331 | 0.013652	0.0010169 | 8.5965e-04	4.4021e-04 | 4.3643e-04	0.044395 | 0.044635
14	6.4115e-04 | 5.5837e-04	0.013333 | 0.013654	0.013333 | 0.013654	0.0011707 | 0.0011316	1.3905e-04 | 1.3907e-04	8.0279e-04 | 6.3972e-04
15	6.5179e-04 | 5.5358e-04	0.013332 | 0.013653	0.013332 | 0.013653	0.0011401 | 0.0010465	1.3107e-04 | 1.3507e-04	0.0015810 | 0.0012187

16	7.0830e-04 | 6.0794e-04	8.2838e-04 | 0.0019849	0.0020956 | 0.0028805	0.0010861 | 0.0010218	1.3082e-04 | 1.3498e-04	0.25264 | 0.25259
17	6.7184e-04 | 5.8101e-04	0.013367 | 0.013691	0.013367 | 0.013691	0.0015000 | 0.0014634	2.1603e-04 | 1.9508e-04	8.4675e-04 | 7.0382e-04
18	7.2576e-04 | 6.2741e-04	0.013928 | 0.014269	0.013506 | 0.013841	0.0011245 | 0.0010415	1.3236e-04 | 1.3540e-04	0.0011725 | 8.7162e-04
19	7.2576e-04 | 6.2741e-04	0.013716 | 0.014049	0.013332 | 0.013653	0.0011382 | 0.0010452	2.1254e-04 | 1.9239e-04	0.25044 | 0.25205
20	7.2576e-04 | 6.2741e-04	0.018952 | 0.019159	0.013352 | 0.013559	0.0012568 | 0.0011323	1.3506e-04 | 1.3687e-04	0.0057010 | 0.0054726

Tabla B

Neuronas en capa oculta	Incremental original	Incremental con momentum	Incremental con Etha adaptativo	Batch original	Batch con Momentum	Batch con Etha adaptativo
1	0.014285 | 0.014559	0.028555 | 0.028680	0.013332 | 0.013654	0.51263 | 0.51387	0.51265 | 0.51389	0.089278 | 0.089225
2	0.013431 | 0.013732	0.013332 | 0.013651	0.013332 | 0.013651	0.51404 | 0.51339	0.51404 | 0.51339	0.013331 | 0.013652
3	0.014998 | 0.015368	0.023643 | 0.024069	0.013332 | 0.013653	0.51265 | 0.51389	0.51265 | 0.51389	7.7041e-4 | 6.5691e-4
4	0.013591 | 0.013881	0.013704 | 0.013969	0.013333 | 0.013651	0.51265 | 0.51389	0.51265 | 0.51389	8.3871e-4 | 6.8221e-4
5	0.013338 | 0.013650	0.014707 | 0.014948	0.013332 | 0.013653	0.51404 | 0.51339	0.51404 | 0.51339	9.7704e-4 | 7.6952e-4
6	0.013757 | 0.014097	0.021019 | 0.021427	0.013332 | 0.013650	0.51404 | 0.51339	0.51404 | 0.51339	7.7207e-4 | 6.5197e-4
7	0.014342 | 0.014700	0.013351 | 0.013644	0.013332 | 0.013651	0.51404 | 0.51339	0.51265 | 0.51389	0.27108 | 0.27409
8	0.013333 | 0.013646	0.013351 | 0.013644	0.013331 | 0.013644	0.51265 | 0.51389	0.51265 | 0.51389	0.14934 | 0.15077
9	0.013771 | 0.014113	0.021092 | 0.021542	0.013333 | 0.013653	0.51265 | 0.51389	0.51404 | 0.51339	7.5738e-4 | 6.4796e-4
10	0.013414 | 0.013742	0.013341 | 0.013644	0.013321 | 0.013654	0.51404 | 0.51339	0.51404 | 0.51339	0.014302 | 0.014575
11	0.013592 | 0.013884	0.04263 | 0.042699	0.013335 | 0.013640	0.51404 | 0.51339	0.51265 | 0.51389	7.6849e-4 | 6.4706e-4

12	0.014071 | 0.014421	0.029310 | 0.029769	0.013332 | 0.013653	0.51265 | 0.51389	0.51265 | 0.51389	7.5033e-4 | 6.3869e-4
13	6.6802e-4 | 5.3963e-4	0.013350 | 0.013639	0.013332 | 0.013651	0.51265 | 0.51389	0.51265 | 0.51389	7.7551e-4 | 6.3318e-4
14	0.014688 | 0.014952	0.022178 | 0.022343	0.013332 | 0.013653	0.51404 | 0.51339	0.51404 | 0.51339	7.9200e-4 | 6.3026e-4
15	0.014114 | 0.014390	0.017806 | 0.018007	0.013332 | 0.013653	0.51265 | 0.51389	0.51404 | 0.51339	7.7325e-4 | 6.4347e-4
16	0.013386 | 0.013704	0.013349 | 0.013641	0.013332 | 0.013653	0.51265 | 0.51389	0.51265 | 0.51389	0.14974 | 0.14691
17	0.014176 | 0.014525	0.019110 | 0.019503	0.013333 | 0.013653	0.51265 | 0.51389	0.51265 | 0.51389	0.11785 | 0.11597
18	0.013334 | 0.013647	0.014381 | 0.014628	0.013332 | 0.013654	0.51265 | 0.51389	0.51265 | 0.51389	7.5467e-4 | 6.3269e-4
19	6.2077e-4 | 5.3968e-4	0.015343 | 0.015693	0.013333 | 0.013653	0.51265 | 0.51389	0.51265 | 0.51389	0.0076364 | 0.0067980
20	0.014074 | 0.014351	0.013352 | 0.013645	0.013332 | 0.013652	0.51404 | 0.51339	0.51404 | 0.51339	0.12684 | 0.12506

Tabla C

neuronas en capa oculta	Incremental original	Incremental con momentum	Incremental con Etha adaptativo	Batch original	Batch con Momentum	Batch con Etha adaptativo
1	0.014191 | 0.014467	0.029269 | 0.029391	0.013332 | 0.013653	0.51260 | 0.51383	0.51404 | 0.51339	0.013335 | 0.013652
2	0.013432 | 0.013733	0.018470 | 0.018665	0.013421 | 0.013754	0.51404 | 0.51339	0.51404 | 0.51339	0.013335 | 0.013652
3	0.014876 | 0.015244	0.022403 | 0.022821	0.012321 | 0.012254	0.51404 | 0.51339	0.51265 | 0.51389	0.013281 | 0.0136
4	0.013552 | 0.013845	0.014262 | 0.014512	0.013607 | 0.013947	0.34753 | 0.34865	0.51404 | 0.51339	0.21751 | 0.21848
5	6.5254e-04 | 5.4721e-04	0.022563 | 0.022871	0.015352 | 0.014643	0.24183 | 0.23413	0.51404 | 0.51339	0.017044 | 0.017493
6	0.013724 | 0.014063	0.013442 | 0.013633	0.014352 | 0.013633	0.51404 | 0.51339	0.51404 | 0.51339	0.11544 | 0.11638
7	6.1858e-04 | 4.9635e-04	0.023232 | 0.023443	0.024404 | 0.023823	0.51265 | 0.51389	0.51265 | 0.51389	0.0020660 | 0.0018121
8	0.013333 | 0.013647	0.023234 | 0.023449	0.023403 | 0.023233	0.51404 | 0.51339	0.51265 | 0.51389	0.11642 | 0.11742

9	0.013733 | 0.014074	0.025433 | 0.025826	0.024448 | 0.024893	0.51265 | 0.51389	0.51265 | 0.51389	0.0024406 | 0.0020953
10	9.1484e-04 | 8.9407e-04	0.035466 | 0.034519	0.035400 | 0.034558	0.51265 | 0.51389	0.51404 | 0.51339	0.0025814 | 0.0022398
11	1.1521e-04 | 1.3232e-04	0.037426 | 0.037509	0.034590 | 0.034548	0.51265 | 0.51389	0.51265 | 0.51389	0.15690 | 0.15733
12	3.4532e-04 | 3.2575e-04	0.037440 | 0.037518	0.035449 | 0.034677	0.51265 | 0.51389	0.51404 | 0.51339	0.0021592 | 0.0017829
13	3.1429e-04 | 2.8040e-04	0.037433 | 0.037578	0.024504 | 0.024735	0.51404 | 0.51339	0.51265 | 0.51389	0.0019638 | 0.0015820
14	2.4985e-04 | 2.3745e-04	0.025431 | 0.025820	0.014530 | 0.013668	0.51404 | 0.51339	0.51404 | 0.51339	0.29075 | 0.29052
15	7.3267e-04 | 6.6259e-04	0.025420 | 0.025821	0.014544 | 0.013690	0.51265 | 0.51389	0.51265 | 0.51389	0.0021193 | 0.0017324
16	5.9988e-04 | 5.4637e-04	0.013332 | 0.013651	0.013607 | 0.013947	0.51404 | 0.51339	0.51265 | 0.51389	0.0025892 | 0.0022042
17	8.1137e-04 | 8.2407e-04	0.013335 | 0.013670	0.013333 | 0.013653	0.51265 | 0.51389	0.51404 | 0.51339	0.20376 | 0.20465
18	5.1400e-04 | 5.1295e-04	0.013341 | 0.013644	0.013492 | 0.013301	0.51265 | 0.51389	0.51265 | 0.51389	0.0023673 | 0.0019785
19	3.0460e-04 | 3.0835e-04	0.013381 | 0.013693	0.013322 | 0.013221	0.51404 | 0.51339	0.51404 | 0.51339	0.0020146 | 0.0016164
20	5.9768e-04 | 5.4437e-04	0.013322 | 0.013678	0.013342 | 0.013251	0.51404 | 0.51339	0.51404 | 0.51339	0.0020146 | 0.0016145

Tabla D

Neuronas capa oculta	Incremental original	Incremental con momentum	Incremental con Etha adaptativo	Batch original	Batch con Momentum	Batch con Etha adaptativo
1	0.013332 | 0.013652	0.013432 | 0.013732	0.013333 | 0.013650	0.013332 | 0.013653	0.013332 | 0.013653	0.013332 | 0.013653
2	0.013333 | 0.013652	0.013472 | 0.013769	0.013333 | 0.013650	0.013328 | 0.013648	0.0053327 | 0.0054849	0.013332 | 0.013653
3	0.013340 | 0.013665	0.014583 | 0.014944	0.014343 | 0.015241	7.5224e-4 | 6.4887e-4	7.4881e-4 | 6.4468e-4	0.013328 | 0.013650
4	0.013333 | 0.013654	0.013336 | 0.013651	0.013333 | 0.013654	7.4347e-4 | 6.4042e-4	5.8298e-4 | 4.6024e-4	0.0049526 | 0.0049436
5	6.8090e-4 | 5.7729e-4	0.013334 | 0.013653	0.014262 | 0.014512	6.3326e-4 | 5.2451e-4	4.4066e-4 | 3.6988e-4	0.044072 | 0.044437

6	0.013336 | 0.013659	0.013641 | 0.013980	0.014326 | 0.015321	6.9666e-4 | 5.9365e-4	5.9330e-4 | 5.8642e-4	0.0020233 | 0.0016754
7	6.6620e-4 | 5.7444e-4	0.013332 | 0.013651	0.013607 | 0.013947	6.1401e-4 | 5.1996e-4	5.0423e-4 | 4.2991e-4	8.8013e-4 | 7.0519e-4
8	0.013334 | 0.013656	0.014262 | 0.014512	0.013335 | 0.013670	6.4232e-4 | 5.4055e-4	2.3008e-4 | 2.1541e-4	0.16794 | 0.16870
9	7.5833e-4 | 6.5932e-4	0.15690 | 0.15733	0.11642 | 0.11742	6.4437e-4 | 5.5143e-4	9.5712e-5 | 1.0453e-4	0.0052041 | 0.0050293
10	7.3150e-4 | 6.3515e-4	0.013377 | 0.013653	0.013385 | 0.013709	6.4649e-4 | 5.5419e-4	9.6337e-5 | 1.0368e-4	0.0011577 | 8.8287e-4
11	7.1007e-4 | 6.2240e-4	0.022039 | 0.035647	0.024853 | 0.037099	6.7776e-4 | 5.8058e-4	8.3226e-5 | 8.7381e-5	0.0010693 | 8.0755e-4
12	7.2007e-4 | 6.2301e-4	0.022229 | 0.035347	0.014326 | 0.015321	6.5542e-4 | 5.6351e-4	8.7794e-5 | 9.2629e-5	0.0012079 | 0.0012023
13	7.1819e-4 | 6.1378e-4	0.024359 | 0.035397	0.024853 | 0.037099	6.9995e-4 | 6.0099e-4	5.4403e-4 | 4.5776e-4	0.0010694 | 9.5984e-4
14	7.0350e-4 | 6.1545e-4	0.022021 | 0.035630	0.024343 | 0.037689	6.7306e-4 | 5.8262e-4	1.0586e-4 | 1.0717e-4	0.26217 | 0.26200
15	6.9790e-4 | 5.9522e-4	0.013655 | 0.013994	0.013332 | 0.013433	6.6473e-4 | 5.6770e-4	3.3834e-4 | 2.8790e-4	0.16514 | 0.16578
16	7.1590e-4 | 6.2031e-4	0.013719 | 0.014054	0.013641 | 0.013980	6.8415e-4 | 5.8944e-4	9.4168e-5 | 1.0161e-4	0.19351 | 0.19404
17	7.1739e-4 | 6.2615e-4	0.017044 | 0.017493	0.11642 | 0.11742	6.9912e-4 | 6.0517e-4	4.7754e-5 | 5.6950e-5	8.0752e-4 | 6.4869e-4
18	7.1403e-4 | 6.1686e-4	0.013332 | 0.013653	0.013332 | 0.013432	6.4269e-4 | 5.5315e-4	5.2652e-4 | 4.4423e-4	0.0013311 | 9.9488e-4
19	7.3284e-4 | 6.2615e-4	0.013332 | 0.013650	0.11642 | 0.11742	6.9271e-4 | 5.9305e-4	5.3038e-4 | 4.4576e-4	0.065478 | 0.065979
20	7.4284e-4 | 6.3615e-4	0.013311 | 0.010109	0.012079 | 0.012023	6.4781e-4 | 5.4250e-4	5.3375e-4 | 4.4280e-4	0.0010109 | 8.7305e-4