Trabajo Práctico 1: redes neuronales

Fernán Oviedo

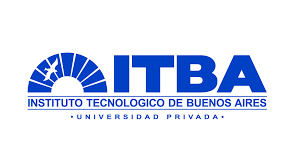
Axel Fratoni

Gastón Rodríguez

Julián Antonielli

Sistemas de Inteligencia Artificial

1er cuatrimestre 2018



[**Objetivo**](#_ixkw2cg27a0h) **2**

[**Metodología**](#_lvfelo25oy7z) **2**

[**Arquitecturas**](#_2qm8n3j0h7g5) **2**

[**Pesos iniciales**](#_43zlcrw6g5nn) **4**

[**Batch vs incremental**](#_9ls1nwtx8dw0) **4**

[**Función de activación**](#_dg9c2pkntvp8) **5**

[**Recorrido aleatorio de patrones**](#_f5iq0usjw4ae) **6**

[**Momentum**](#_a11w4xg91kt) **6**

[**Eta adaptativo**](#_swse23w4y3up) **7**

[**Comparación de arquitecturas**](#_3pdmpynf8tmv) **8**

[**Aprendizaje**](#_q6tfkgtnytm) **10**

[**Mejoras posibles**](#_kolmygqqm6d4) **10**

[Cantidad de capas y neuronas](#_dsjd9te4tyca) 10

[Optimizaciones globales de parámetros](#_auwibx1ux91q) 10

[Funciones de activación](#_evci4tiz1qtv) 10

[Mayor espectro de parámetros](#_f87g8jbok9n0) 11

# Objetivo

El objetivo de este trabajo práctico es estudiar diferentes arquitecturas de redes neuronales feed-forward para resolver el problema de aproximar un terreno. Las redes tendrán como entrada dos coordenadas cartesianas (como latitud y longitud) y devolverán la altura del terreno.

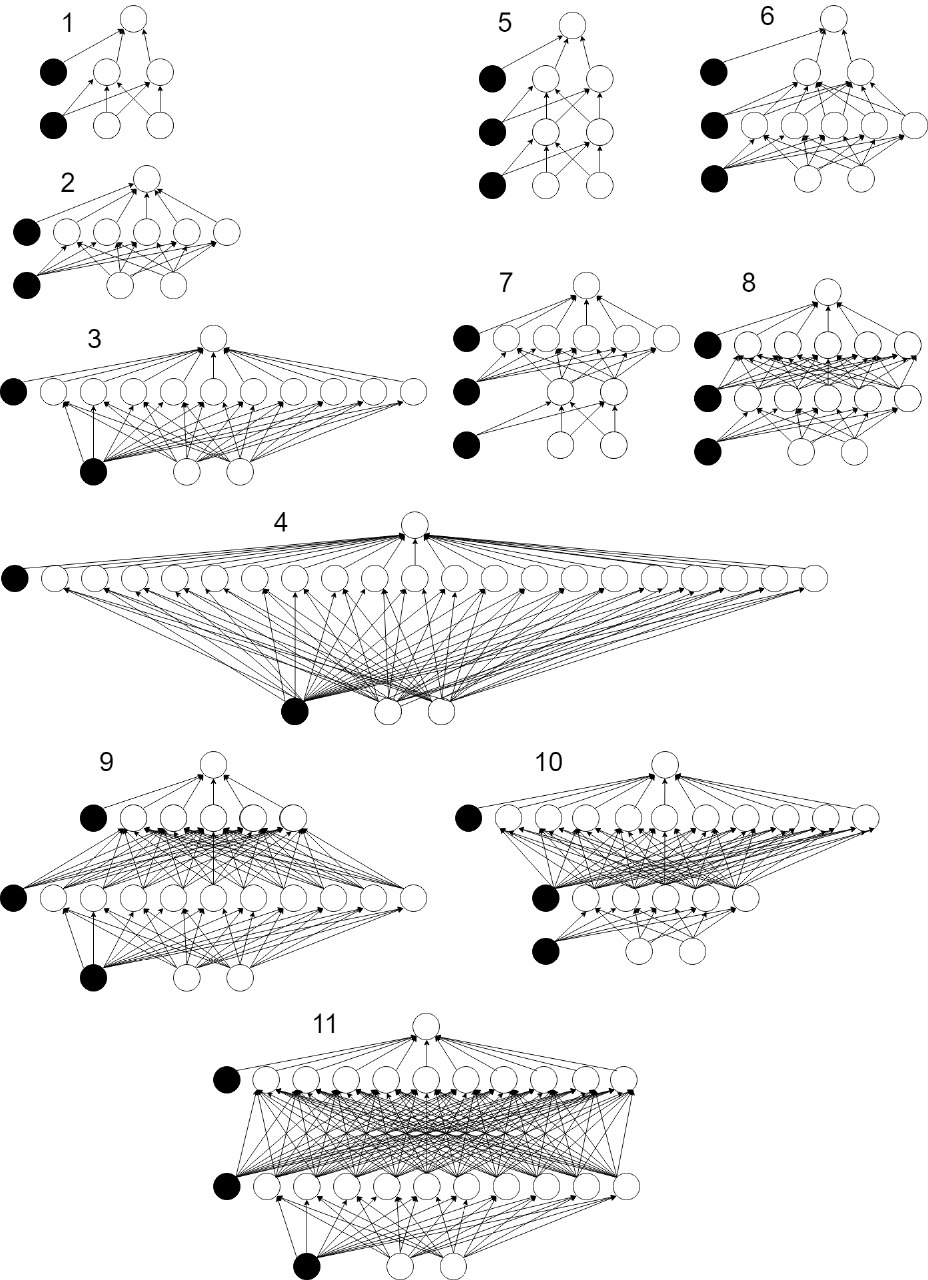
# Metodología

Para poder hacer pruebas con diversas arquitecturas, se desarrolló en Octave un motor genérico que pudiera ejecutar cualquier red neuronal feed-forward (con algunas restricciones). Una vez desarrollado, la idea era buscar la mejor arquitectura para resolver el problema. Dada la gran cantidad de parámetros, probar con todas las combinaciones posibles, incluso dentro de un marco acotado, parecía inviable, dado el tiempo de cómputo requerido. Es por esto que se tomó la decisión de proponer algunas arquitecturas e ir optimizando los parámetros de a uno para cada una de ellas, para al final elegir a la arquitectura ganadora. Otro criterio utilizado para acotar el espacio de búsqueda fue valerse de resultados teóricos que permitieran tomar decisiones *a priori*.

Para probar cuán bien funcionaba una arquitectura, se optó por estudiar cuán bien memorizaba la red el terreno completo. Esto es por el hecho de que una arquitectura que memorice muy bien el terreno es una arquitectura para la cual existe una serie de pesos que simulan el terreno fielmente. Una vez obtenida la mejor arquitectura, se estudió su entrenamiento con subconjuntos del terreno.

# Arquitecturas

Existe un resultado teórico que dice que cualquier función continua puede ser simulada con precisión arbitraria con una red feed-forward de una capa oculta, con unidades sigmoideas en las capas ocultas y unidades lineales en las demás. El mismo resultado existe para funciones cualesquiera con dos capas ocultas. El resultado no dice nada sobre la cantidad de neuronas requeridas en las capas ocultas ni sobre el proceso de aprendizaje. Sin embargo, nos valimos de éste resultado para probar en principio sólo arquitecturas de una o dos capas ocultas. Así llegamos arbitrariamente a once arquitecturas de prueba:



Las siguientes pruebas fueron todas realizadas con las once arquitecturas para determinar una ganadora en función de su capacidad de memorización.

# Pesos iniciales

Para elegir los pesos iniciales, en lugar de probar con diferentes combinaciones de pesos para cada arquitectura, nos valimos de otro resultado que recomienda que los pesos de las conexiones entrantes a una neurona sean elegidos de forma aleatoria de manera tal que sean del orden de la inversa de la raíz cuadrada de la cantidad de conexiones entrantes de la neurona. Así, todas las matrices de pesos fueron generadas aleatoriamente dentro de este orden.

# Batch vs incremental

El primer aspecto que se eligió para evaluar de las arquitecturas fue cuán bien funciona el entrenamiento batch contra el incremental. Para ello se entrenó a todas las arquitecturas durante una cantidad fija de épocas y con un factor de aprendizaje constante. De aquí se determinó que el aprendizaje incremental funcionaba mejor por dos cuestiones. El primero es que en todos los casos, el error disminuía más a lo largo de las épocas con el aprendizaje incremental. El segundo es que, aun con factor de aprendizaje pequeño, en algunos casos el error del entrenamiento batch oscilaba violentamente, sobre todo cuando la última capa oculta tenía muchas neuronas.

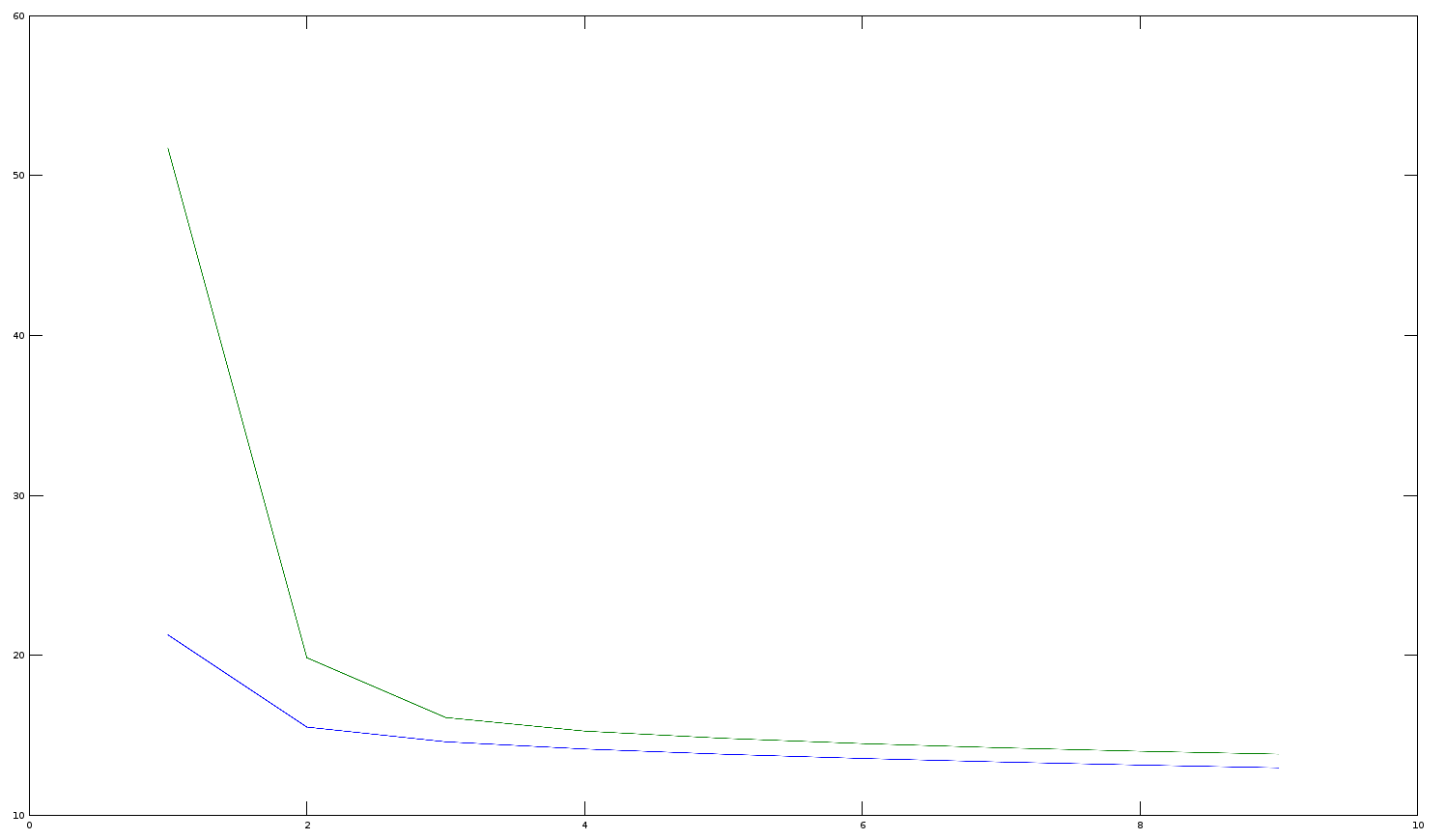


Gráfico I: error de aprendizaje en función de las épocas para la arquitectura 2 (en verde, batch; en azul, incremental)

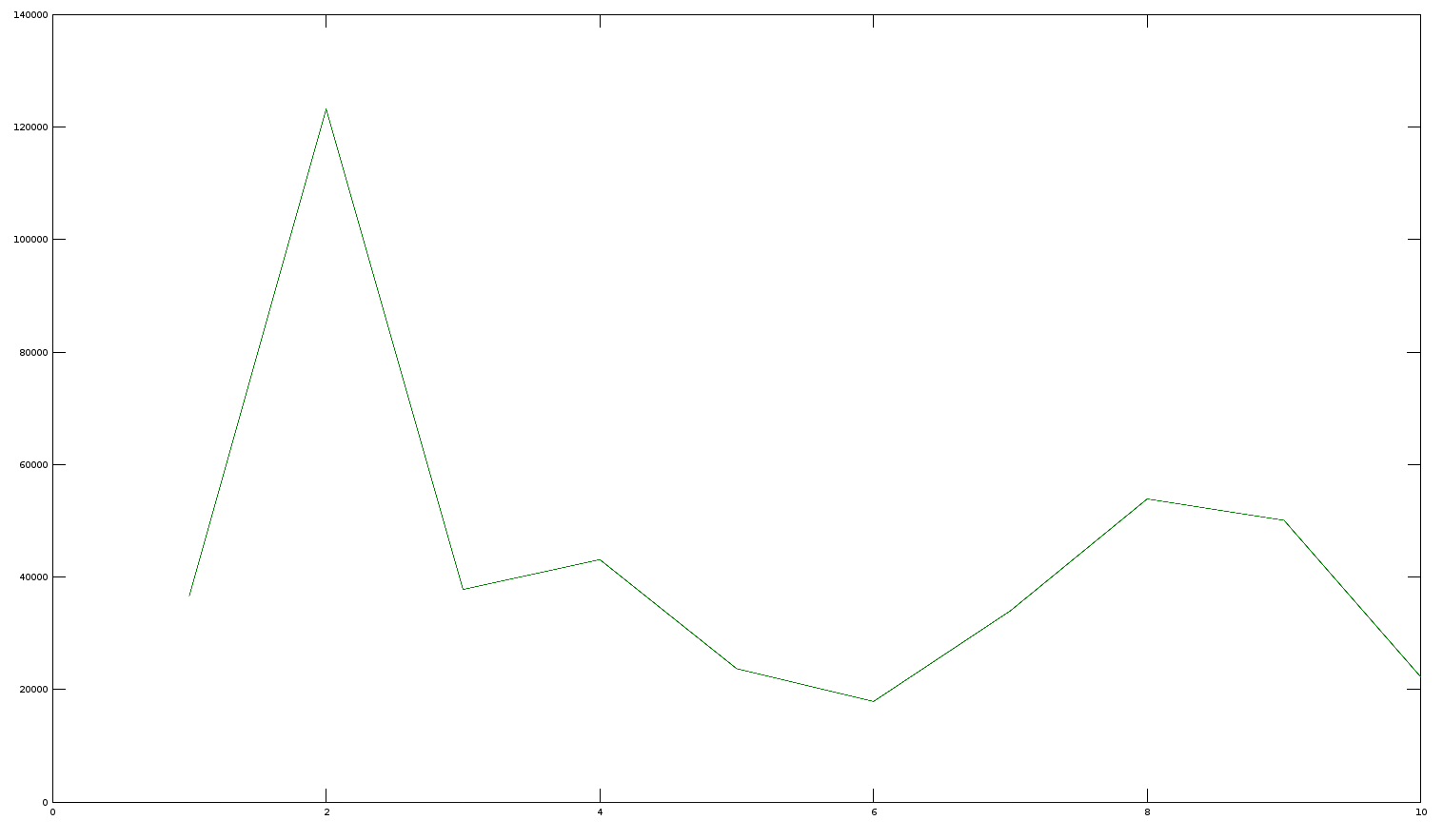


Gráfico II: error de aprendizaje en función de las épocas para la arquitectura 10 (en verde, batch; en azul, incremental)

# Función de activación

Al estar entrenando a las redes con back-propagation y gradiente descendiente, requeríamos que las funciones de activación fueran diferenciables. Al no estar la salida acotada, la última capa debía tener unidades lineales. Para las capas ocultas, decidimos probar dos funciones: la tangente hiperbólica y la función sigmoide. Nuevamente lo probamos para las once arquitecturas, y nuevamente hubo un ganador para todos los casos: la tangente hiperbólica. Ésta no sólo produjo un menor error en todos los casos, sino que la sigmoidea mostró tendencia a estancarse en mínimos locales.

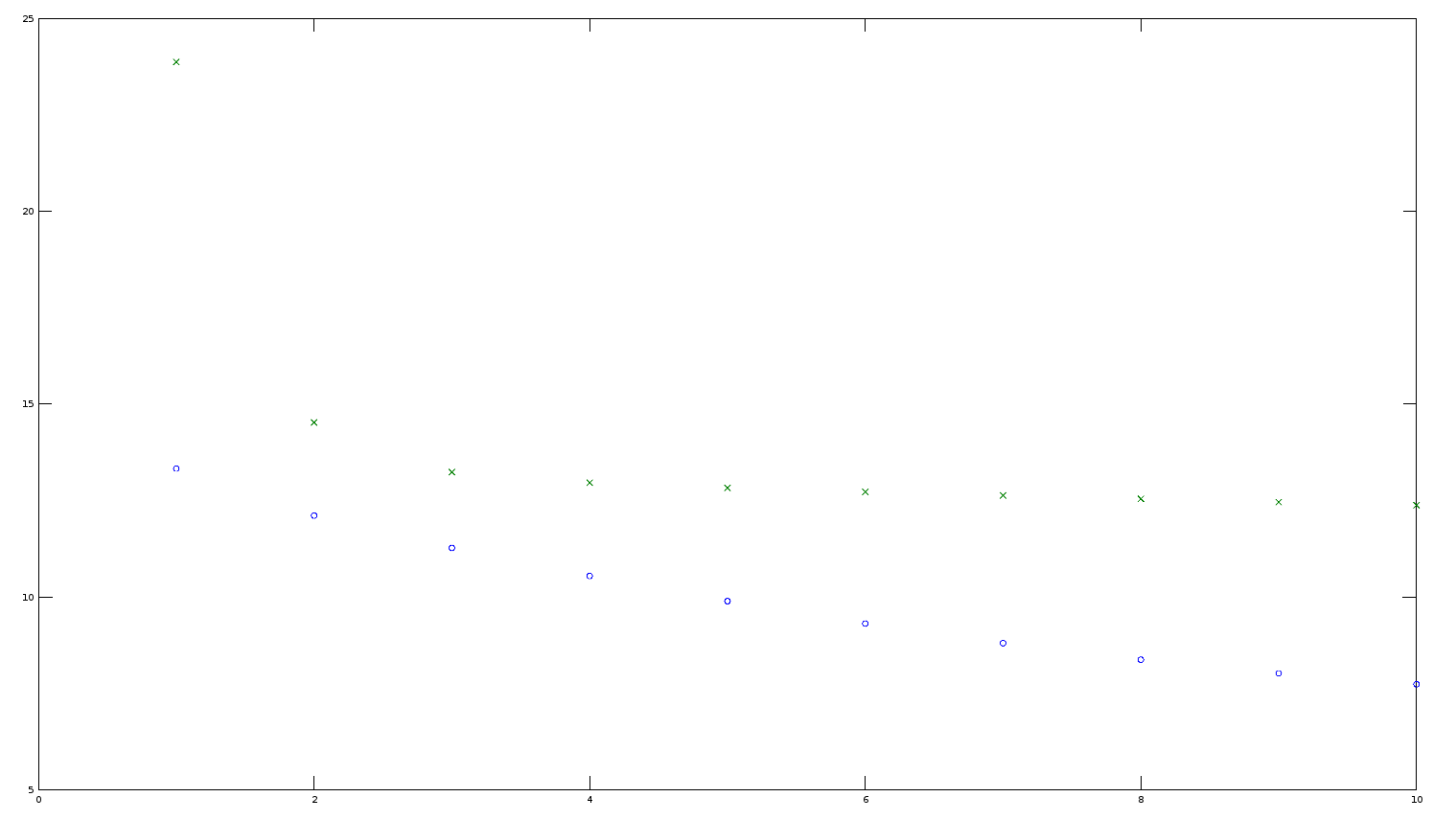


Gráfico III: error de aprendizaje en función de las épocas para la arquitectura 2 con diferentes funciones de activación (en cruces verdes, sigmoide; en círculos azules, tangente hiperbólica)

# Recorrido aleatorio de patrones

Una mejora que se puede incluir en el aprendizaje es pasar por los patrones de forma aleatoria. Esto puede sacar a la red de mínimos locales. Entonces, probamos utilizar los patrones siempre en el mismo orden contra utilizar un orden aleatorio distinto por cada época (de forma tal que los ordenamientos fueran equiprobables). Estas pruebas se corrieron más de una vez, ya que el resultado podría variar aleatoriamente. Lo que se observó es que en algunos casos, el aprendizaje era más rápido, mientras que en otros era más lento. En ningún caso se observó que la aleatoriedad sacara a la red de algún mínimo local. Ante la imposibilidad de elegir un ganador según estos resultados, optamos por utilizar el modo aleatorio, al ser el recomendado por la bibliografía.

# Momentum

Existe la posibilidad de adicionar un término de inercia al aprendizaje de la red, donde en cada actualización de pesos se le adiciona una proporción de la actualización pasada. Esta proporción que llamamos parámetro de momentum (o simplemente momentum) influye en la velocidad con la que aprende la red. En principio, probamos en todas las arquitecturas qué diferencia había entre no utilizar momentum o utilizar 0,9 (recomendado por la bibliografía). En todos los casos, el momentum mejoró la velocidad de aprendizaje, en algunos más sustantivamente que en otros.

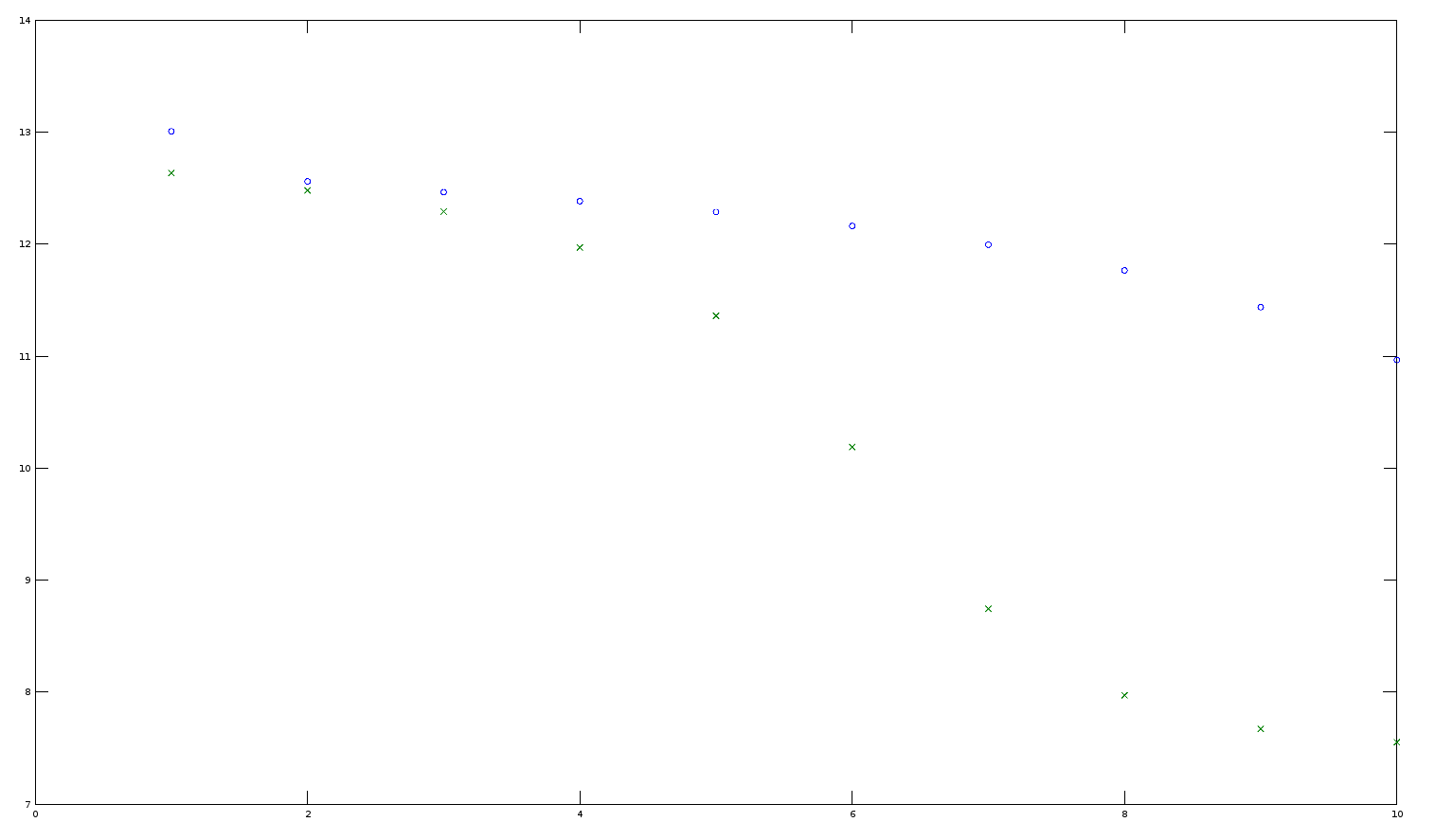


Gráfico IV: error de aprendizaje en función de las épocas para la arquitectura 1 (en cruces verdes, con momentum 0,9; en círculos azules, sin momentum)

Para no quedarnos solamente con el valor teórico de momentum, probamos utilizar también 0,5 y compararlo con el de 0,9. Nuevamente, el momentum 0,9 fue mejor para todas las arquitecturas, aunque no hubo una diferencia tan sustancial como en el caso anterior.

# Eta adaptativo

Otra característica que se puede implementar es un factor de aprendizaje variable. La idea es aumentarlo aditivamente en un factor *a* si el error cometido por la red durante *k* épocas consecutivas disminuyó y disminuirlo multiplicativamente en un factor *b* si el error de una época a otra aumentó. La idea es que se evitan las oscilaciones que producen un factor de aprendizaje grande mientras que se acelera el aprendizaje respecto a un factor de aprendizaje chico. Sin embargo, realizamos pruebas con diversas combinaciones de estos tres factores y ninguna mejoró el aprendizaje de la red. Además, la utilización de eta adaptativo ralentiza el aprendizaje de la red, ya que luego de cada época debe calcular el error para todos los patrones. Esto hizo que descartáramos la utilización de eta adaptativo.

# Comparación de arquitecturas

Una vez que determinamos los parámetros óptimos para cada arquitectura (que resultaron ser los mismos para todas), pusimos a todas a correr durante 1.000 épocas con un factor de aprendizaje de 0,05 para analizar el aprendizaje de cada una y obtener una ganadora.

Las arquitecturas con menos neuronas (1, 2, 5 y 7) mostraron no poder superar un cierto umbral de error. Además, presentaron bruscas oscilaciones en el proceso de aprendizaje. Las arquitecturas con más neuronas parecen converger hacia error cero. De todas las arquitecturas, la que mejor memorizó el terreno (más rápido y con menos error) fue la 11.

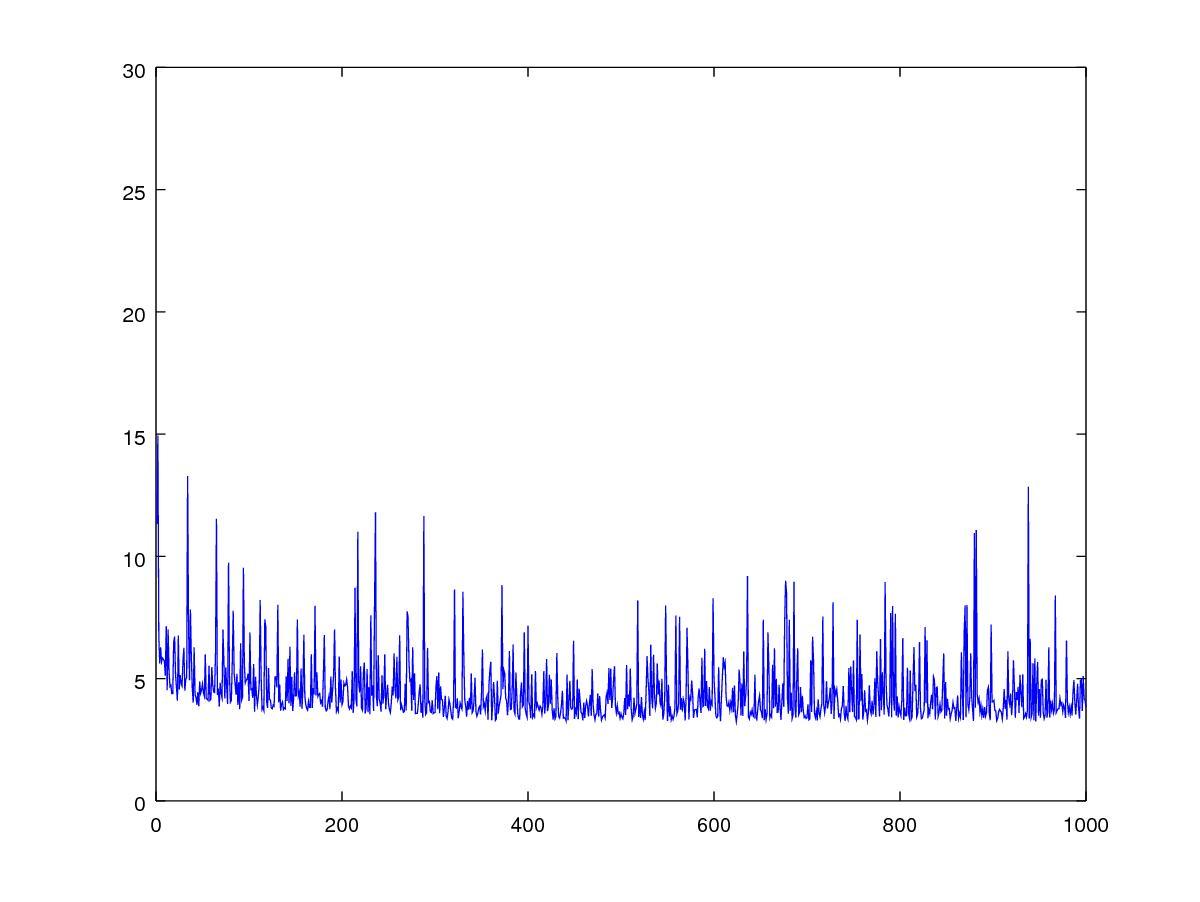


Gráfico V: error en función de épocas para la memorización del terreno en la arquitectura 2

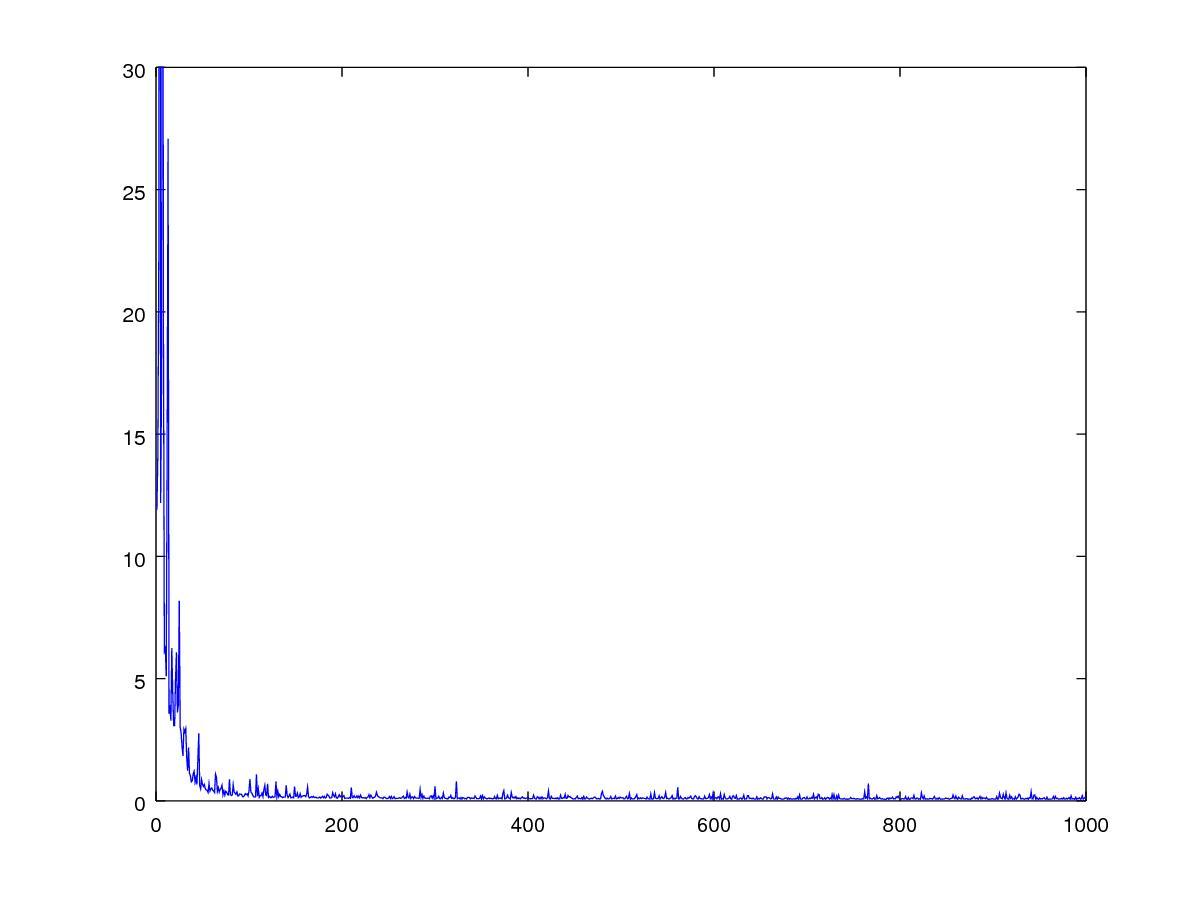


Gráfico VI: error en función de épocas para la memorización del terreno en la arquitectura 11

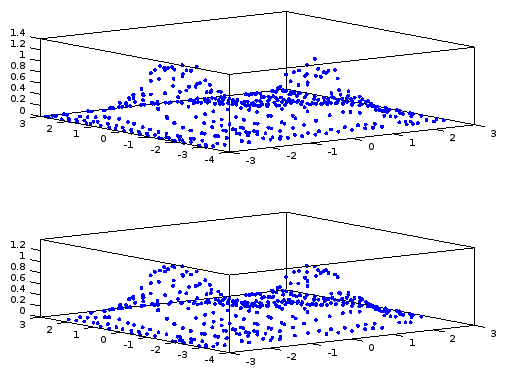


Gráfico VII: arriba, el terreno de entrenamiento; abajo, lo que aprendió la arquitectura 11

# Aprendizaje

Una vez decidida la arquitectura ganadora, comenzamos a estudiar el aprendizaje con subconjuntos del terreno, es decir, sin darle todo el terreno de entrada. La idea era ver qué generalizaba la red con diferentes cantidades de puntos tomados al azar que estuvieran distribuidos de forma “pareja” a lo largo de la superficie. Para ello se tomaron muestras de 10, 50, 100 y 200 puntos (el terreno constaba de 441) y se entrenó a la arquitectura con 100 y con 1000 épocas.

Los resultados del aprendizaje fueron los esperados. A mayor cantidad de épocas y mayor cantidad de puntos, mejor era la generalización. No se observó ningún caso de *overfitting*. La generalización más interesante se dio con 100 puntos y 1.000 épocas, ya que con un 22% de los puntos se obtuvo un error remarcablemente pequeño en la generalización (0,97 de error cuadrático medio en 441 puntos de entre 0 y 1,2 de altura). Los aprendizajes con menos puntos no fueron exitosos en la generalización, quedando el error sin bajar de cierto umbral no menor.

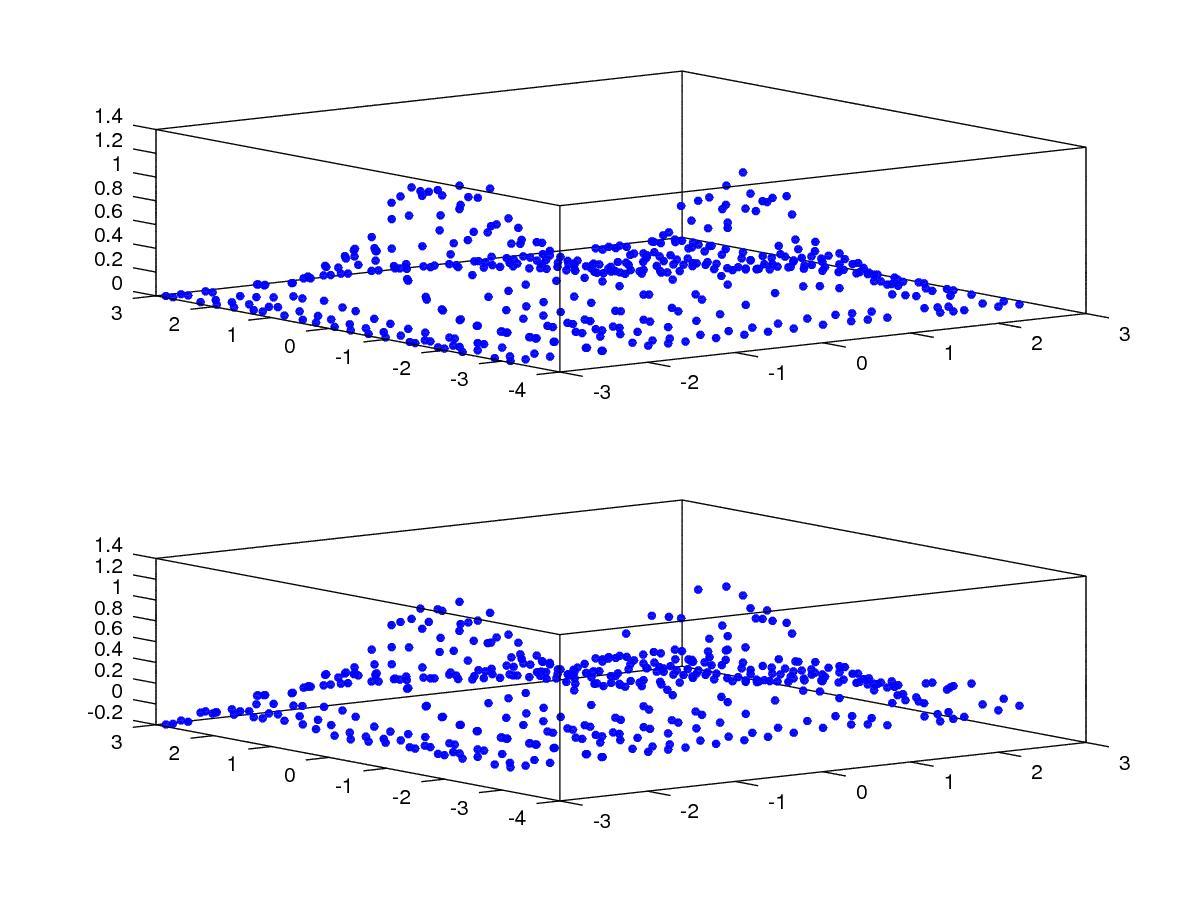


Gráfico VIII: arriba, el terreno completo; abajo, lo que generalizó la arquitectura 11 en 1.000 épocas con 100 puntos de 441

# Mejoras posibles

Hay varias cuestiones que no se tuvieron en cuenta a la hora de buscar la mejor arquitectura. Hacemos mención de las más destacables.

## Cantidad de capas y neuronas

Sólo probamos utilizar redes de hasta dos capas ocultas. Si bien el mencionado resultado teórico enuncia que debería ser suficiente para aproximar cualquier función, el aprendizaje podría ser más rápido o la generalización ser mejor con mayor cantidad de capas, o mayor cantidad de neuronas por capas ocultas.

## Optimizaciones globales de parámetros

Nosotros optamos por optimizar los parámetros de a uno. Sin embargo, esto no necesariamente da como resultado la combinación de parámetros óptima. Quizás el modo batch funcionaba mejor con momentum que el incremental. La desventaja es que buscar este óptimo global de parámetros requeriría muchísimas más pruebas.

## Funciones de activación

Las únicas funciones de activación elegidas para las capas ocultas fueron sigmoide y tangente hiperbólica. Nunca se probó utilizar una en una capa y la otra en otra capa. Tampoco se probó utilizar otras funciones de activación ni ajustar los parámetros de éstas (por ejemplo, ).

## Mayor espectro de parámetros

Para todos los parámetros se podría haber utilizado más valores de prueba. En lugar de probar tres valores de momentum, se podría haber probado 100. También se podría haber probado 500 valores de eta diferentes. Para el eta adaptativo podríamos haber utilizado muchas más combinaciones de coeficientes.