

# Trabajo Práctico Especial 1 - Métodos de Búsqueda No Informados e Informados

Sistemas de Inteligencia Artificial



Grupo 12

Pedro Balaguer - 55795

Alexander Dryselius - 60649

Agustín Izaguirre - 57774

Juan Li Puma - 55824

# Índice

<b>Índice</b>	<b>2</b>
<b>Trabajo Realizado</b>	<b>4</b>
Elección del Error Cuadrático Medio Esperado	4
Elección de Máxima Cantidad de Épocas	4
<b>Optimizaciones</b>	<b>4</b>
Momentum	4
Eta adaptativo	5
Eta Adaptativo Sin Volver Atrás los Pesos	5
Problemas Encontrados	5
<b>Resultados</b>	<b>6</b>
Determinación de la Arquitectura Óptima	6
3 Capas	7
2 Capas	7
1 Capa	7
Determinación del Factor de Aprendizaje Inicial	8
Determinación del Alfa del Momento	8
Determinación de las Constantes del Eta Adaptativo	8
Comparación de Funciones de Activación y Factor Beta	9
Determinación de Método de Inicialización Óptimo de Pesos	9
Determinación del Porcentaje de Entrenamiento	9
Configuración Óptima	10
<b>Anexo</b>	<b>13</b>
Gráfico Superficie a aprender	13
Determinación de la Arquitectura Óptima	14
3 Capas	14
2 Capas	24
1 Capa	35
Determinación del Factor de Aprendizaje Inicial	41
Determinación del Alfa del Momento	42
Alfa del momento = 0.2	44
Alfa del momento = 0.21	46
Alfa del momento = 0.22	47
Determinación de las Constantes del Eta Adaptativo	50

Eta Adaptativo con Memoria vs. Sin Memoria	65
Memoria Activada	65
Memoria Desactivada	67
Comparación de funciones de activación y factor beta	68
Determinación de Método de Inicialización Óptimo de Pesos	71
1er Corrida	72
Inicialización Random	72
Inicialización fan-in	73
2da Corrida	74
Inicialización Random	74
Inicialización fan-in	75
3er Corrida	76
Inicialización Random	76
Inicialización fan-in	77
4ta Corrida	78
Inicialización Random	78
Inicialización fan-in	79
5ta Corrida	80
Inicialización Random	80
Inicialización fan-in	81
Determinación del Porcentaje de Entrenamiento	81
Porcentaje de Entrenamiento = 88%	87
Porcentaje de Entrenamiento = 89%	89

# Trabajo Realizado

Este trabajo práctico consistió de la implementación de una red neuronal feed-forward multicapa altamente configurable.

La red puede realizar procesamiento incremental y batch. Los pesos pueden ser inicializados de manera random en el intervalo [-valor, valor] (*valor* es configurable), o utilizando fan-in. También puede elegirse la función de activación (tangente hiperbólica, exponencial o lineal) que se utilizará en todas las capas (exceptuando la de salida que siempre será lineal).

Se utilizó como función de costo el error cuadrático medio y como proceso de minimización backpropagation con el gradiente descendente. También para evitar saturación se normalizaron los valores de entrada, viéndolos como un vector y dividiéndolos por su norma.

A continuación se discuten algunos criterios tomados para el entrenamiento de la red, seguido de las optimizaciones implementadas y sus efectos.

## Elección del Error Cuadrático Medio Esperado

Para determinar el error cuadrático medio esperado, consideramos dentro de qué dominio se encontraban las respuestas. En nuestro caso siempre estaban en el intervalo [-1, 1],

A partir de eso decidimos un valor de epsilon (diferencia entre el valor esperado y el obtenido por la red) que considerábamos adecuado, basándonos en primer lugar en cuanto porcentaje representa de la magnitud total y en segundo lugar graficando la superficie obtenida por la red y observando cuanto difería de la esperada. A partir de esto decidimos que el epsilon sería  $\epsilon = 0.05$ . A partir del epsilon calculamos el error cuadrático medio como:

$$\text{error cuadrático medio} = \frac{\epsilon^2}{2}.$$

## Elección de Máxima Cantidad de Épocas

Se encontró empíricamente luego de varias corridas sin límite de épocas con distintos parámetros que pasadas las 400 el aprendizaje de la red no era significativo para el tiempo que le llevaba continuar su entrenamiento. Por esta razón, todas las redes mencionadas en el informe se entrenaron con un límite de 400 épocas.

# Optimizaciones

## Momentum

La optimización de momentum fue implementada como se vio en la clase. Para esto se guardan las modificaciones de los pesos que se iban haciendo y en cada actualización de los pesos se usaba un porcentaje de la última actualización (alfa del momentum, configurable).

$$W_{nuevo} = W_{actual} + \Delta W_{actual} + \alpha * \Delta W_{anterior}$$

## Eta adaptativo

La optimización de eta adaptativo implementada fue tal como se discutió en clase. El algoritmo toma dos decisiones:

- Si de una época a otra el error cuadrático medio aumenta, realizar lo siguiente:
  - $\eta = \eta - b\eta$ , donde  $b$  es configurable con la variable *learningFactorDecreaseFactor*.
  - Setear  $\alpha = 0$  (el alfa de la optimización de momento) hasta que haya por lo menos un decremento en el error entre dos épocas. Una vez que esto ocurra,  $\alpha$  retorna a su valor anterior.
  - Setear los pesos de la red a los que tenía en la época anterior.
- Si por  $K$  épocas consecutivas ( $K$  es configurable con la variable *adaptiveEtaDeltaSteps*) el error disminuye:
  - $\eta = \eta + a$ , donde  $a$  es configurable con la variable *learningFactorIncreaseConstant*.

Cabe destacar que si  $K > 1$  y por  $K' < K$  épocas el error disminuye, pero en la época  $K' + 1$  el error vuelve a aumentar, el algoritmo disminuye el factor de aprendizaje y resetea un contador de  $K$ . Es decir, el error tiene que volver a disminuir por  $K$  épocas y no por  $K - K' - 1$ .

## Eta Adaptativo Sin Volver Atrás los Pesos

Esto es un cambio adicional sobre la optimización de eta adaptativo (es decir, sólo tiene efecto cuando eta adaptativo está activado). La única diferencia con eta adaptativo es que cuando el error aumenta, los pesos no se setean a los del paso anterior. Esto permite salir de mínimos locales en algunas situaciones, aunque en muchas otras puede hacer que la red diverja. La idea es usar esta modificación cuando en una corrida en particular donde el eta adaptativo no deja salir de un mínimo local, y reproducir la corrida con esta variante activada para intentar salir del mínimo local.

Originalmente se planeaba implementar simulated annealing pero por falta de tiempo no se llegó a implementar. Como ya se había implementado esta variante simple como un primer paso hacia simulated annealing, se decidió dejar.

## Problemas Encontrados

El primer problema con el que nos encontramos fue que al realizar batch, en pocas iteraciones los deltas se iban a  $+\infty$  o  $-\infty$  en la mayoría de los casos, esto se debía a que en el momento de calcular la matriz del cambio de los pesos ( $\Delta W$ ) si se tomaban muchos patrones para entrenar, el producto de matrices era el producto escalar de vectores de mucha dimensión y eso hacía que los deltas a pesar de ser pequeños y en muchos casos

compensarse, al sumarse todos a veces superaban 1 y eso producía que en la próxima iteración crecieran aún más, continuando así de manera exponencial hasta llegar a +Infinito o -Infinito.

Para solucionar este problema podíamos o reducir la cantidad de patrones a usar significativamente o reducir el learning factor significativamente (ya que este era multiplicado por la matriz cambio de los pesos), otra manera posible era usar batch en conjunción con eta adaptativo, aunque a veces el eta adaptativo no alcanzaba a corregir a tiempo el learning factor.

Otro problema que tuvimos es que al usar el momento con el alfa recomendado (0,9) la red empeoraba su funcionamiento. Para solucionar este problema se buscó un alfa para el momentum que fuera mejor, como se abordará en la sección de resultados.

## Resultados

### Determinación de la Arquitectura Óptima

Para determinar la arquitectura óptima generamos ejecuciones con una, dos y tres capas ocultas, en cada caso variando la cantidad de nodos por capa. Para este análisis todas las posibles optimizaciones o alteraciones del algoritmo básico se han desactivado (momentum, eta adaptativo, etc.) y se utiliza la función exponencial debido a que suele reducir el error más velozmente que *tanh*. Decidimos limitarnos a tres capas ocultas, por lo general no se suele utilizar más para problemas de estas características, y así explorar redes más anchas que profundas. Para obtener los resultados de cada configuración establecimos una misma semilla para la generación aleatoria de números y así poder comparar situaciones iguales.

De todas las redes probadas elegímos según tres criterios:

- Cuánto es el error cuadrático medio (cuanto más chico mejor)
- Cuán rápido, en épocas, bajaba el error (más rápido es mejor)
- Cantidad de unidades (menor cantidad de unidades mejor).

Para cada tipo de red se varió la cantidad de unidades en cada capa de la manera que se lista a continuación:

- 3 capas: Cada capa varía su cantidad unidades por capa(sin bias) entre 2 y 26 con un paso de 8.
- 2 capas: Cada capa varía su cantidad unidades por capa(sin bias) entre 2 y 22 con un paso de 4.
- 1 capa: Cada capa varía su cantidad unidades por capa(sin bias) entre 2 y 30 con un paso de 2.

## 3 Capas

Las mejores 5 configuraciones para tres capas, listadas de mejor a peor de acuerdo a los criterios ya especificados son:

- #1: 18 10 18
- #2: 26 26 10
- #3: 10 10 18
- #4: 18 18 18
- #5: 18 18 10

Donde los números separados por espacio son la cantidad de unidades por capa numeradas desde la capa oculta más cercana a la capa de entrada hasta la más lejana, es decir la mejor configuración tiene 18 unidades en la primer capa oculta, 10 unidades en la segunda capa oculta y 18 unidades en la tercer capa oculta.

La discusión de porqué están en ese orden será detallada en el anexo.

## 2 Capas

Las mejores 5 configuraciones para dos capas, listadas de mejor a peor de acuerdo a los criterios ya especificados son:

- #1 22 6
- #2 14 10
- #3 14 6
- #4 18 6
- #5 10 22

Donde los números separados por espacio son la cantidad de unidades por capa numeradas desde la capa oculta más cercana a la capa de entrada hasta la más lejana, es decir la mejor configuración tiene 22 unidades en la primer capa oculta y 6 unidades en la segunda capa oculta.

La discusión de porqué están en ese orden será detallada en el anexo.

## 1 Capa

Las mejores 5 configuraciones para una capas, listadas de mejor a peor de acuerdo a los criterios ya especificados son:

- #1: Con 4 unidades en la capa oculta.
- #2: Con 28 unidades en la capa oculta.
- #3: Con 10 unidades en la capa oculta.
- #4: Con 16 unidades en la capa oculta.

#5: Con 18 unidades en la capa oculta.

La discusión de porqué están en ese orden será detallada en el anexo.

A partir de los resultados obtenidos de las 5 mejores configuraciones para cada tipo de arquitectura podemos concluir que:

**Error 3 capas < Error 2 capas < Error 1 capa**

## Determinación del Factor de Aprendizaje Inicial

Con el objetivo de sólo concentrarse en el factor de aprendizaje inicial, se realizaron pruebas sin utilizar momento ni eta adaptativo, con el proceso incremental, utilizando como función de activación a la función exponencial con un beta de 1.

Para determinar el factor de aprendizaje se realizaron 10 corridas donde se iba variando su valor de 0.1 a 0.9 con un paso de 0.1. De esto se obtuvo que la mejor fue la de 0.2 (justificación en anexo) y como 0.3 era mejor que 0.2 se realizó otra corrida variando el valor del learning factor entre 0.2 y 0.3 con un paso de 0.01

De esta última corrida se obtuvo el mejor learning factor.

**Mejor learning factor: 0.25**

La justificación de este valor se encuentra en el anexo.

## Determinación del Alfa del Momento

Con el objetivo de concentrarse solo en el alfa del momento, se realizaron las pruebas dejando las demás variables constantes y solo se varió el alfa (además con el eta adaptativo desactivado).

Para determinar el alfa del momento se realizaron 2 pruebas, primero una con alfa entre 0.1 y 0.9 incrementando en cada iteración su valor en 0.1.

Luego de realizar la primer prueba obtuvimos que el mejor alfa era de 0.2 y que 0.3 era mejor que 0.1.(Estos resultados son analizados en el anexo)

La segunda prueba era variar el alfa entre 0.2 y 0.3 con un paso de 0.01 y se obtuvo:

**Mejor Alfa de Momento: 0.2**

## Determinación de las Constantes del Eta Adaptativo

Para explorar el efecto de la regulación del factor de aprendizaje por parte de eta adaptativo, se fijó el K (es decir, la cantidad de pasos consecutivos en los cuales el error debe disminuir para que se considere un decrecimiento “consistente”) en 5, y se variaron las constantes a y b, utilizadas para el incremento y decremento respectivamente del factor de aprendizaje. Se tomaron como valores de a 0.05, 0.0875 y 0.125, mientras que para b se

tomaron los valores 0.1, 0.25 y 0.5. Como se muestra en el anexo, la combinación óptima de estos valores resultó ser **a = 0.125** y **b = 0.1**.

## Comparación de Funciones de Activación y Factor Beta

Se analizaron dos funciones de activación, la exponencial y la tangente hiperbólica. Además, se varió la constante beta, un factor dentro de cada una de las funciones a aplicar, con valores de 0.25, 0.5, 1 y 2.

Podemos observar en las tablas de resultados (ver anexo) que el método que reduce más el error es utilizando la función exponencial con beta igual a 1. Sin embargo, observando los gráficos de error, en ambos casos, exponencial y tangente hiperbólica, al aumentar el valor de beta la caída inicial del error es más pronunciada y llega a un error no significativamente mayor. Si esta diferencia de error no fuese fundamental y sí lo fuera en mayor medida la cantidad de épocas para el entrenamiento se podría optar por la función exponencial con un beta igual a 2, se obtendría un error similar en menor cantidad de épocas.

## Determinación de Método de Inicialización Óptimo de Pesos

Para esta prueba se utilizó la red con el momento y el eta adaptativo desactivado y manteniendo las demás variables constantes, a excepción de la manera con la que se inicializan los pesos.

La prueba consistió en 5 corridas para inicialización random y 5 corridas para inicialización de pesos con fan-in.

Se entrenó la red con ambas manera de inicialización los pesos y en cada ejecución se usó la misma semilla en ambos métodos(para que sea determinístico).

Se obtuvo que la mejor manera de inicializar los pesos es la inicialización con pesos random.

### **Mejor inicialización de pesos: Inicialización Random [-0.5, 0.5]**

La razón de esta elección es discutida en el anexo.

## Determinación del Porcentaje de Entrenamiento

Para esta prueba se utilizó la red con el momento y el eta adaptativo desactivado y manteniendo las demás variables constantes, a excepción del porcentaje de entrenamiento.

La prueba consistió en ir variando el porcentaje de entrenamiento entre el 10% y el 90% con una modificación del 10% en cada corrida.

Luego de realizar esta prueba, concluimos que los dos mejores porcentajes de entrenamiento son 80% y 90%(ver anexo).

Realizamos una segunda prueba donde variamos el porcentaje de entrenamiento entre el 80% y el 90% con una modificación del 1% en cada corrida.

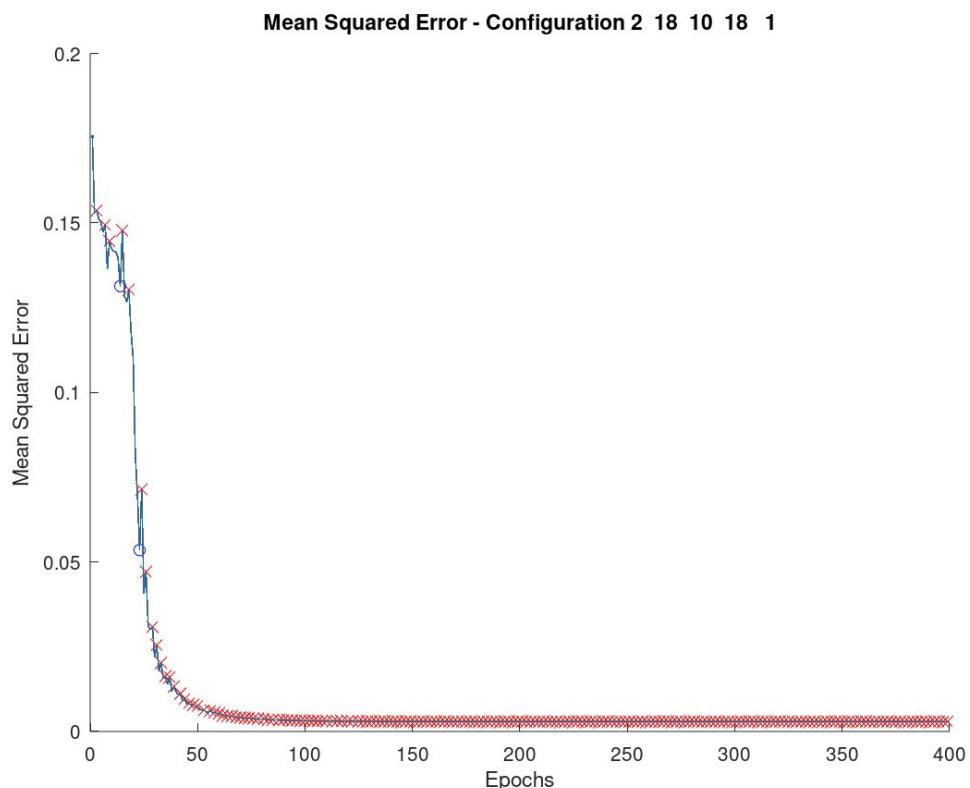
En esta última corrida se obtuvo que el mejor porcentaje de entrenamiento es:

**Mejor Porcentaje de Entrenamiento: 88%**

## Configuración Óptima

Tomando los resultados obtenidos podemos crear la configuración óptima de la red:

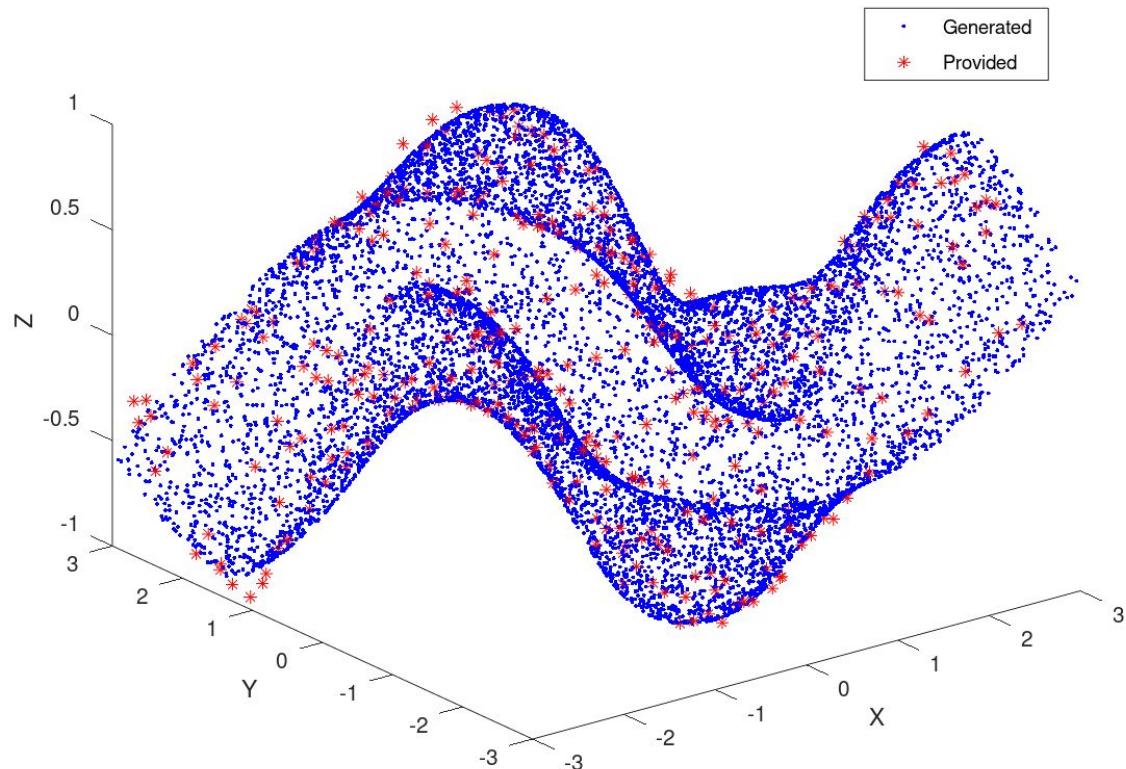
- Tres capas ocultas con 18, 10 y 18 unidades en cada una
- Eta inicial = 0.25
- Inicialización de pesos: aleatorio o fan-in
- Función de activación exponencial con beta = 1
- Alpha de momentum = 0.2
- Factor de decrecimiento del eta adaptativo = 0.125
- Factor de crecimiento del eta adaptativo = 0.1



El error baja muy rápidamente antes de estancarse. Si bien el error no es el mínimo encontrado a lo largo de todo el análisis, la ventaja de esta configuración y sus optimizaciones es que en aproximadamente 50 épocas obtenemos un error muy pequeño, comparable a lo que se obtenía en casi de 100 épocas con la misma configuración de red (18-10-18) y sin optimizaciones.

Error Cuadrático Medio = 0.002913

Both terrain positions - Configuration 2 18 10 18 1



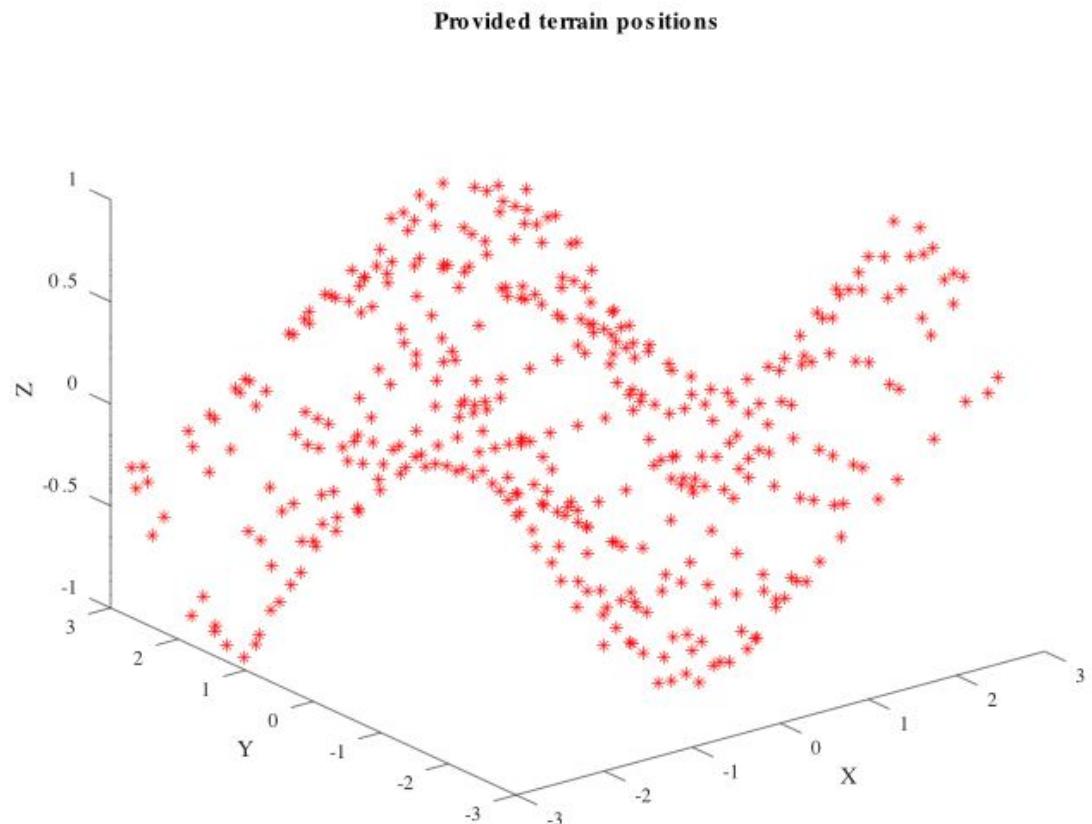
## Conclusiones

Existen diferentes optimizaciones que permiten obtener un error menor o reducirlo más rápidamente (ej: momentum, eta adaptativo).

Es posible encontrar rápidamente una configuración de red que aprenda el problema, sin embargo llegado un punto es muy complicado seguir reduciendo el error, por lo que se vuelve más importante reducir la cantidad de épocas para obtener un error similar.

## Anexo

### Gráfico Superficie a aprender

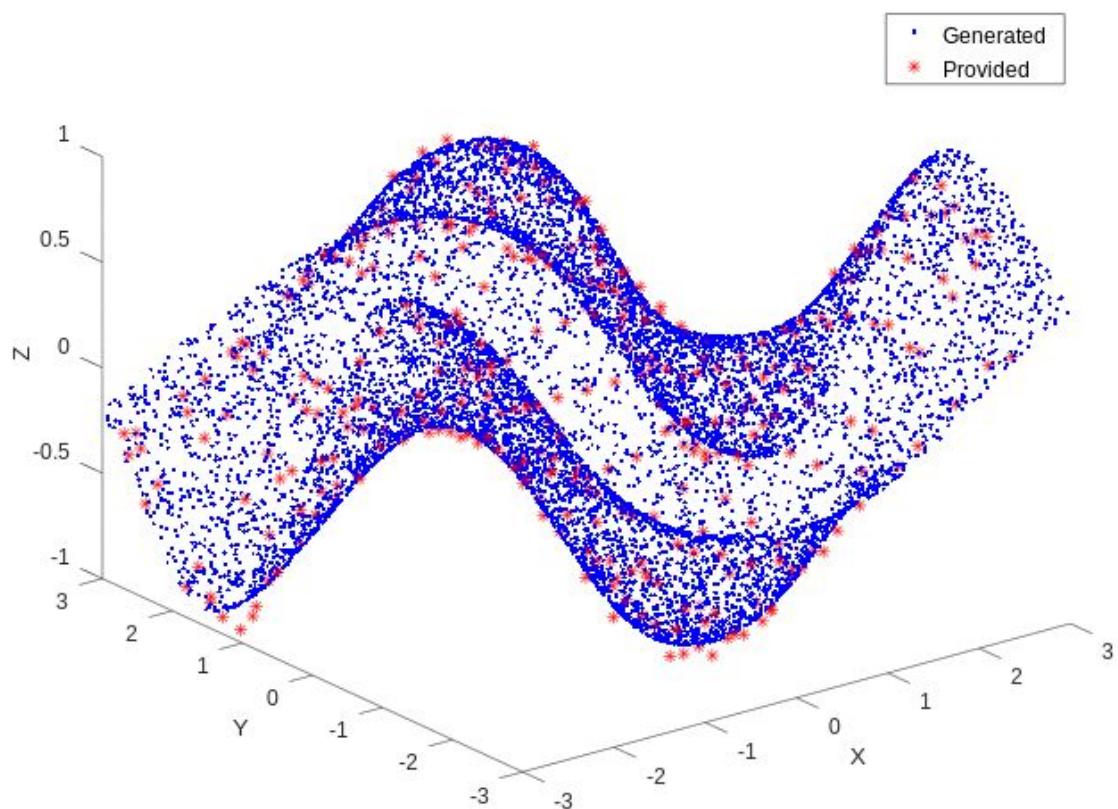


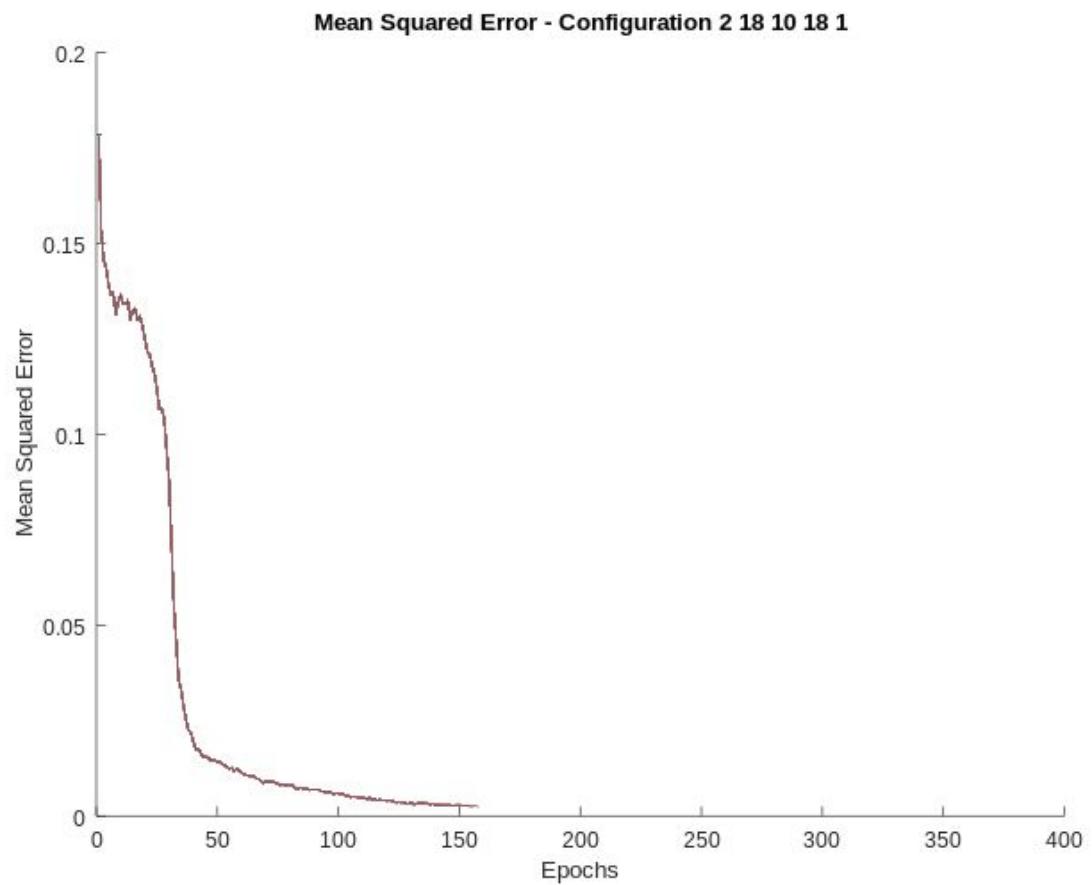
## Determinación de la Arquitectura Óptima

3 Capas

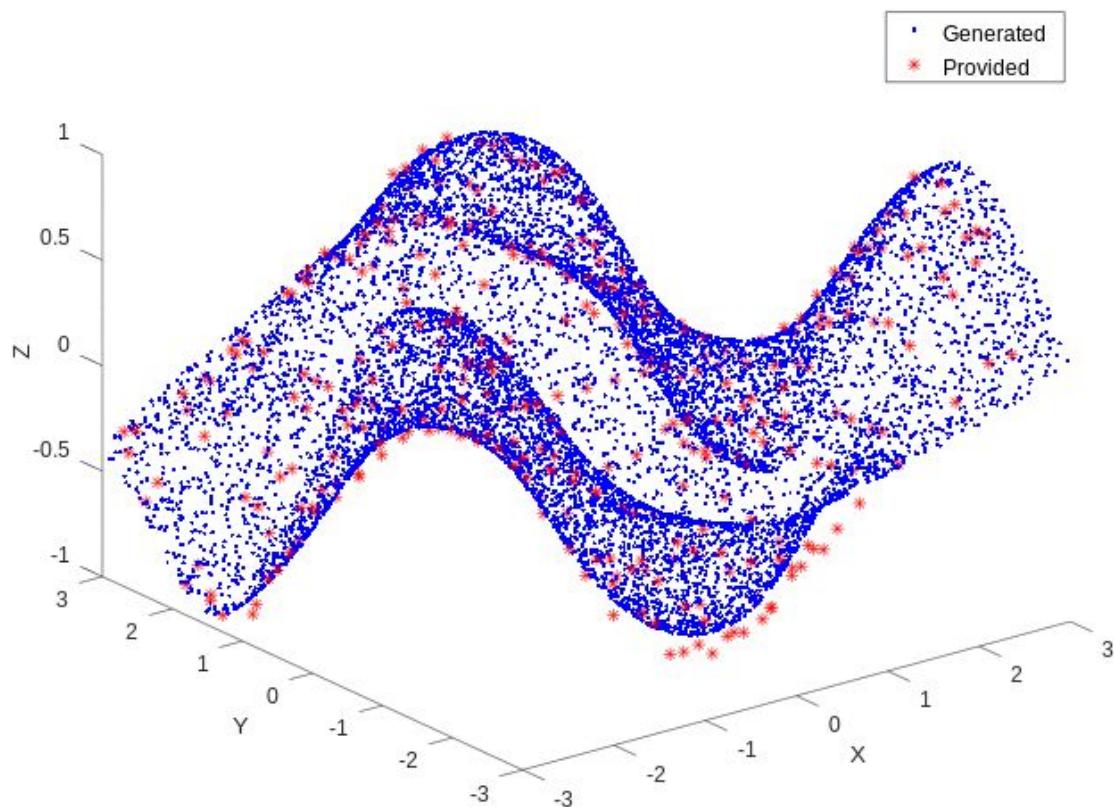
Épocas	Error	Cantidad de neuronas en capa oculta #1	Cantidad de neuronas en capa oculta #2	Cantidad de neuronas en capa oculta #3
159	0.0022577	18	10	18
166	0.0024389	26	26	10
189	0.0022340	10	10	18
179	0.0023109	18	18	18
199	0.0024438	18	18	10

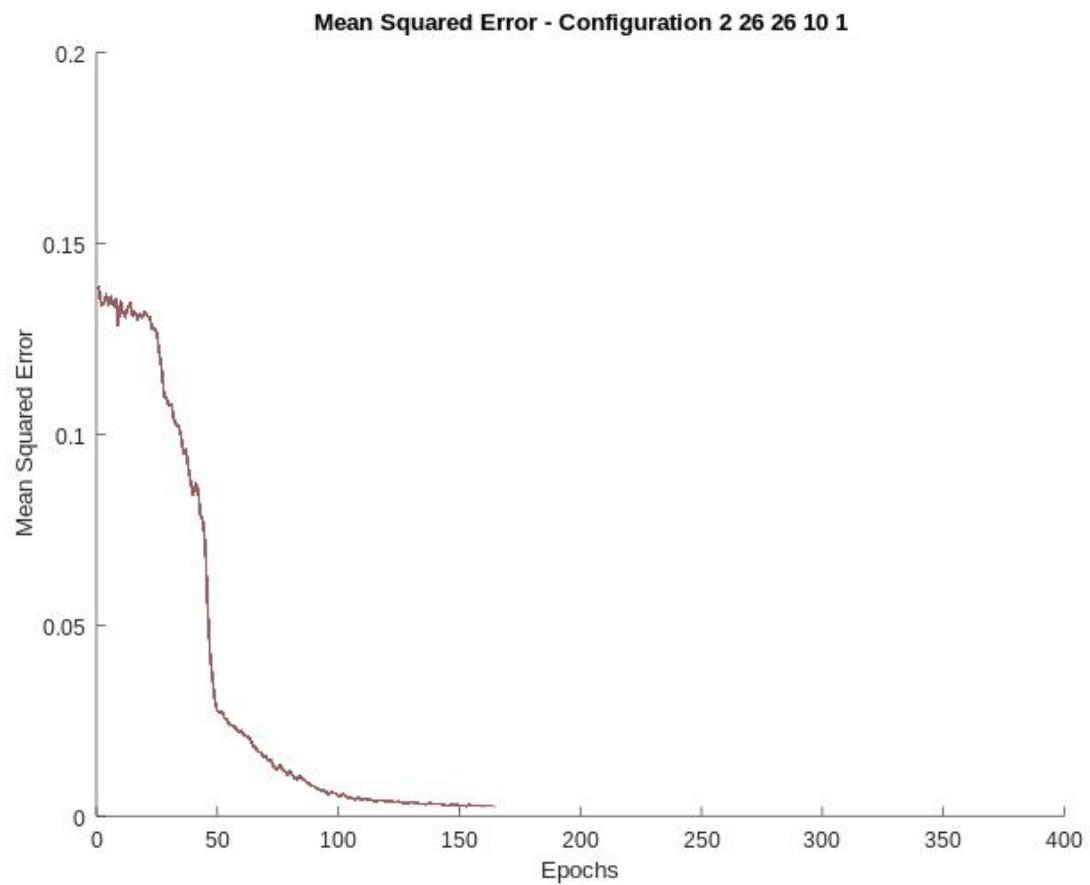
Both terrain positions - Configuration 2 18 10 18 1



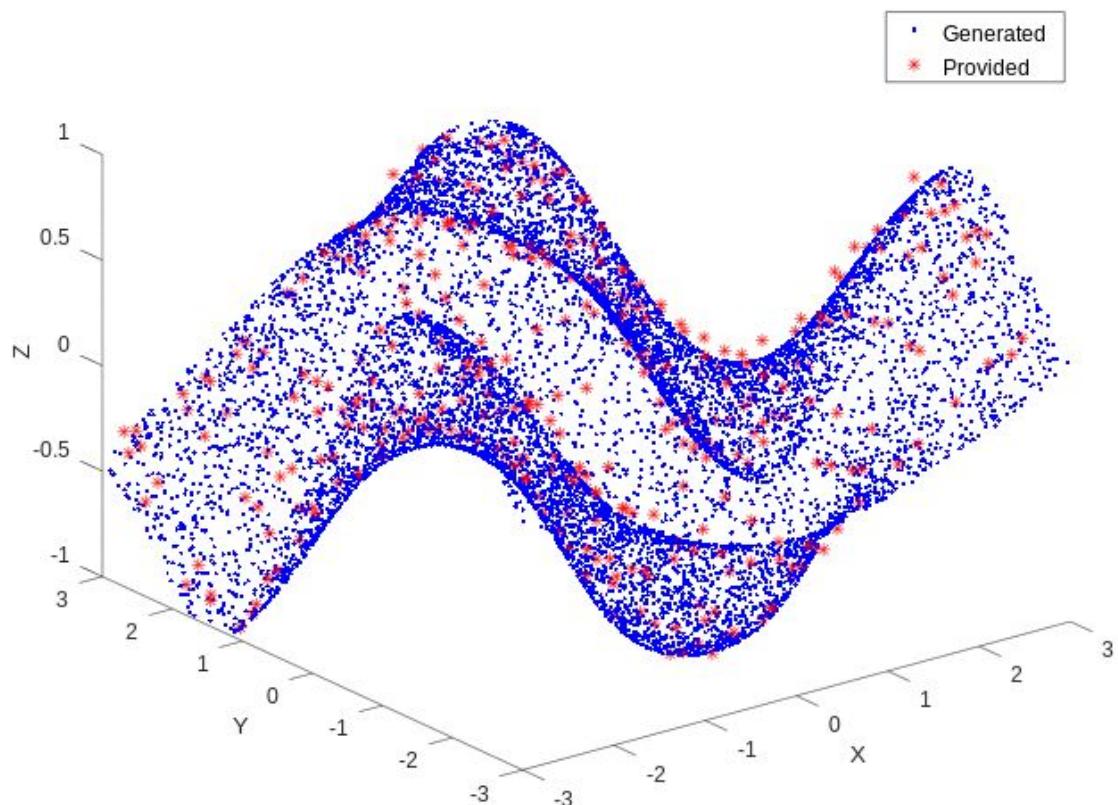


**Both terrain positions - Configuration 2 26 26 10 1**

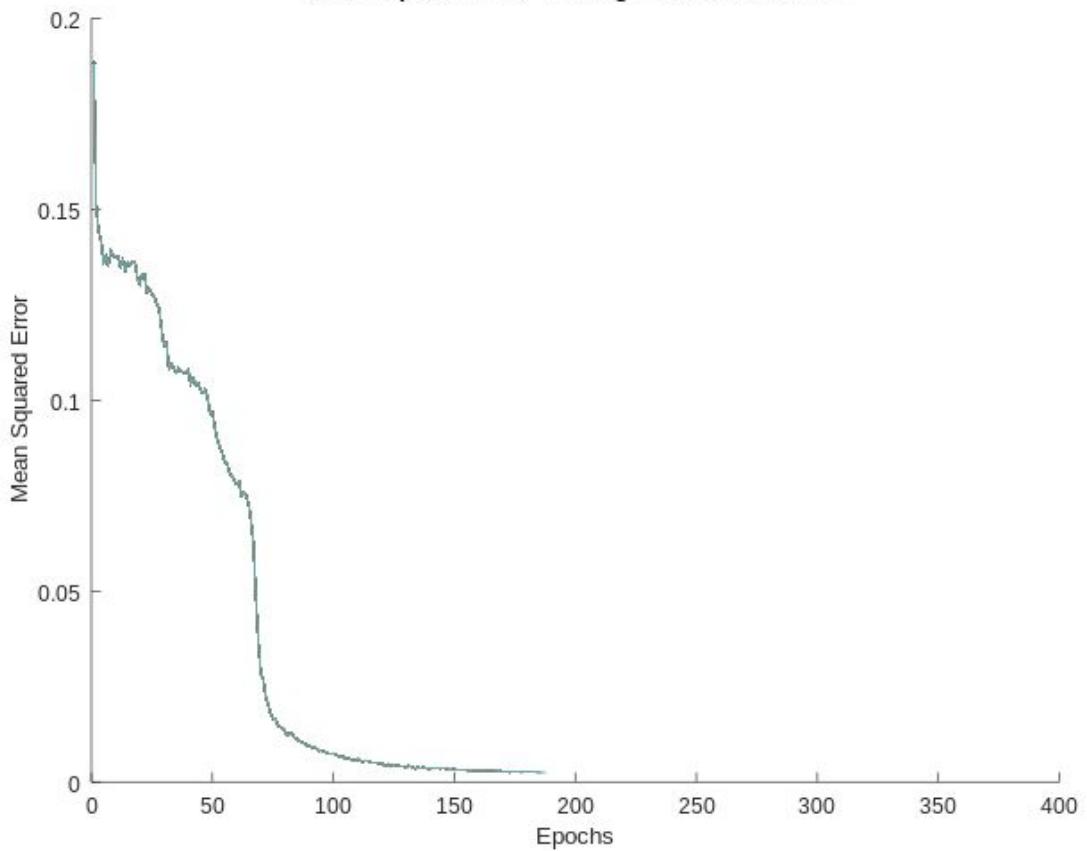




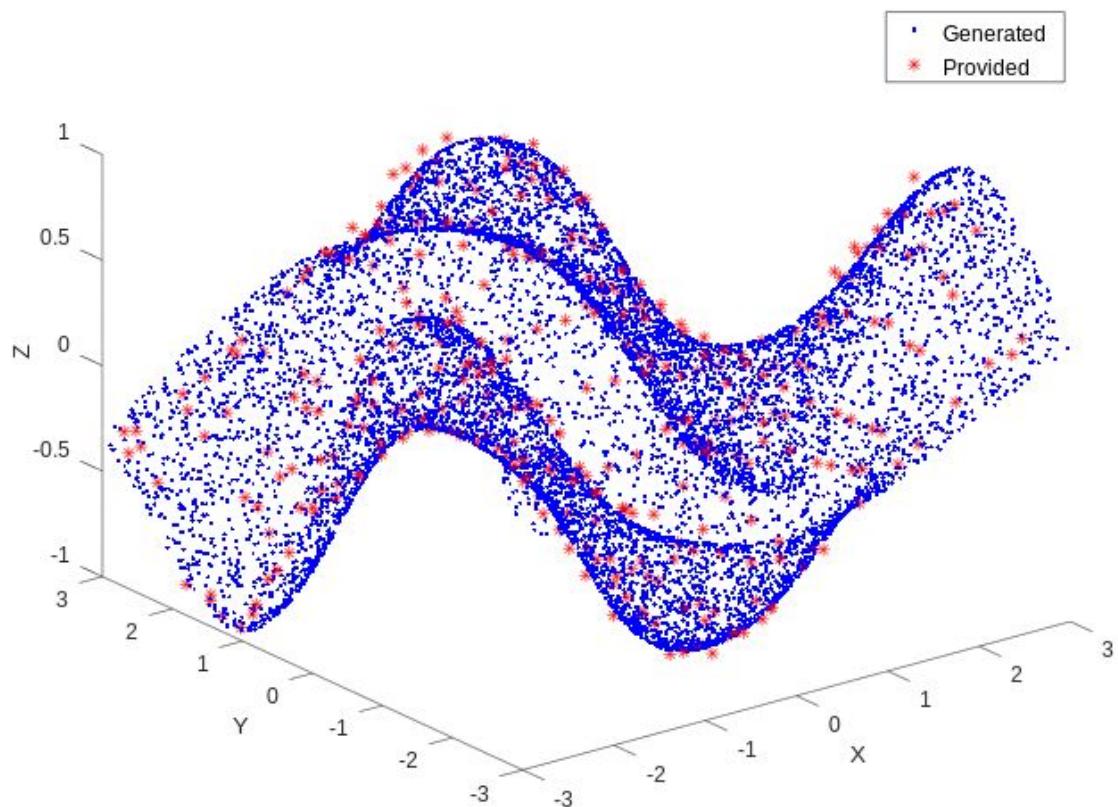
Both terrain positions - Configuration 2 10 10 18 1

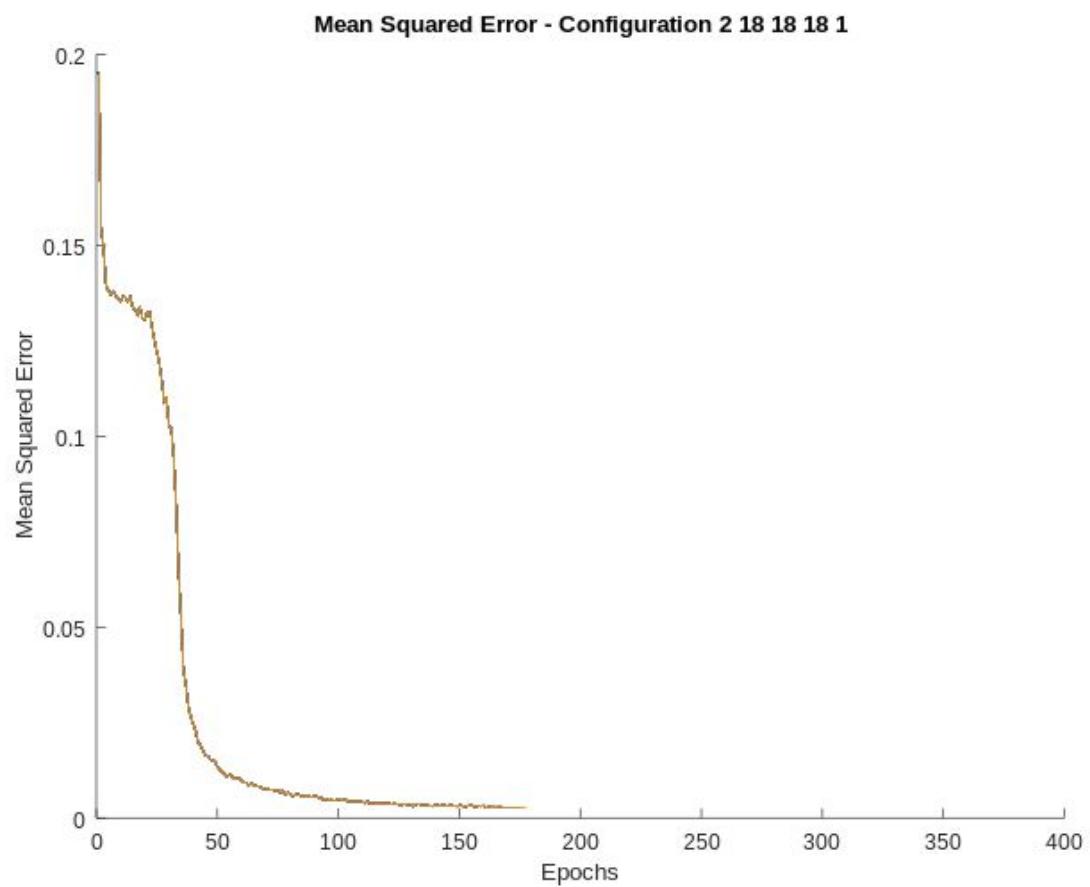


**Mean Squared Error - Configuration 2 10 10 18 1**

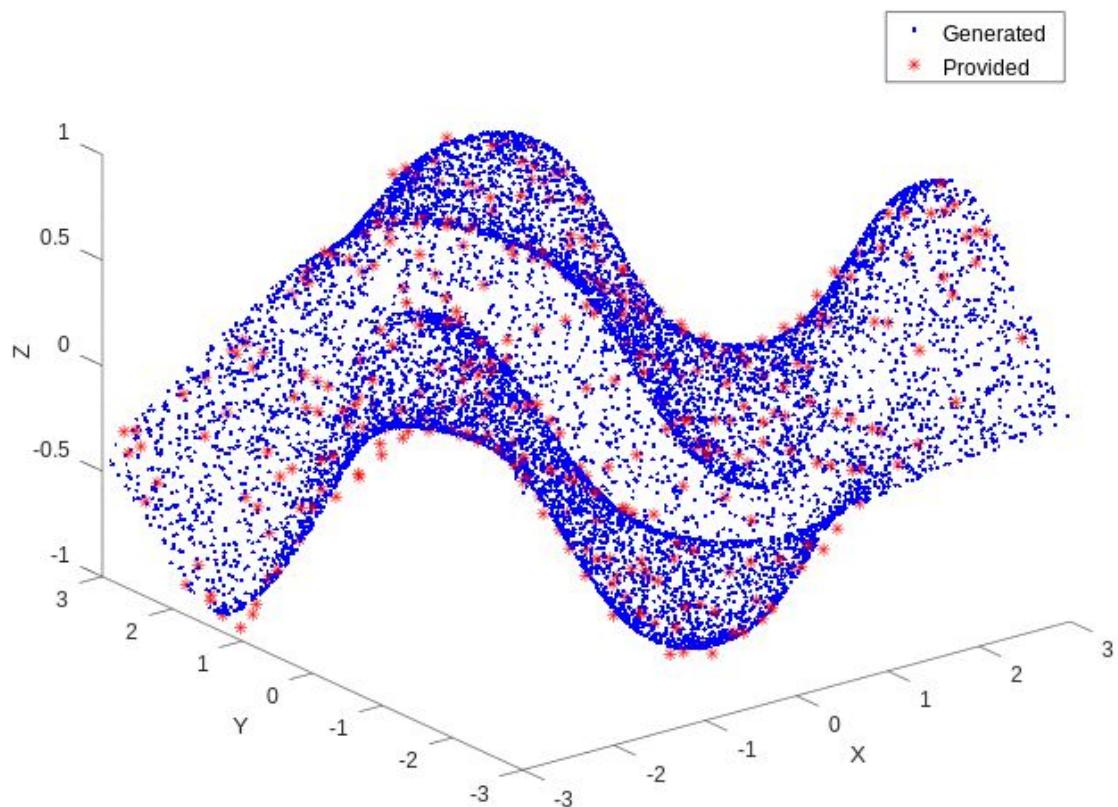


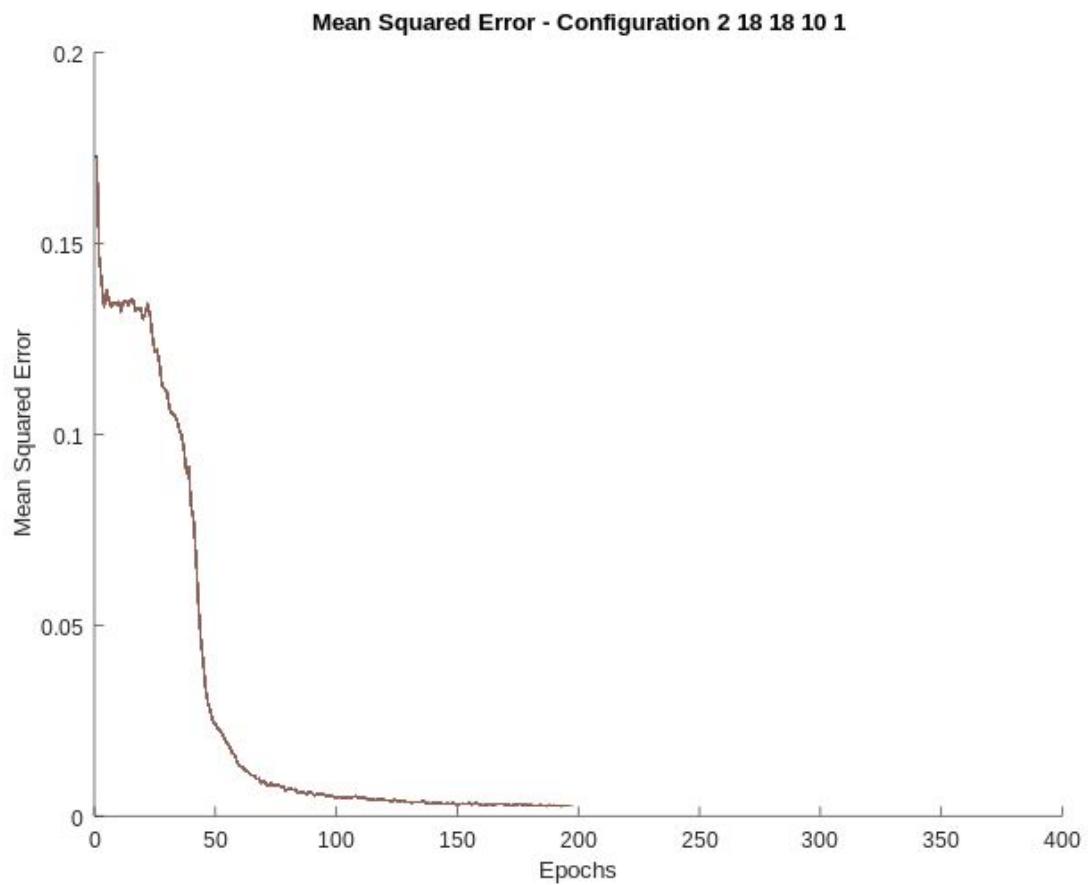
Both terrain positions - Configuration 2 18 18 18 1





Both terrain positions - Configuration 2 18 18 10 1

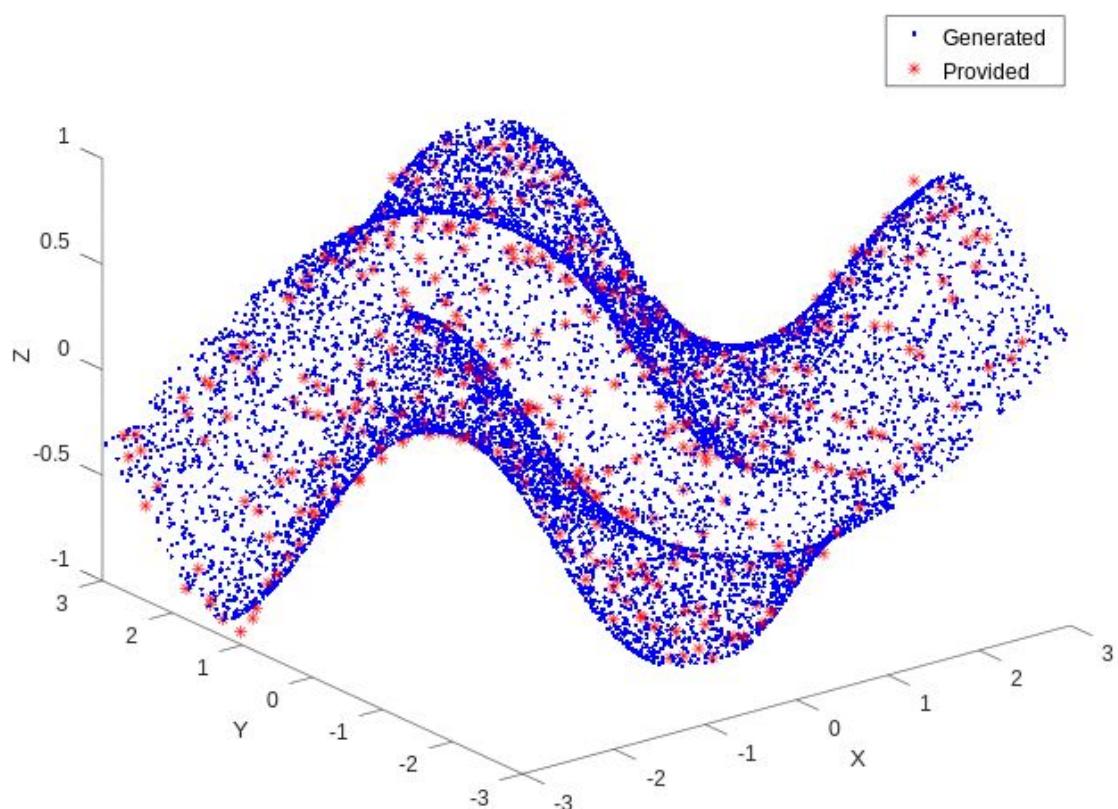


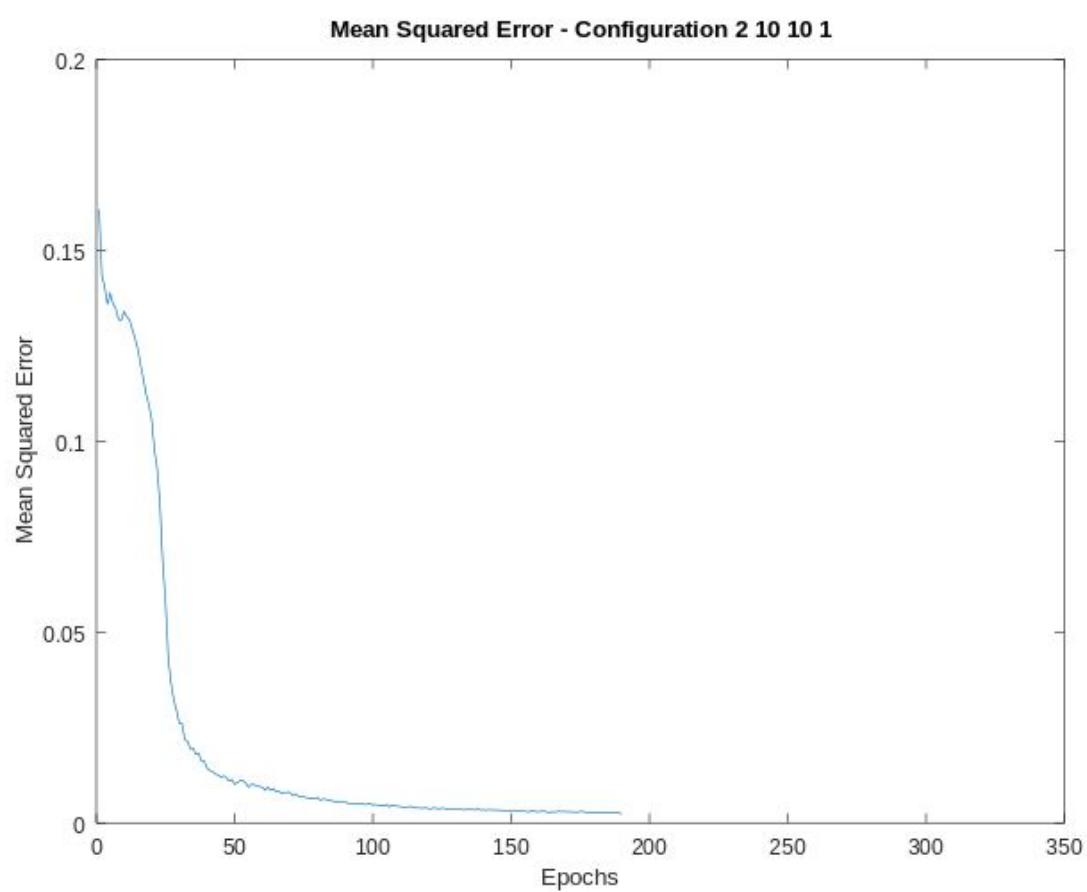


## 2 Capas

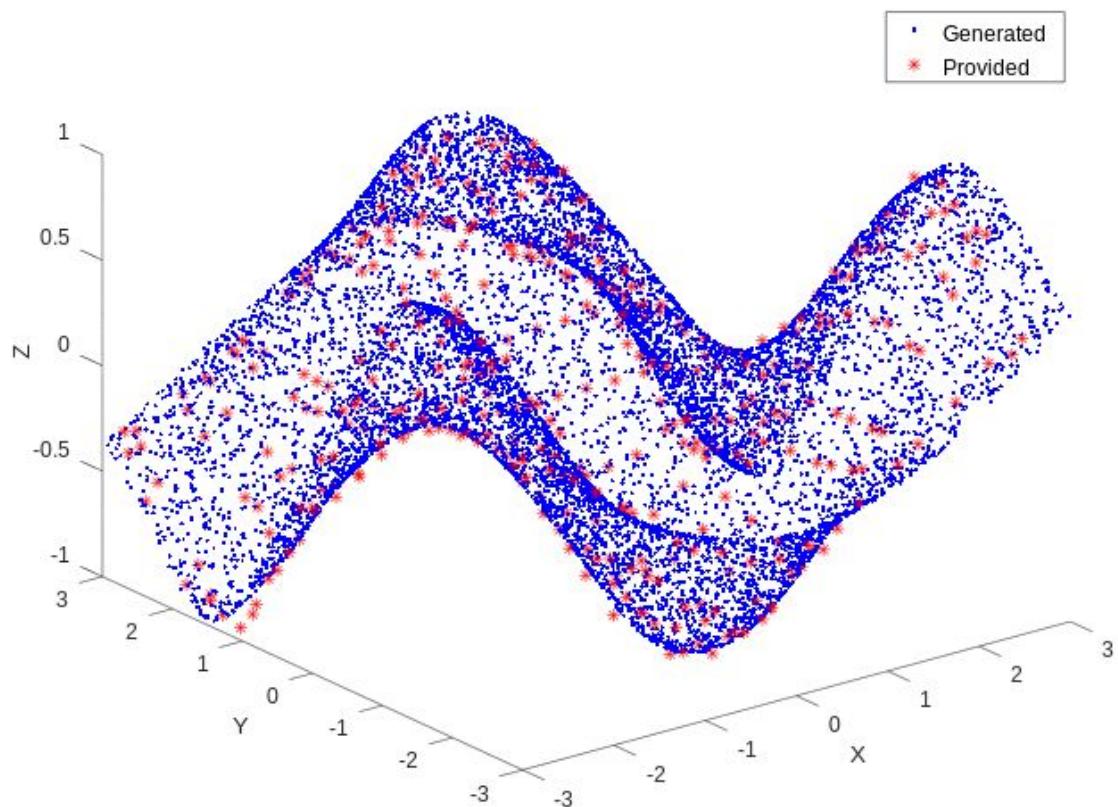
Épocas	Error	Cantidad de neuronas en capa oculta #1	Cantidad de neuronas en capa oculta #2
190	0.002362	10	10
207	0.002447	22	22
240	0.002424	10	18
256	0.002420	10	22
310	0.002431	10	14

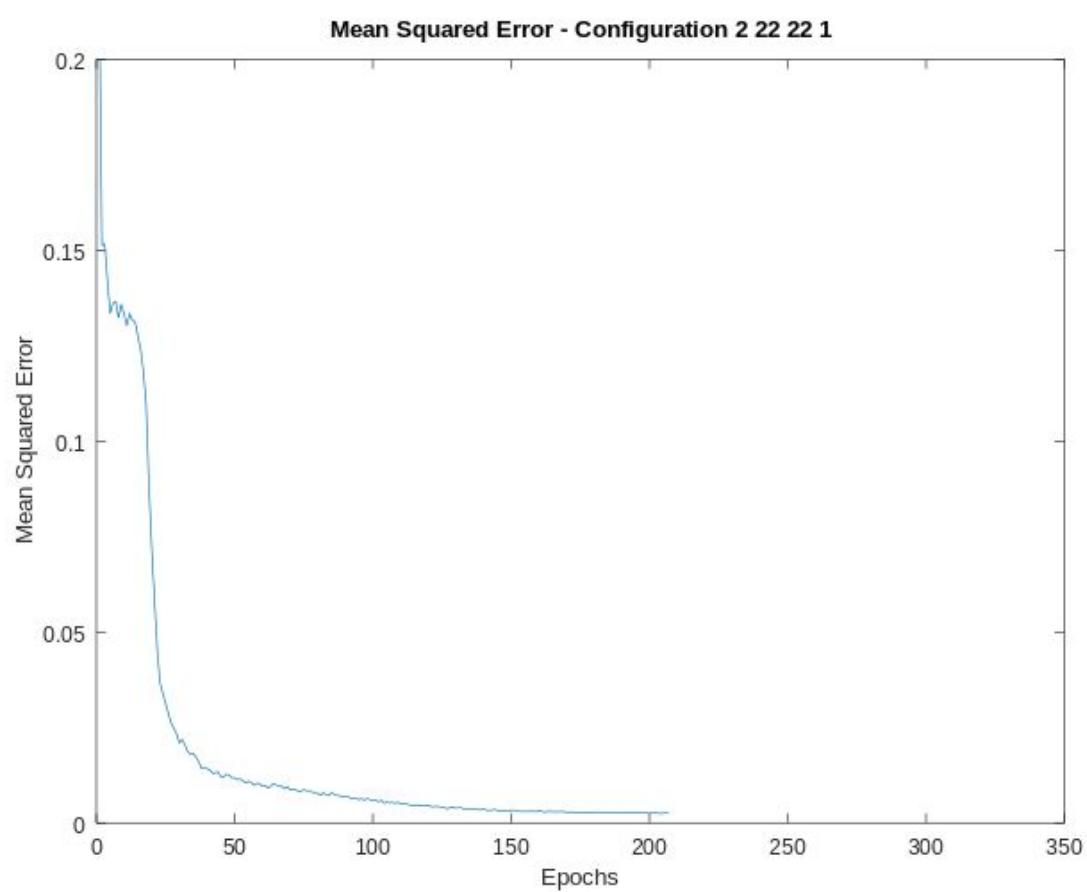
**Both terrain positions - Configuration 2 10 10 1**



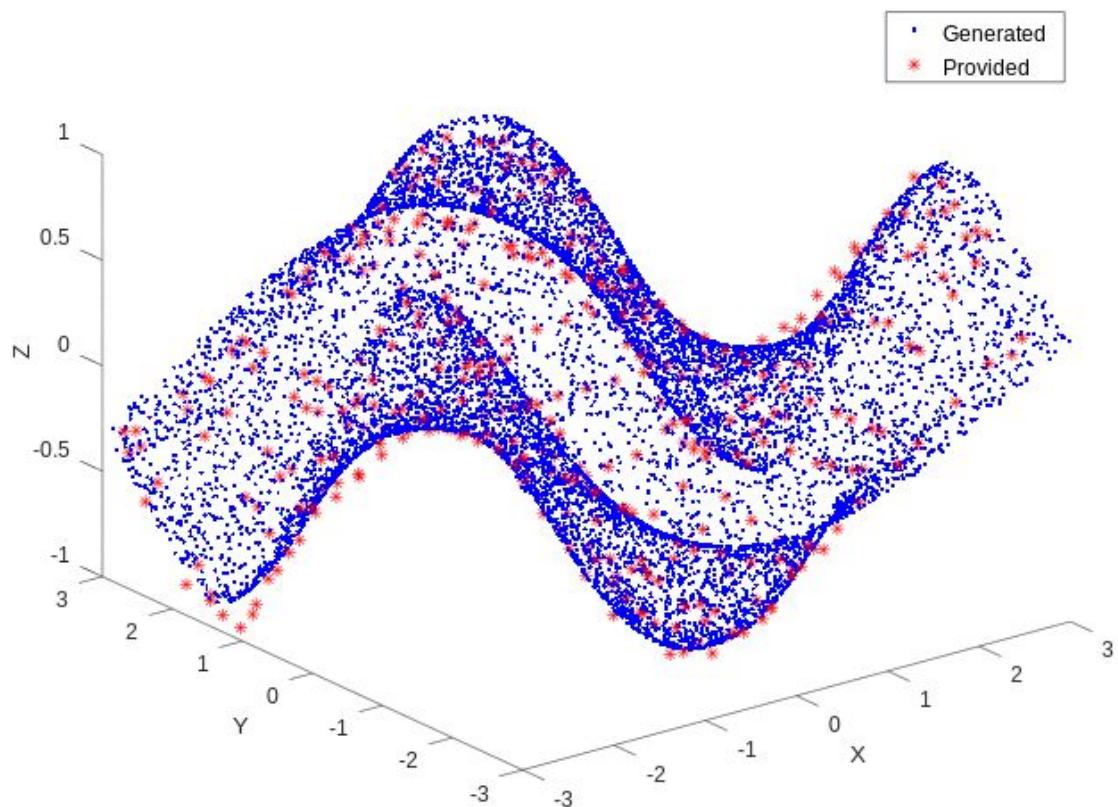


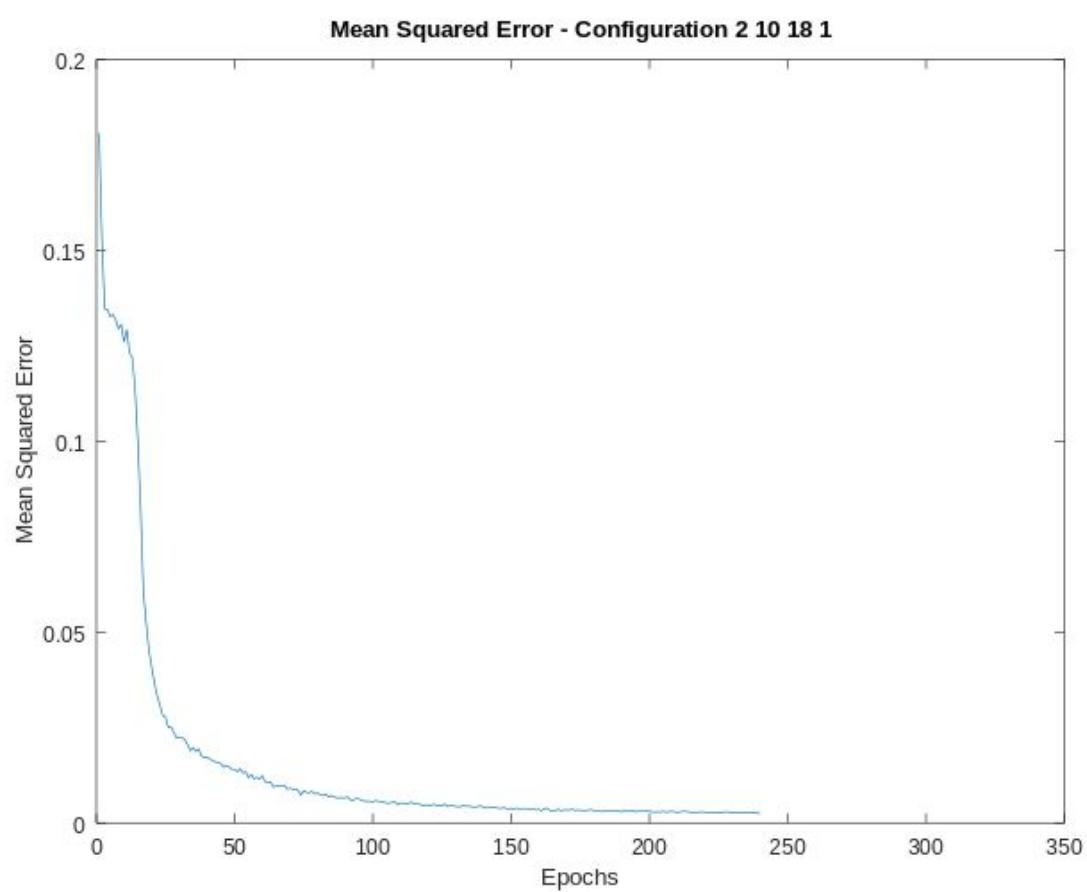
**Both terrain positions - Configuration 2 22 22 1**



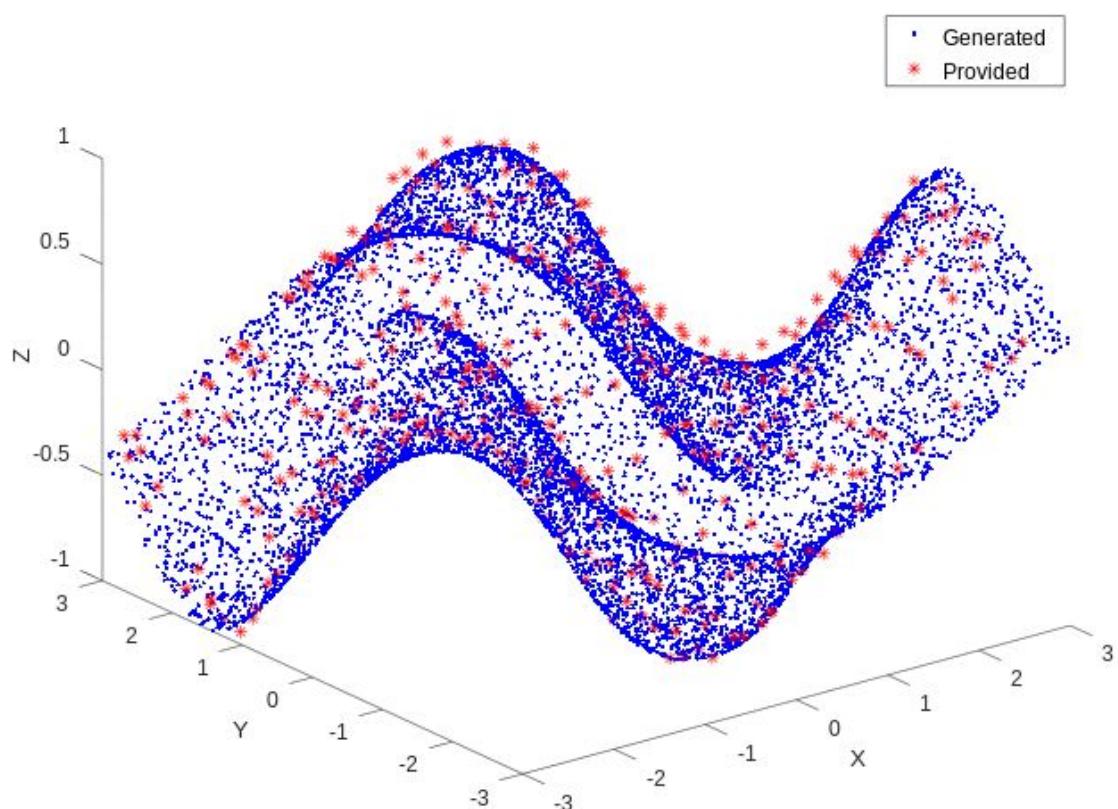


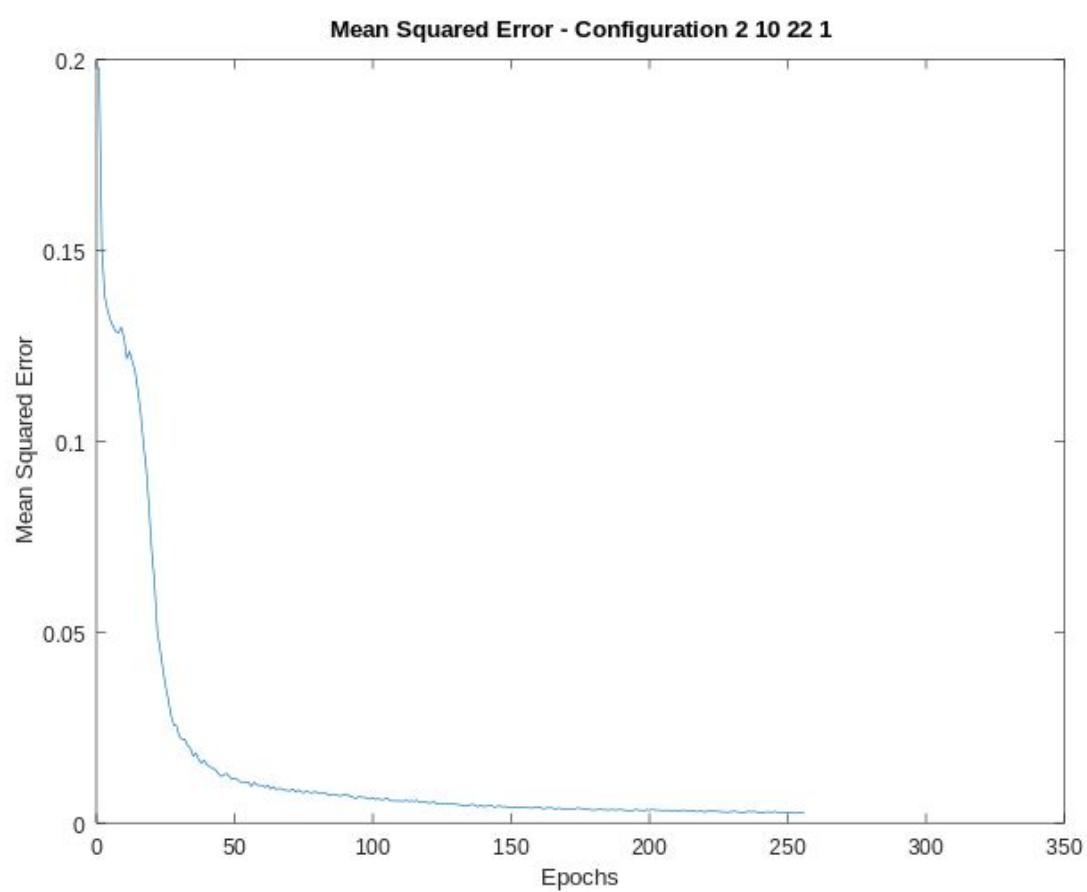
**Both terrain positions - Configuration 2 10 18 1**



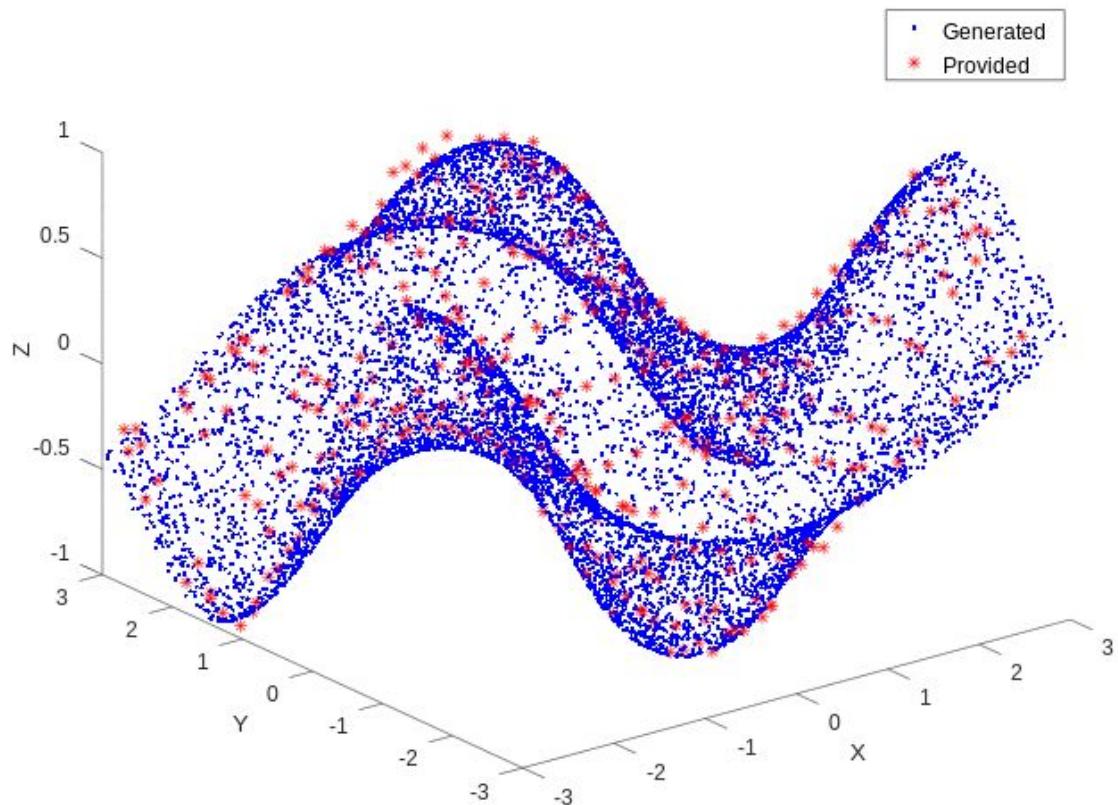


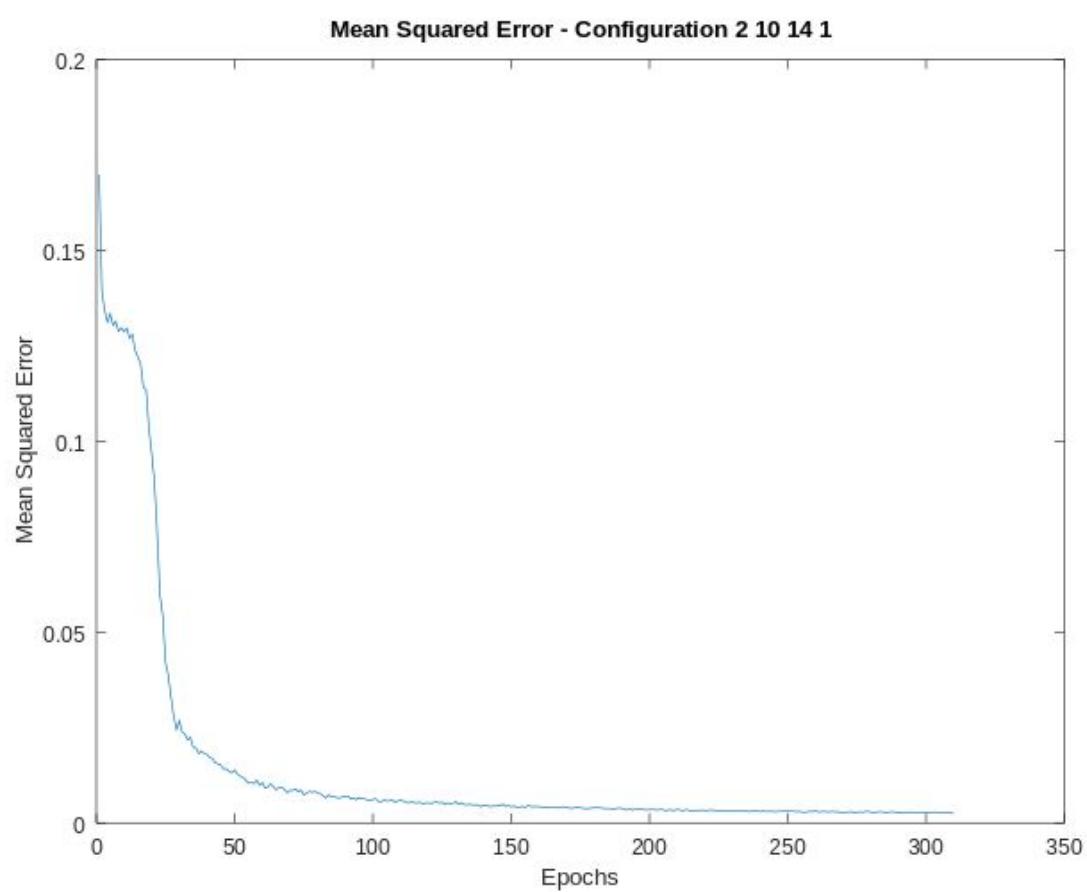
Both terrain positions - Configuration 2 10 22 1





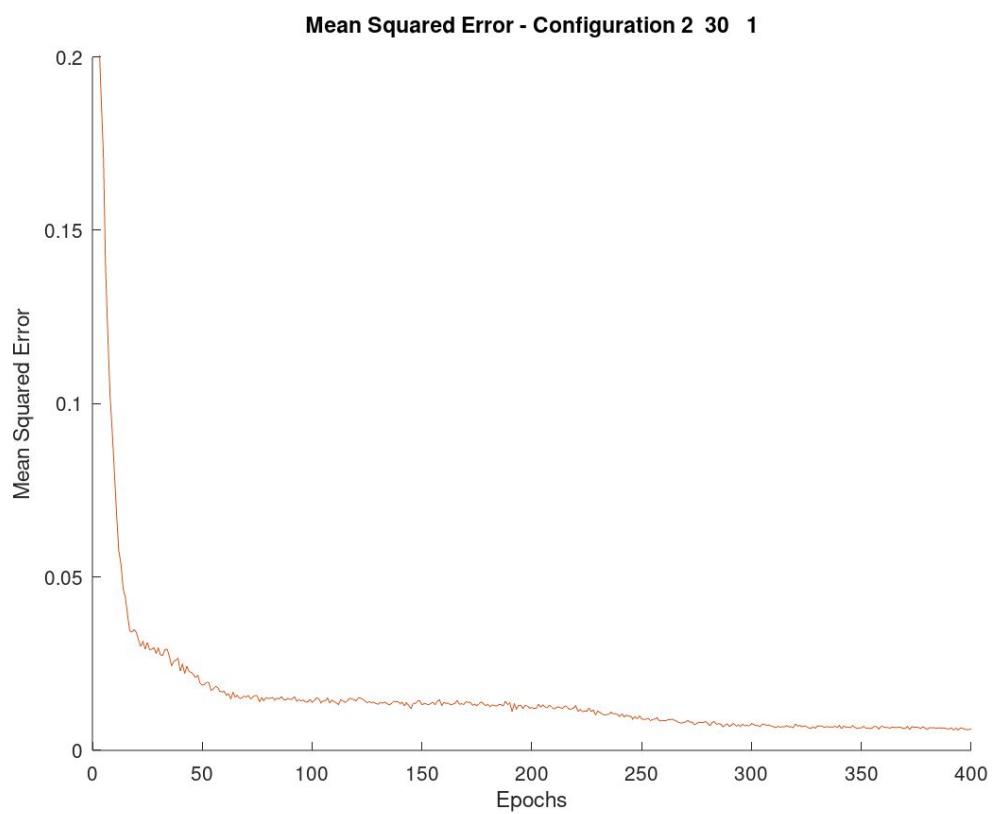
**Both terrain positions - Configuration 2 10 14 1**



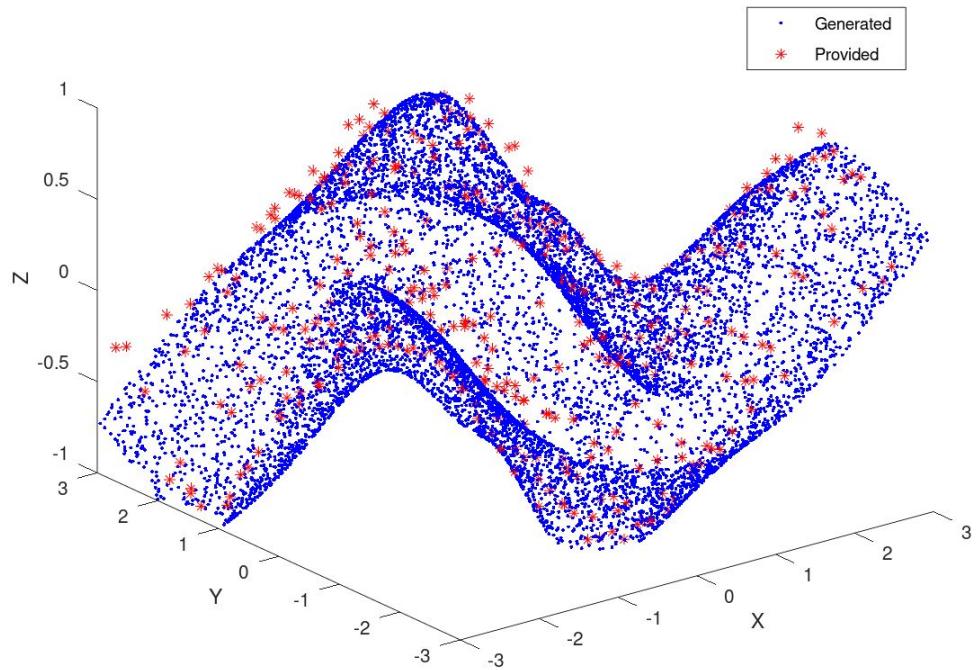


## 1 Capa

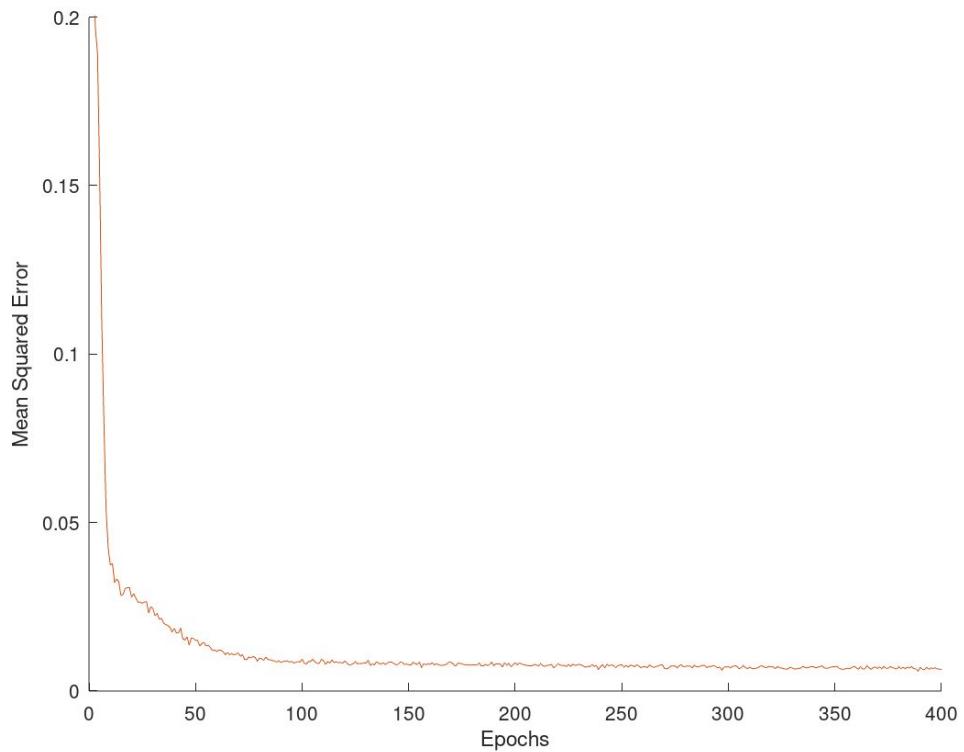
Épocas	Error	Cantidad de neuronas en capa oculta #1
400	0.006301	30
400	0.006372	28
400	0.006437	24
400	0.006493	22
400	0.006725	18



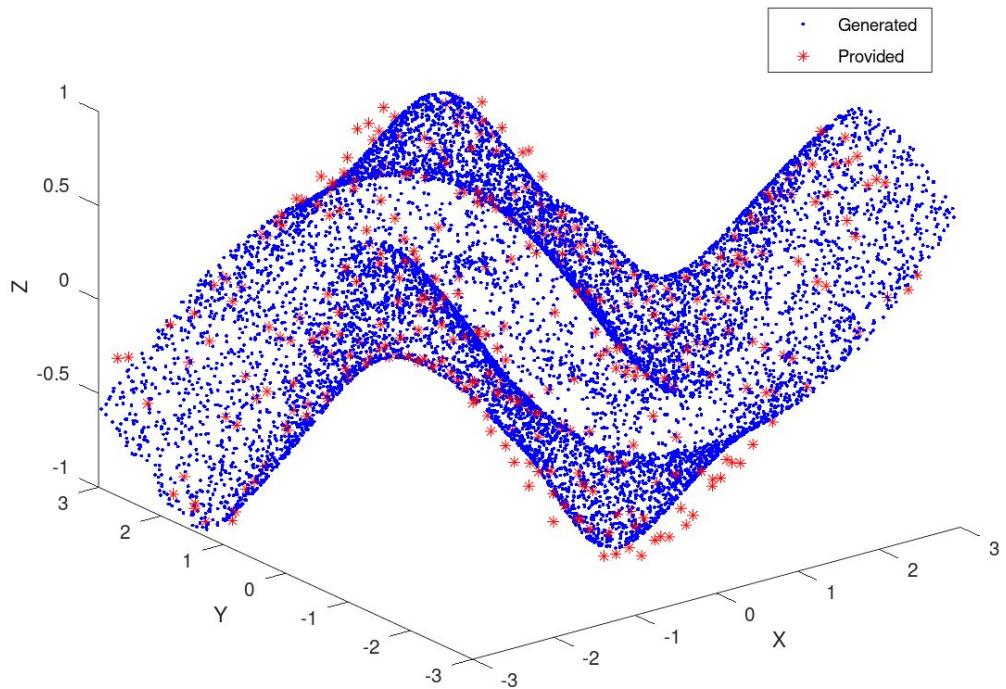
**Both terrain positions - Configuration 2 30 1**



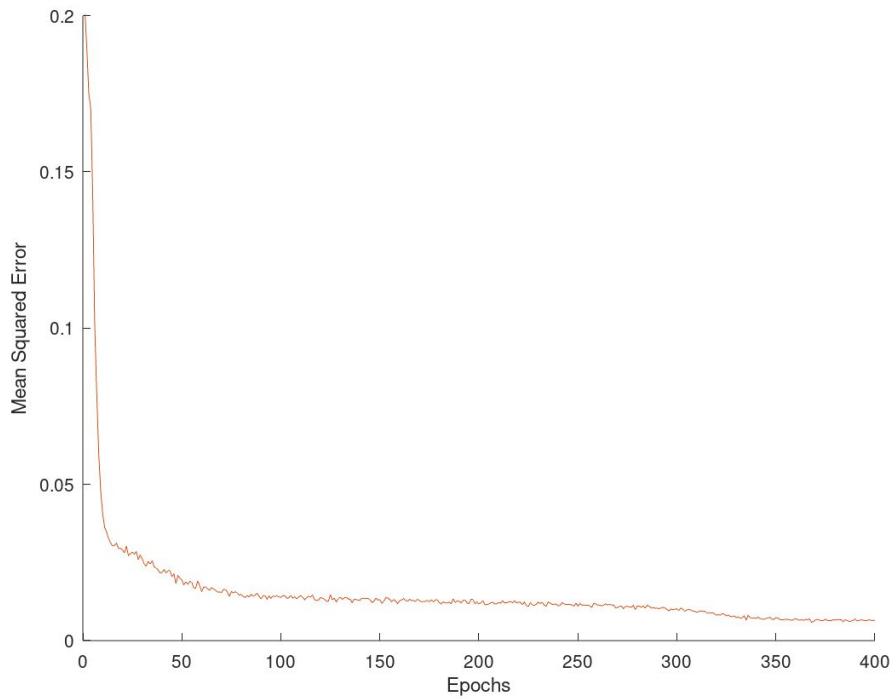
**Mean Squared Error - Configuration 2 28 1**



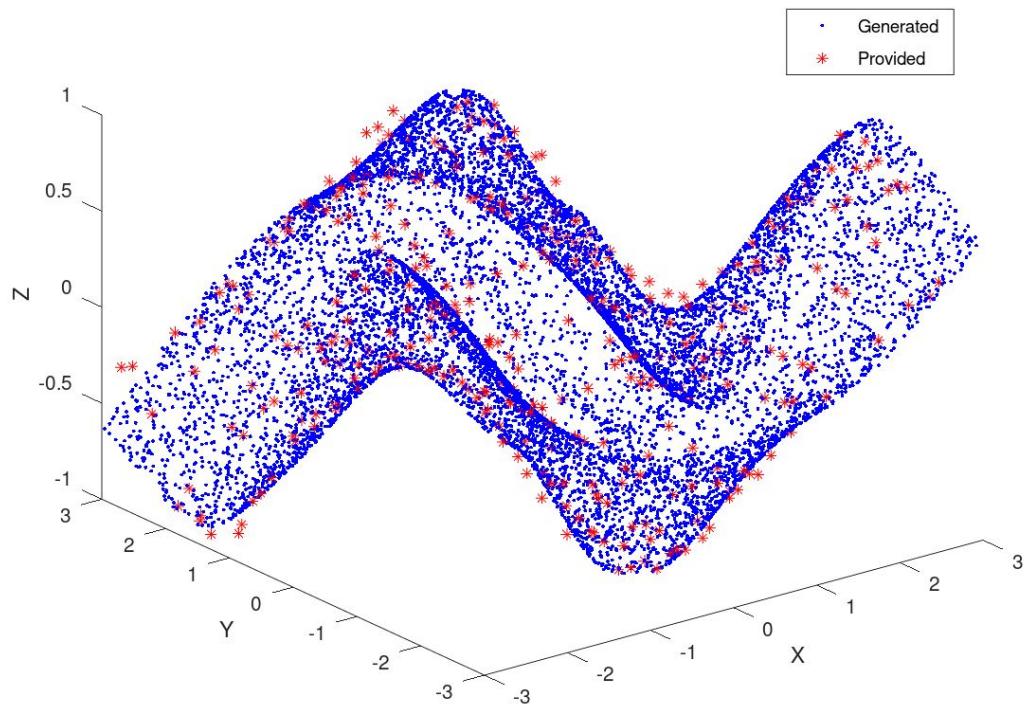
**Both terrain positions - Configuration 2 28 1**



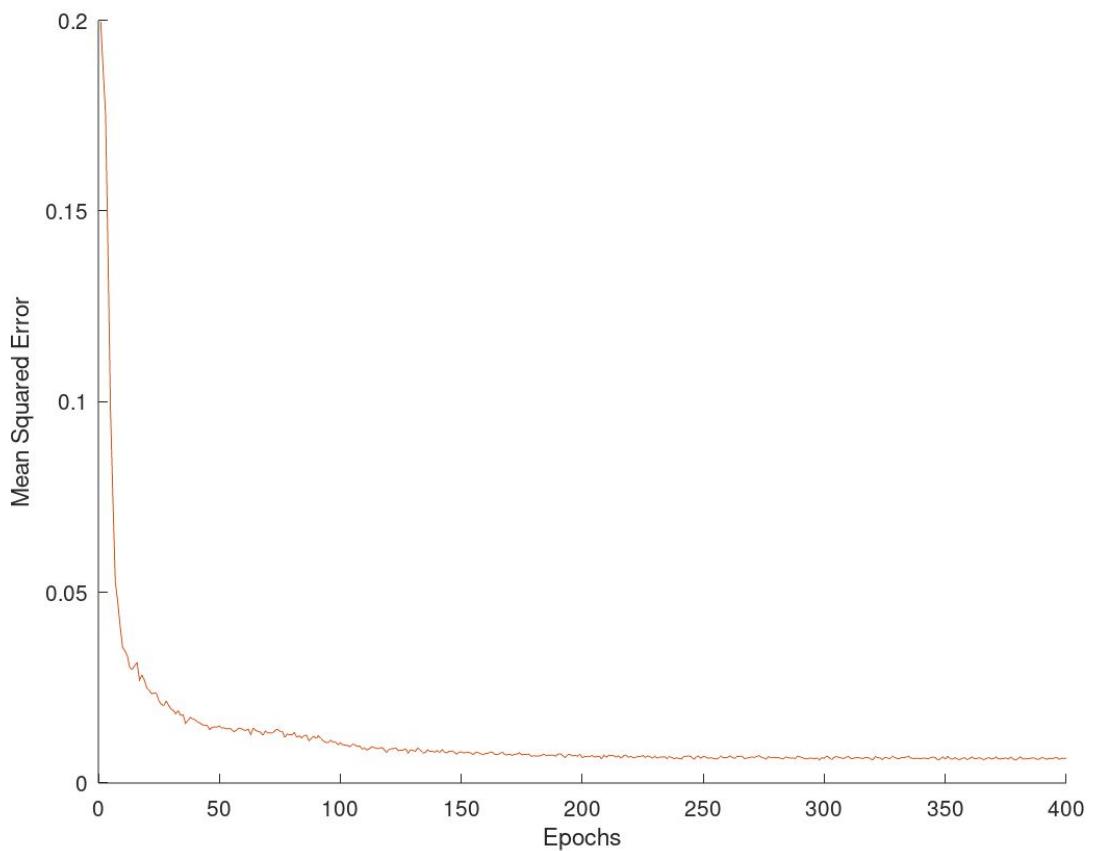
**Mean Squared Error - Configuration 2 24 1**



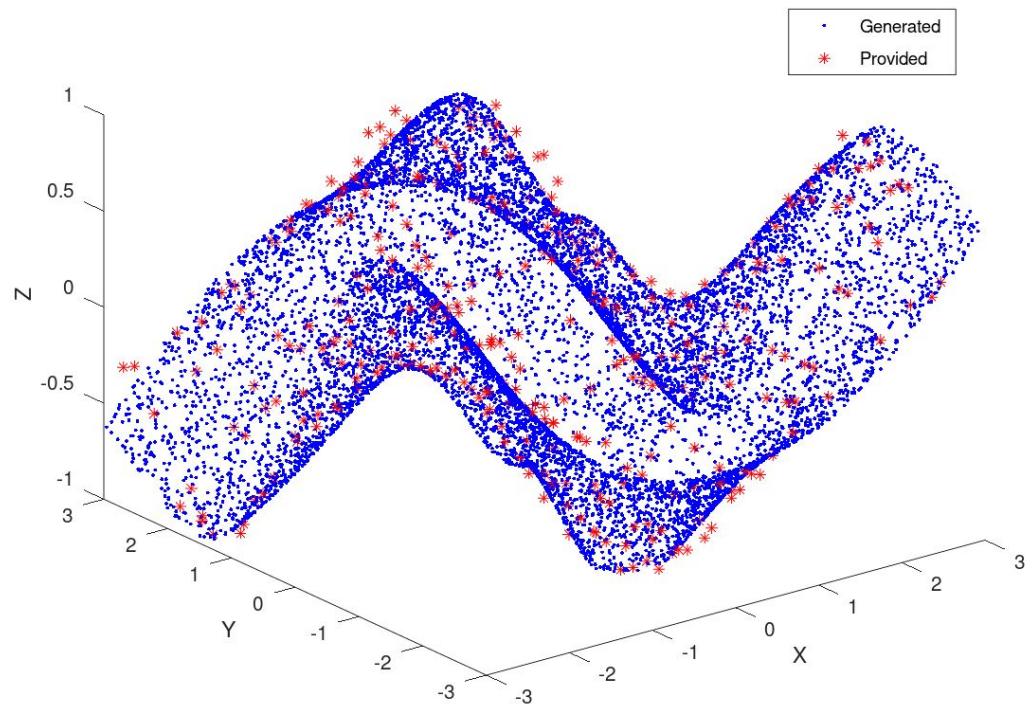
**Both terrain positions - Configuration 2 24 1**



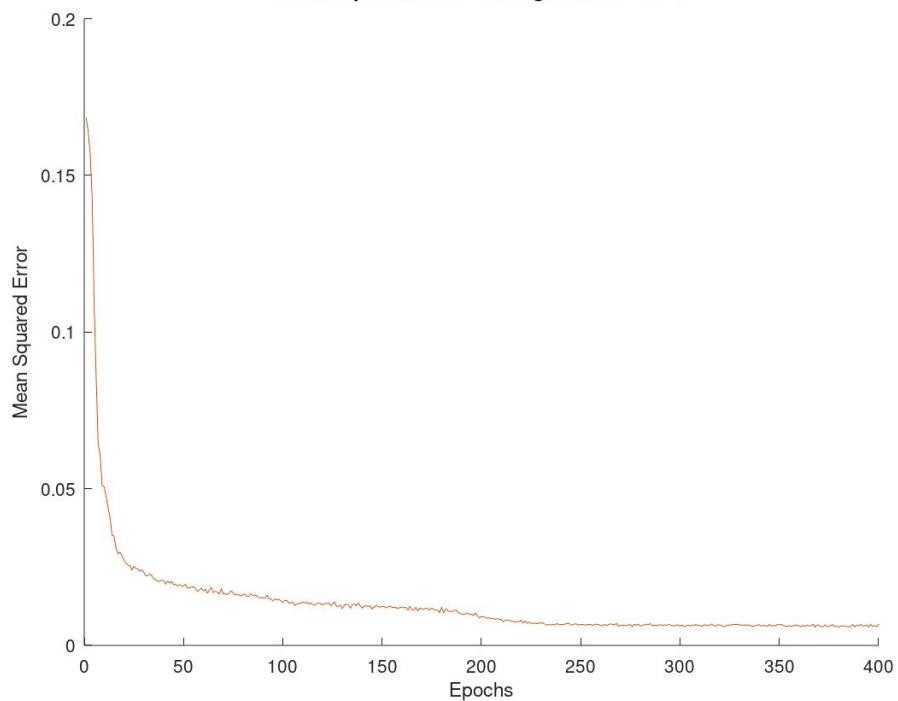
**Mean Squared Error - Configuration 2 22 1**



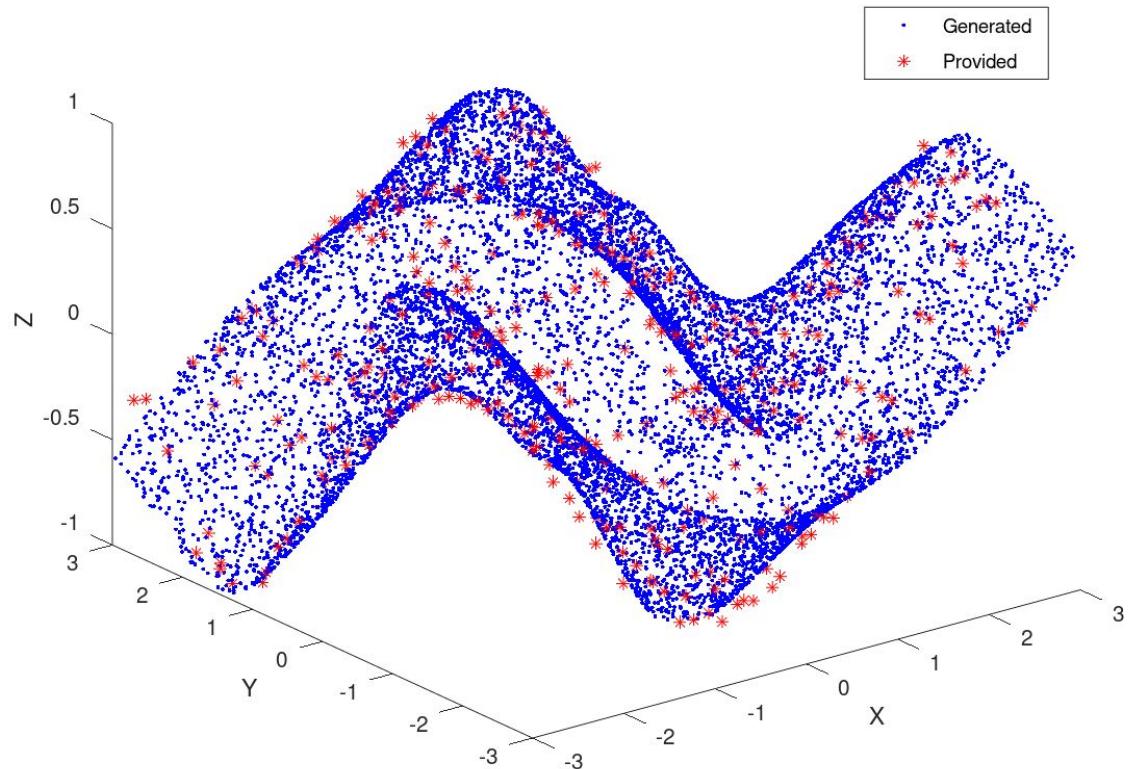
**Both terrain positions - Configuration 2 22 1**



**Mean Squared Error - Configuration 2 18 1**



**Both terrain positions - Configuration 2 18 1**



## Determinación del Factor de Aprendizaje Inicial

<b>Factor de aprendizaje inicial</b>	<b>Error</b>
0.1	0.002357
0.2	0.002262
0.3	0.002325
0.4	0.002357
0.5	0.004361
0.6	0.006950
0.7	0.005100
0.8	0.134363
0.9	0.210761

<b>Factor de aprendizaje inicial</b>	<b>Error</b>
0.21	0.002426
0.22	0.002387
0.23	0.002417
0.24	0.002433
0.25	0.002162
0.26	0.002414
0.27	0.002370
0.28	0.002249
0.29	0.002229

## Determinación del Alfa del Momento

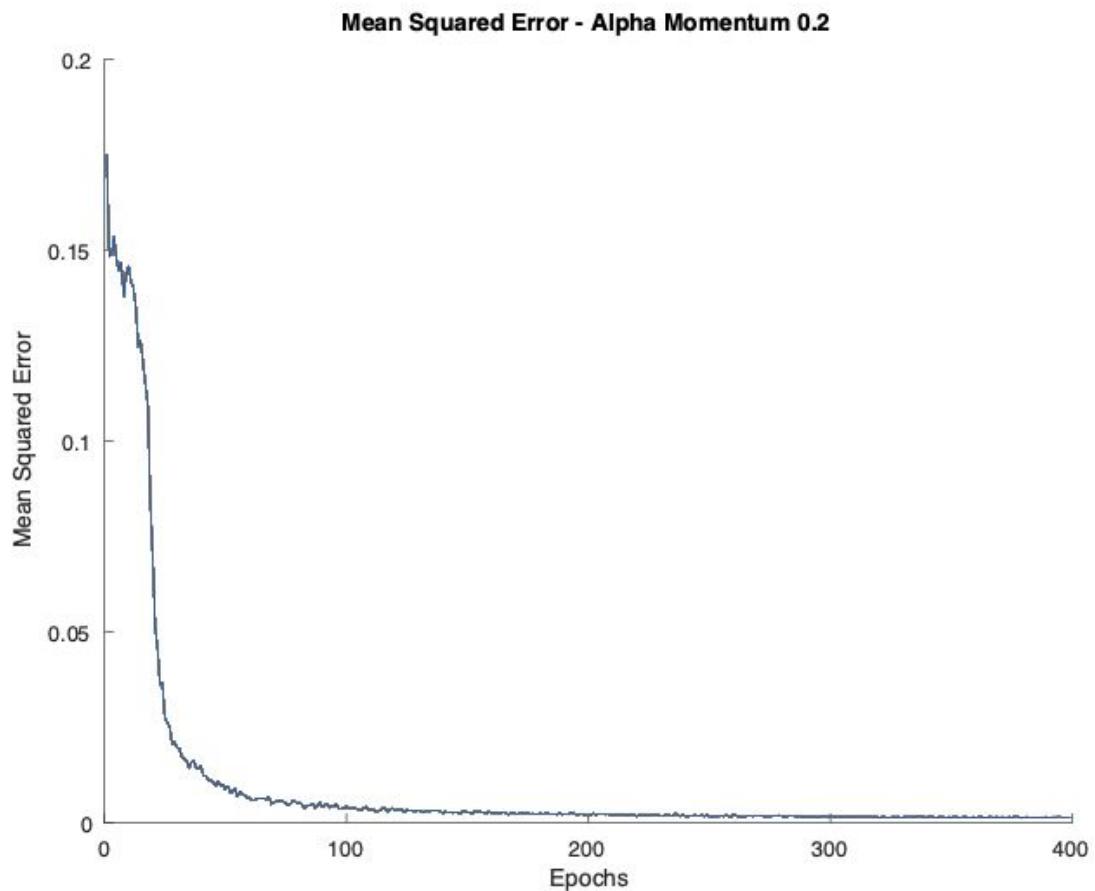
Alfa del Momento	Error
0.1	0.001493
0.2	0.001439
0.3	0.001489
0.4	0.001688
0.5	0.001847
0.6	0.001605
0.7	0.003923
0.8	0.003447
0.9	0.236886

A partir de estos resultados podemos ver que el mejor alfa es de 0.2 seguido del de 0.3.  
Analicemos ahora en el intervalo [0.2, 03)

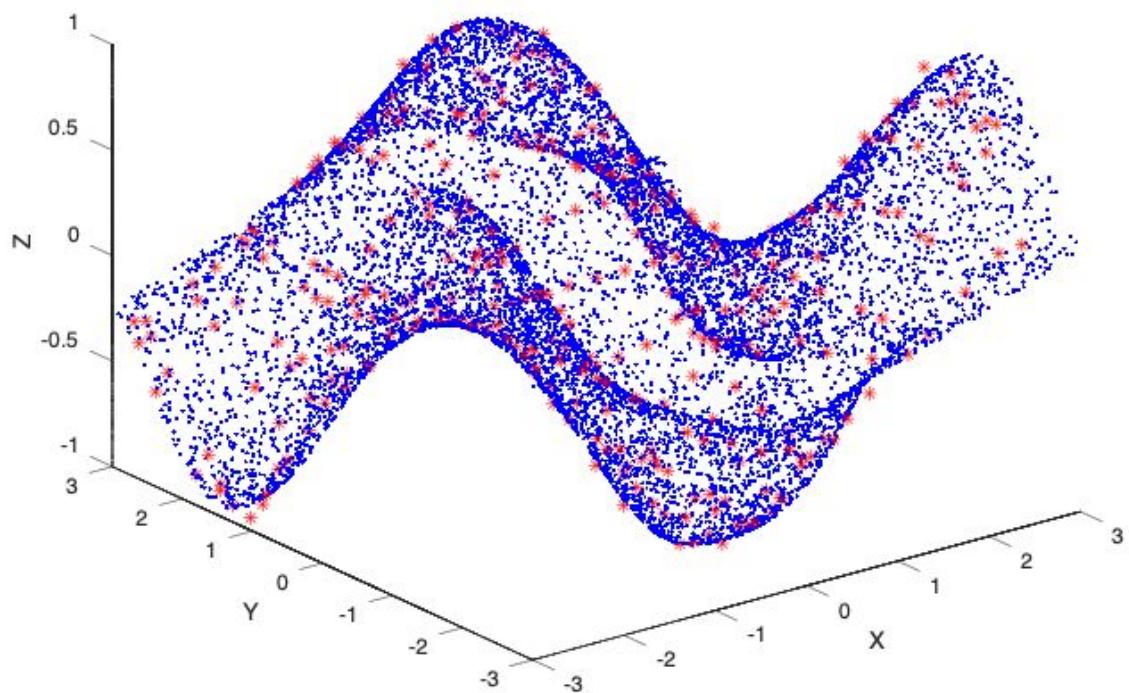
Alfa del momento	Error
0.2	0.001439
0.21	0.001447
0.22	0.001456
0.23	0.001463
0.24	0.001471
0.25	0.001478
0.26	0.001483
0.27	0.001488
0.28	0.001490
0.29	0.001489

Se puede ver que el mejor alfa es 0.2, veamos los gráficos del error de las tres mejores

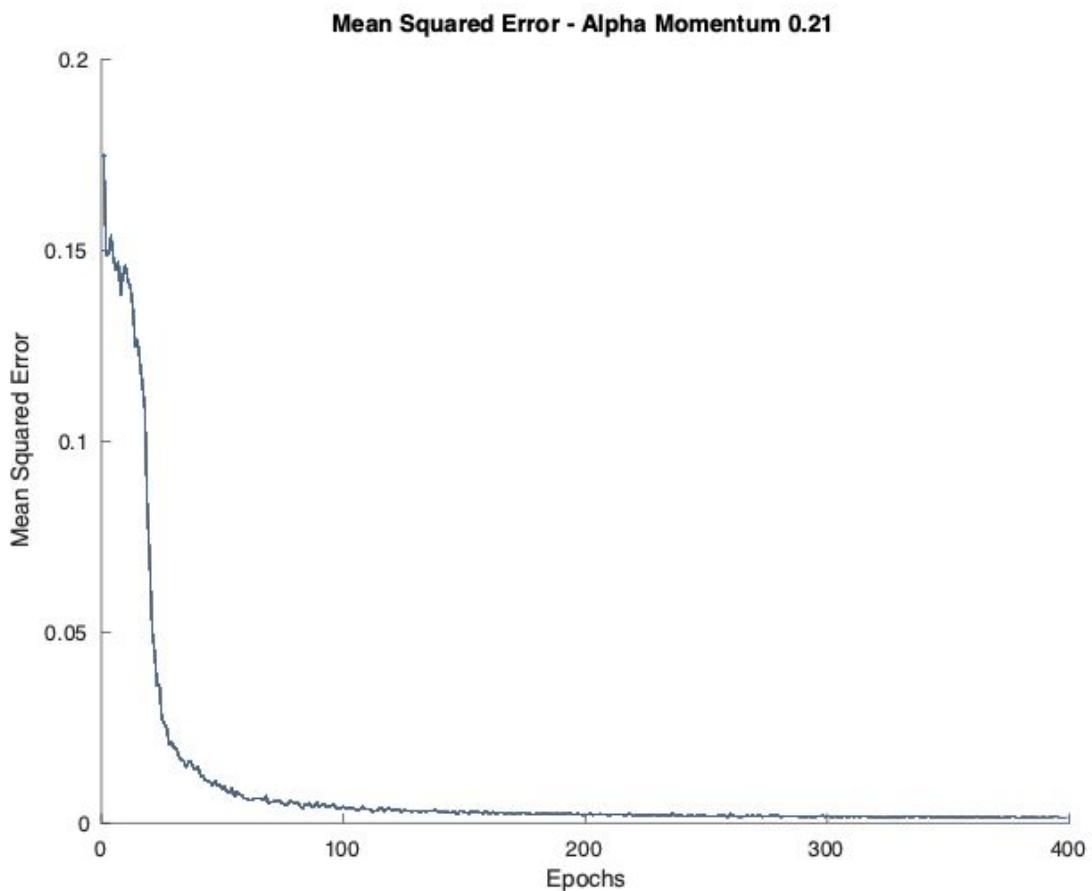
Alfa del momento = 0.2



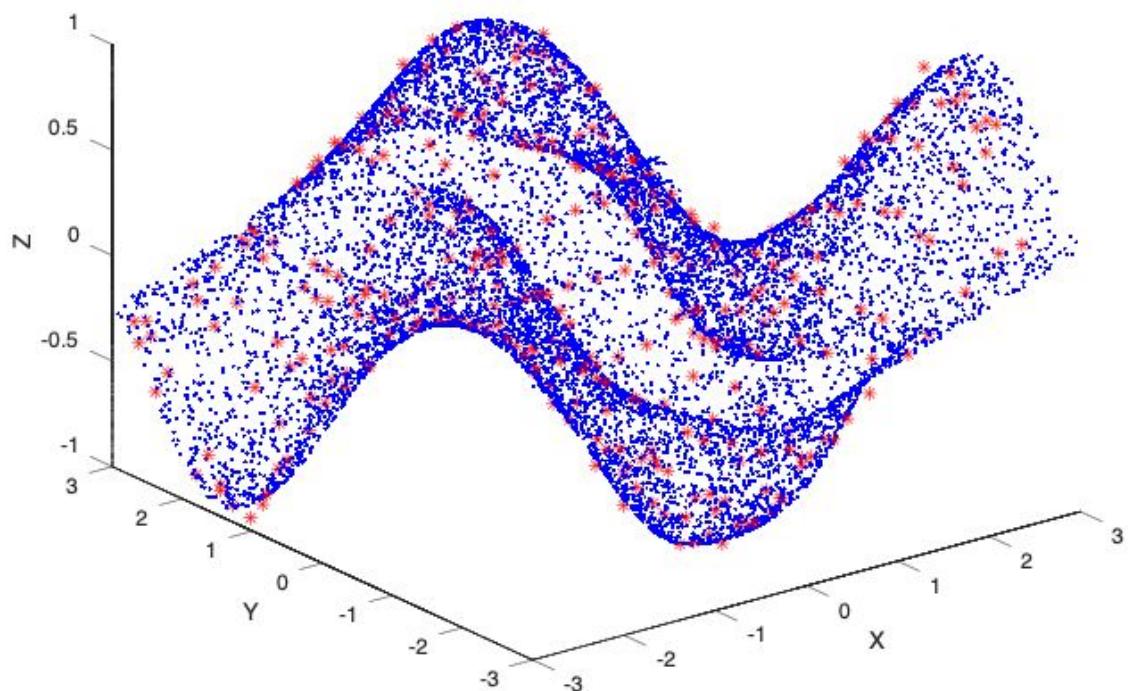
**Generated terrain positions - Alpha Momentum 0.2**



Alfa del momento = 0.21

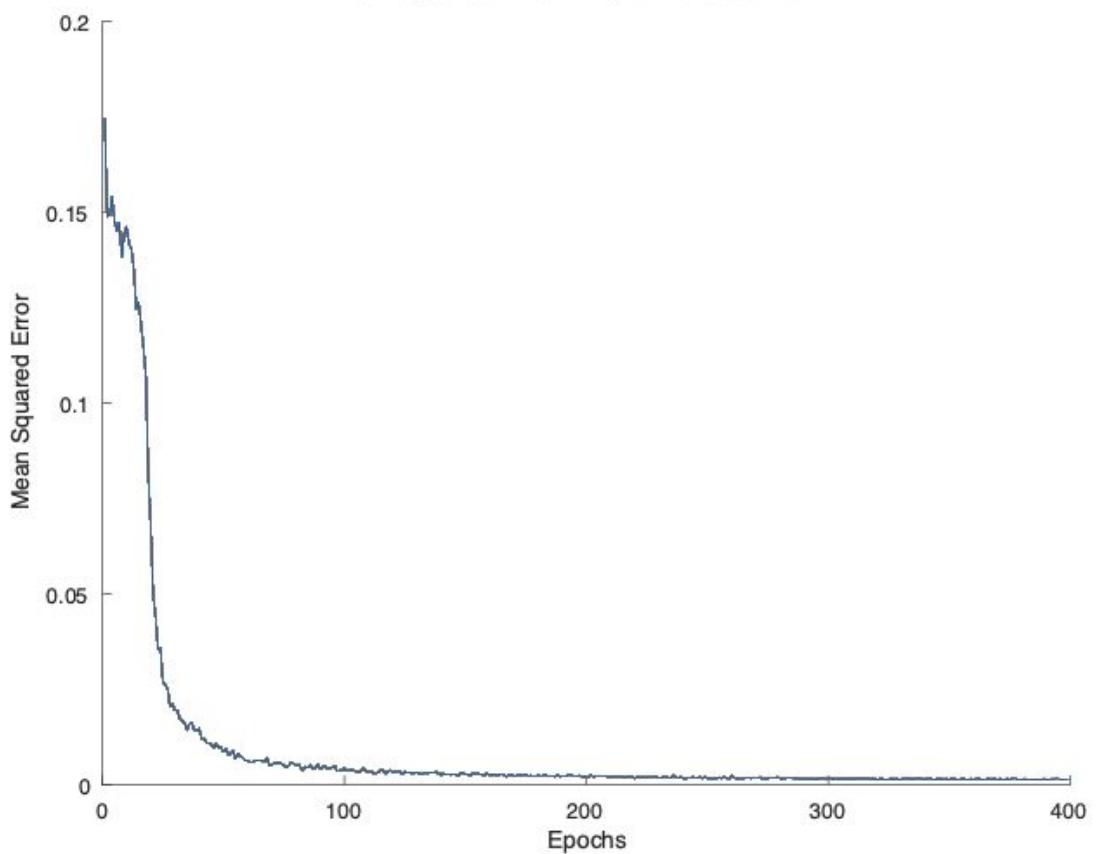


**Generated terrain positions - Alpha Momentum 0.21**

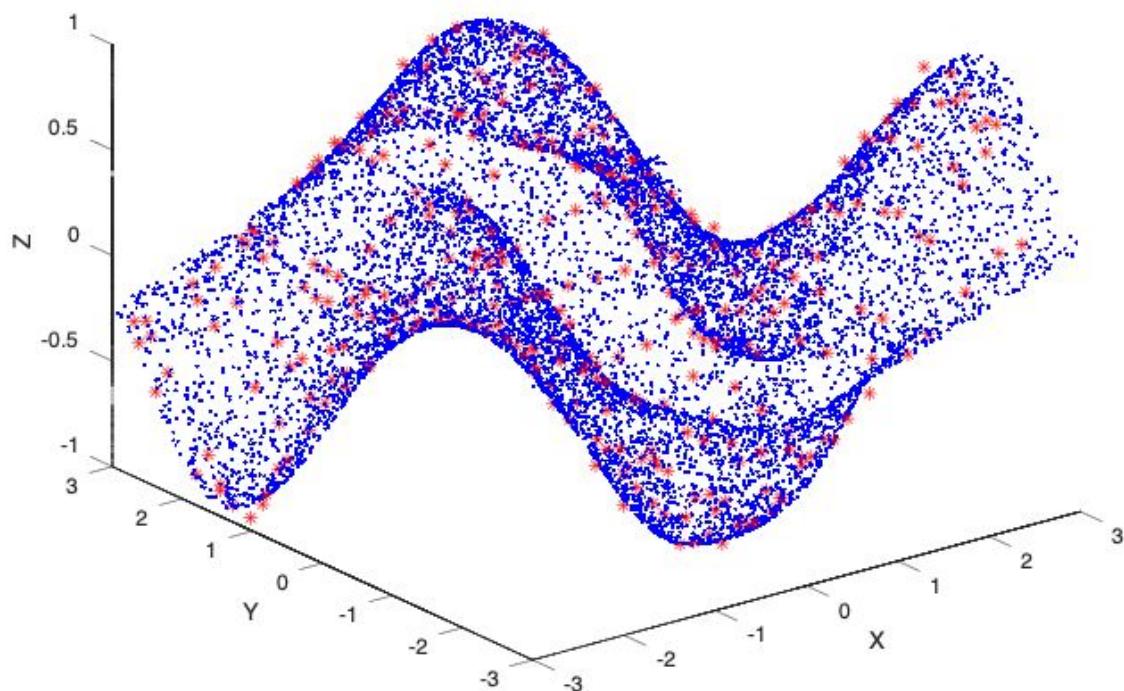


Alfa del momento = 0.22

**Mean Squared Error - Alpha Momentum 0.22**



**Generated terrain positions - Alpha Momentum 0.22**

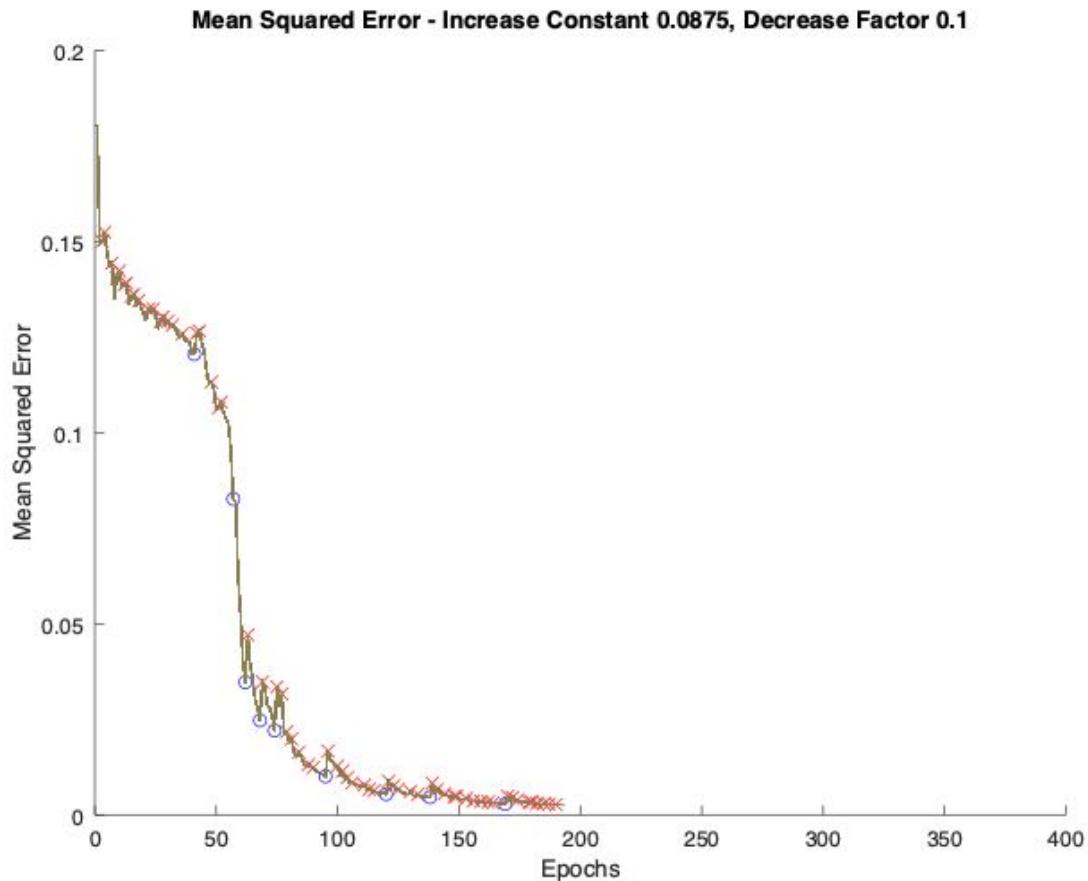


Como sucede con la tabla, en los gráficos no se observa casi diferencia. Basándonos en los errores obtenidos podemos decir que el mejor alfa para el momento es un alfa = 0.2

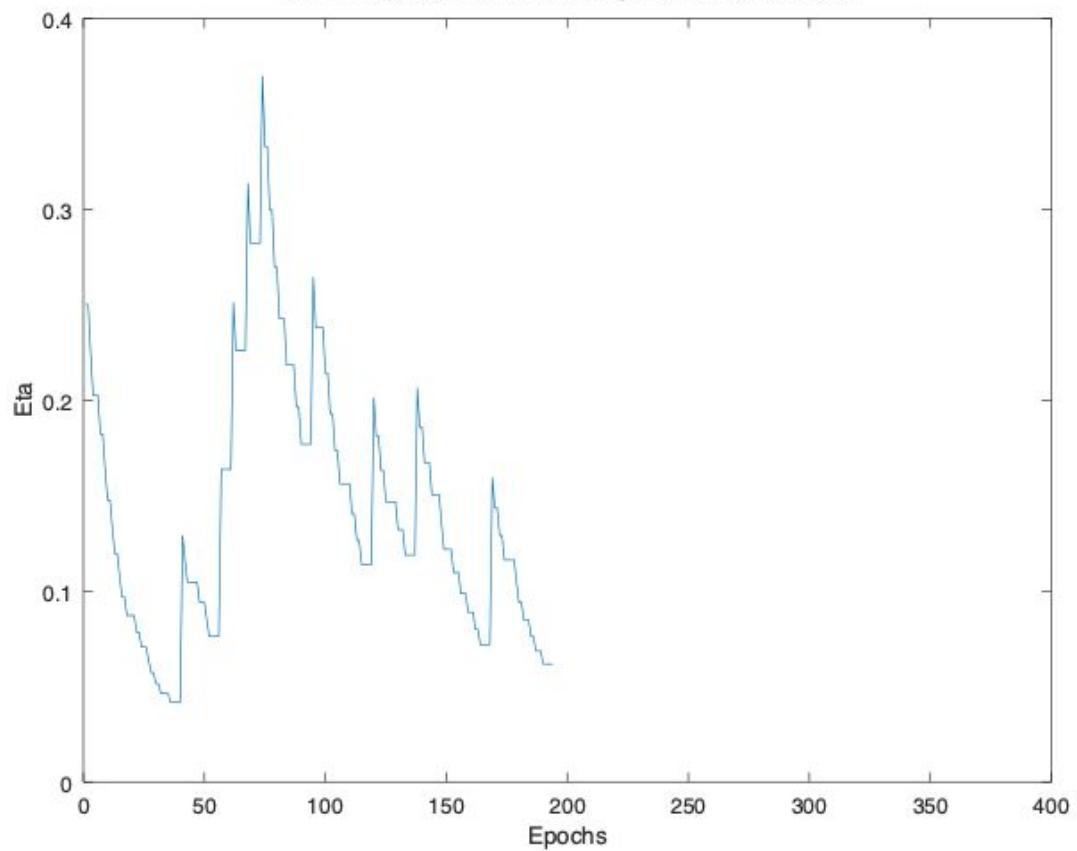
## Determinación de las Constantes del Eta Adaptativo

Épocas	Error	Constante de aumento	Factor de decrecimiento
194	0.002446	0.0875	0.1
196	0.002332	0.125	0.1
400	0.003255	0.125	0.25
400	0.002628	0.05	0.1
400	0.004069	0.0875	0.25

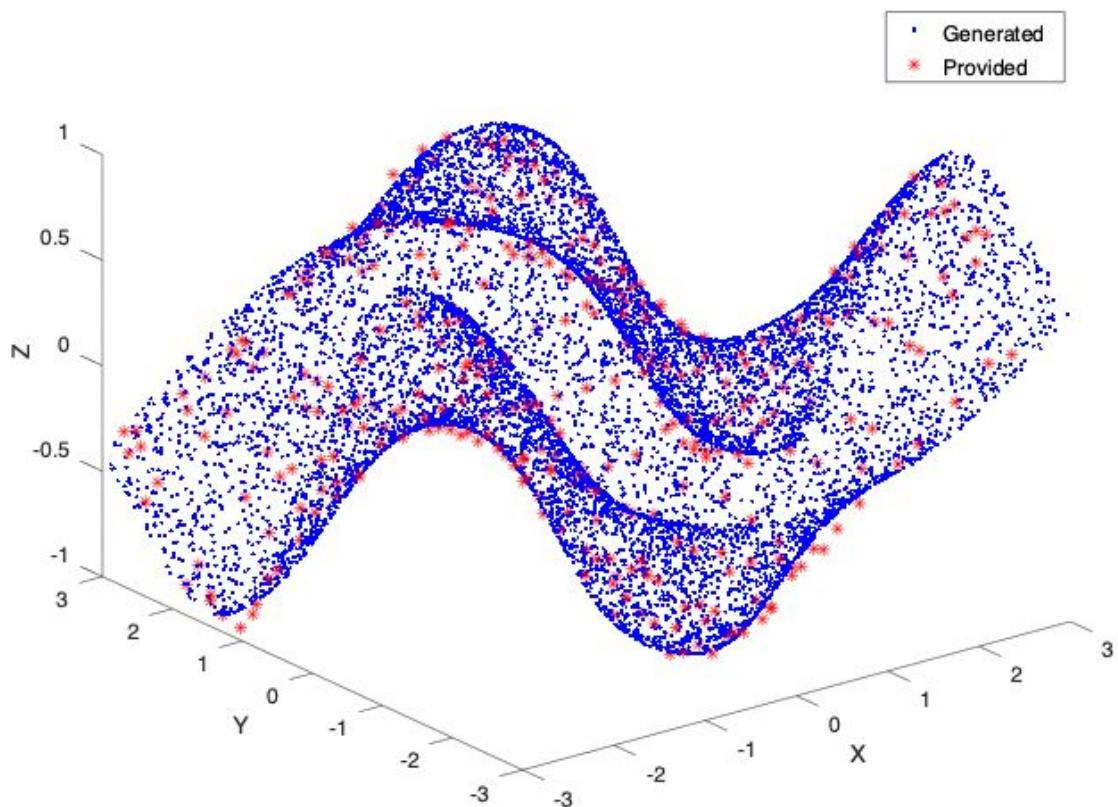
En los siguientes gráficos de error cuadrático medio se grafican cruces rojas cuando el algoritmo de eta adaptativo detecta un aumento en el error, y por ende baja el factor de aprendizaje. Por otro lado, cuando el algoritmo detecta una disminución de error consistente, y aumenta el factor de aprendizaje, se grafica un círculo azul.



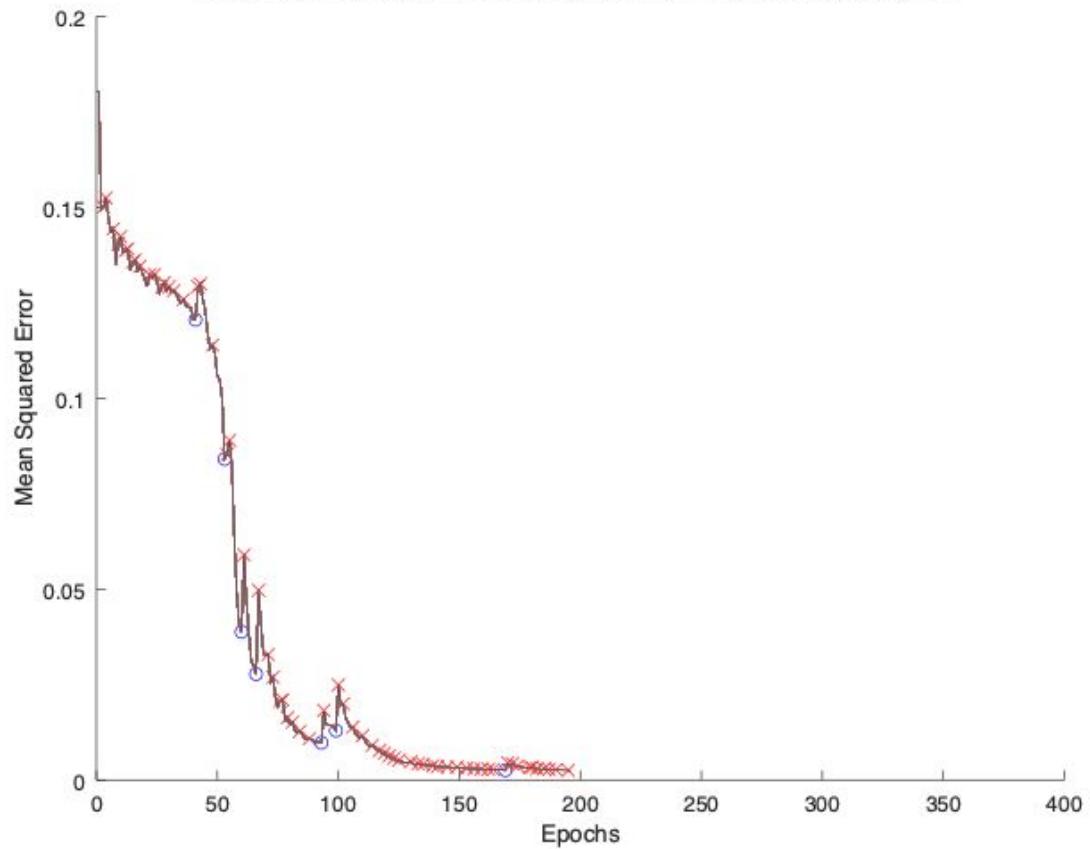
**Eta - Increase Constant 0.0875, Decrease Factor 0.1**



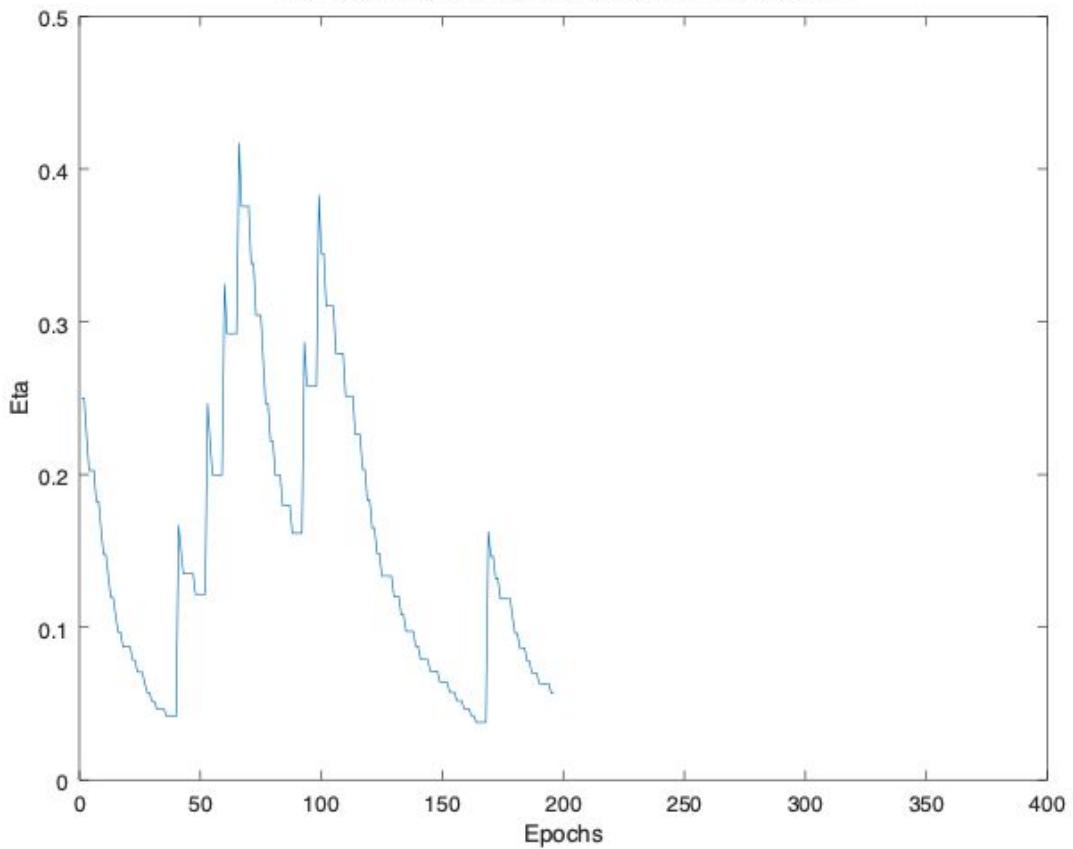
**Both terrain positions - Increase Constant 0.0875, Decrease Factor 0.1**



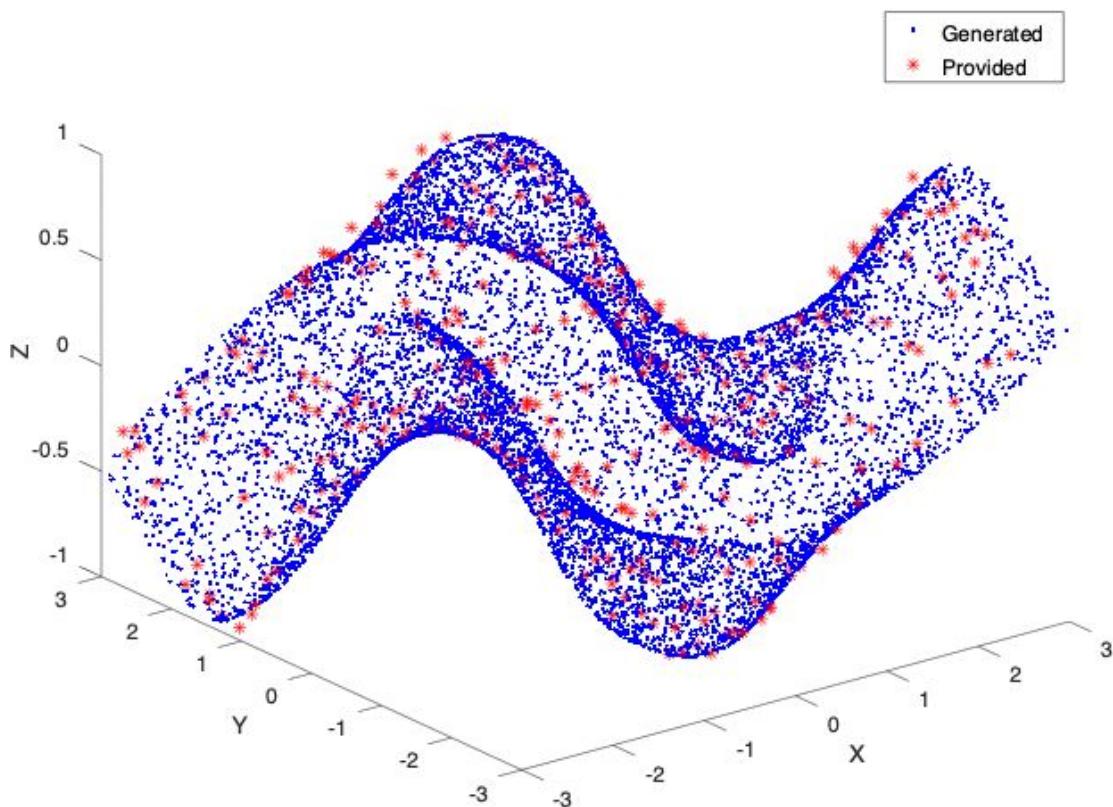
**Mean Squared Error - Increase Constant 0.125, Decrease Factor 0.1**



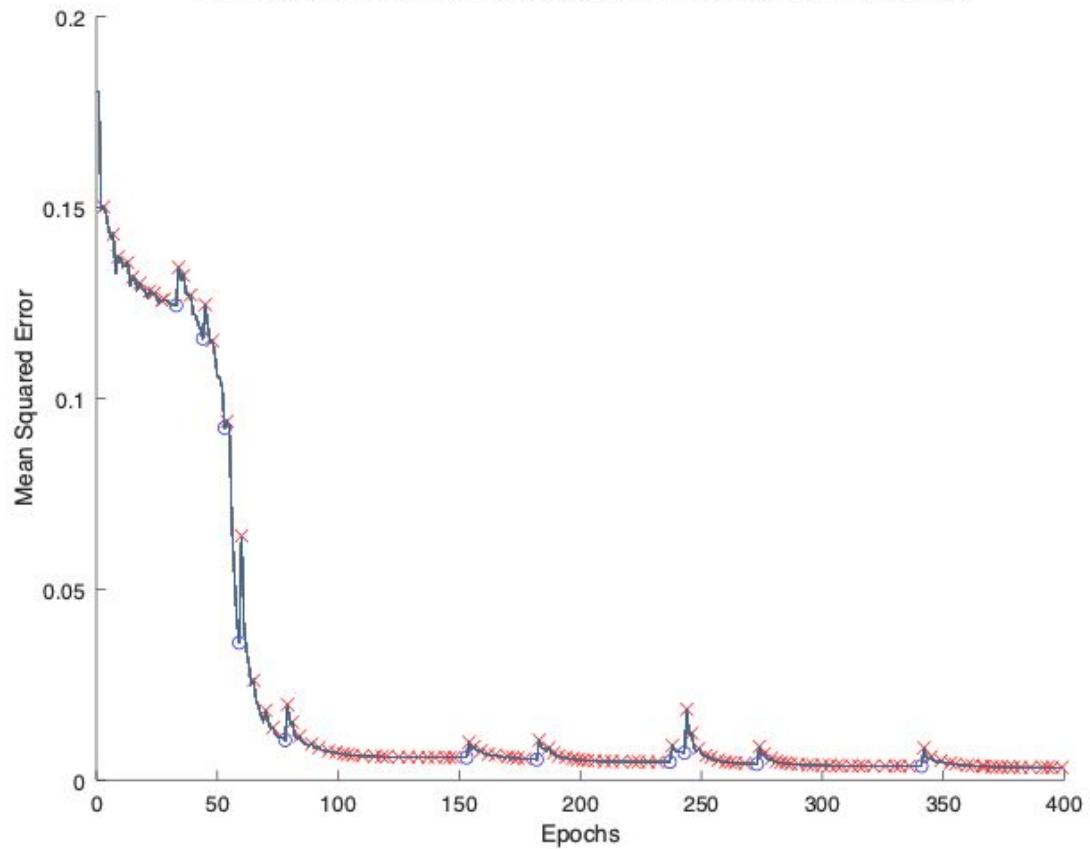
**Eta - Increase Constant 0.125, Decrease Factor 0.1**



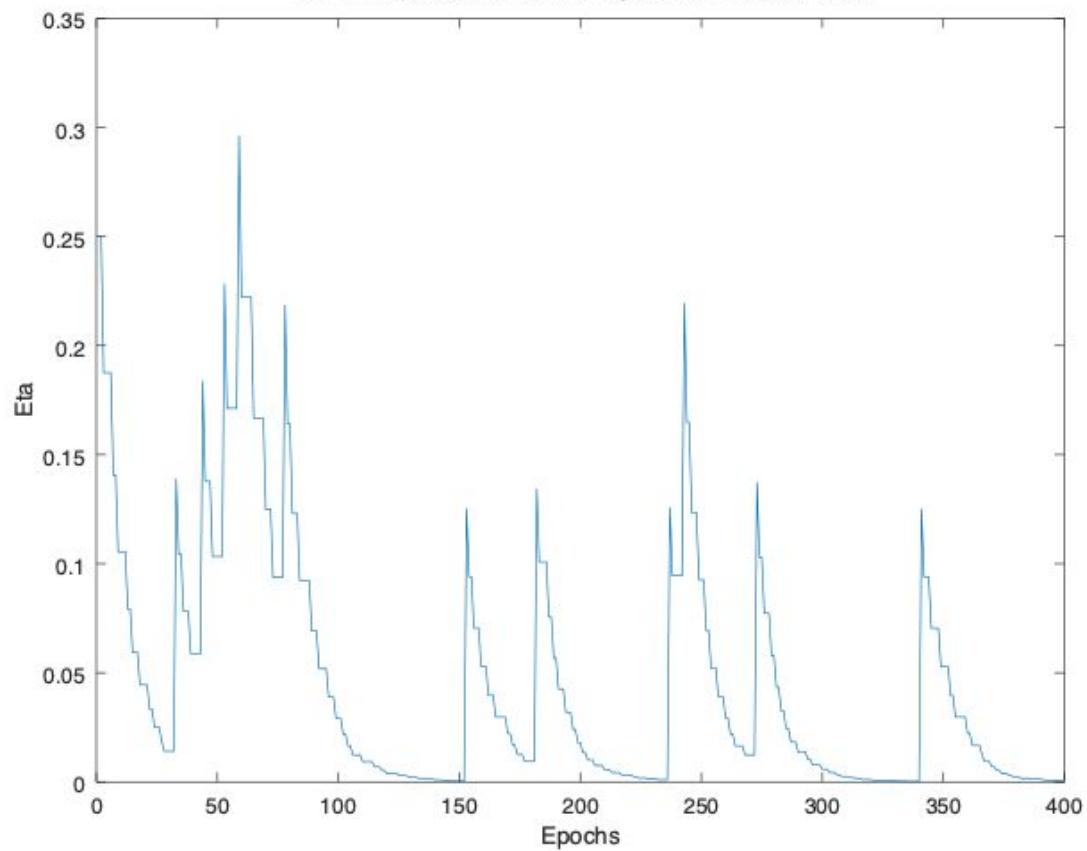
**Both terrain positions - Increase Constant 0.125, Decrease Factor 0.1**



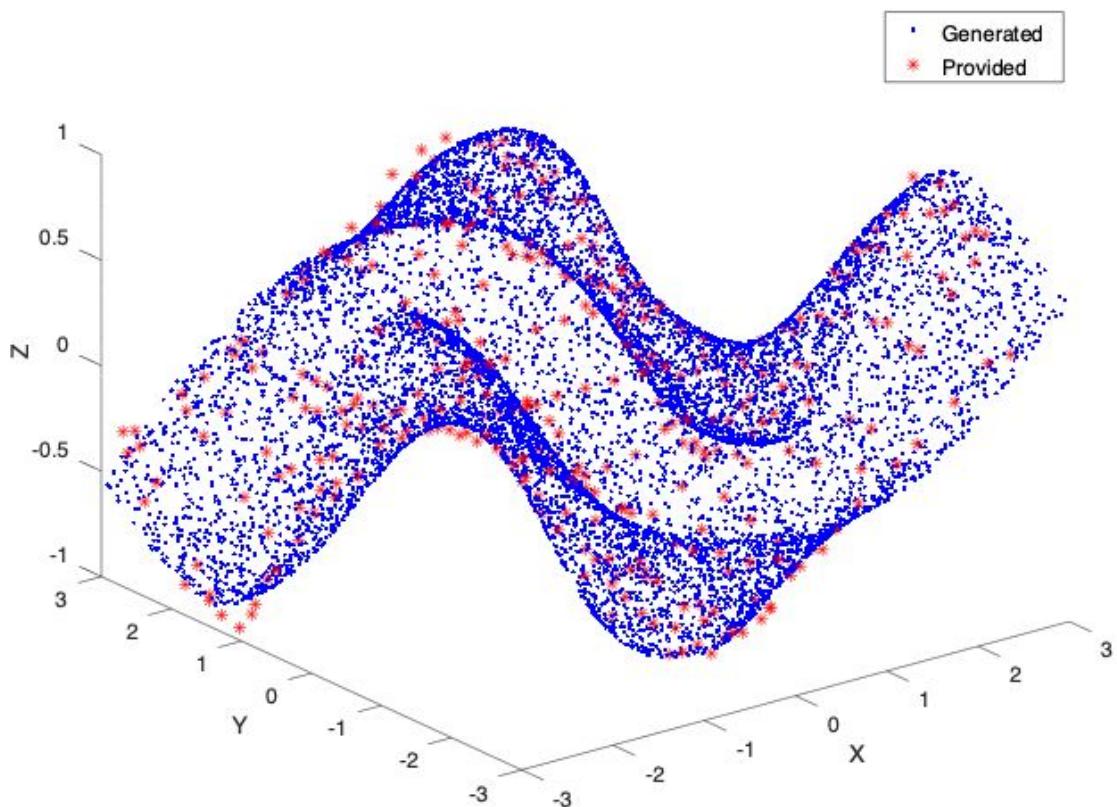
**Mean Squared Error - Increase Constant 0.125, Decrease Factor 0.25**



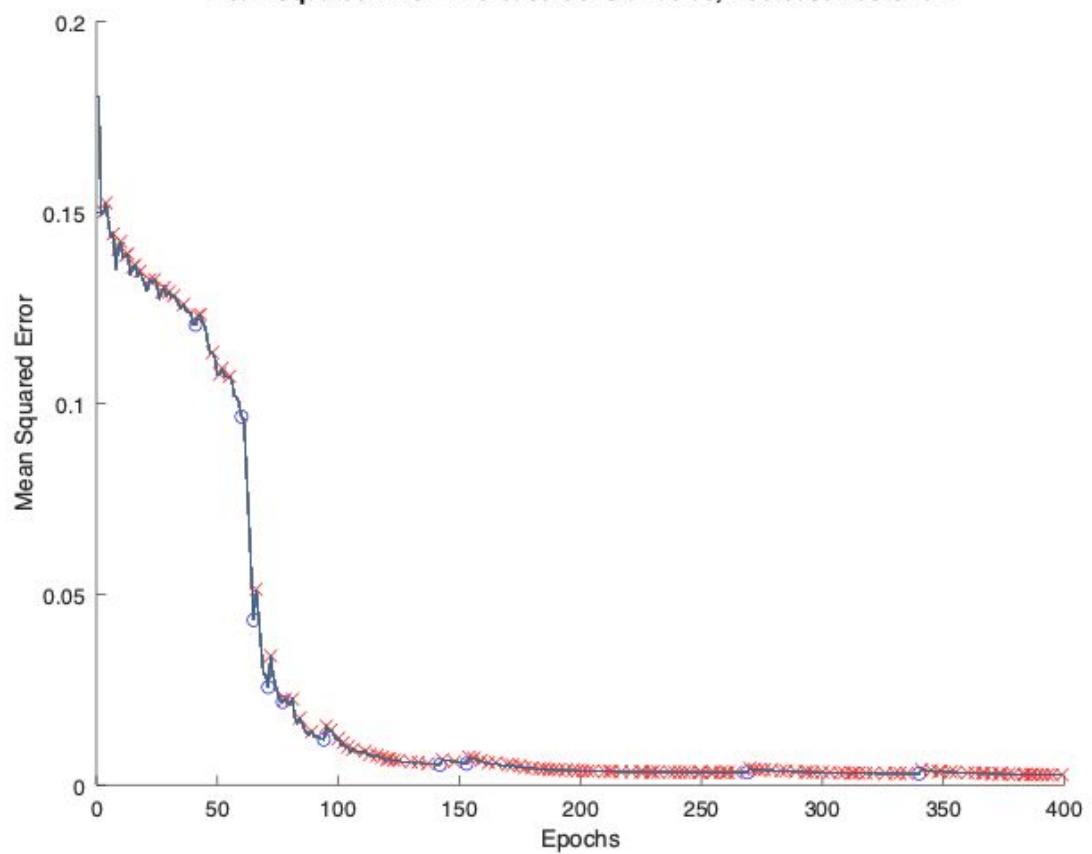
**Eta - Increase Constant 0.125, Decrease Factor 0.25**



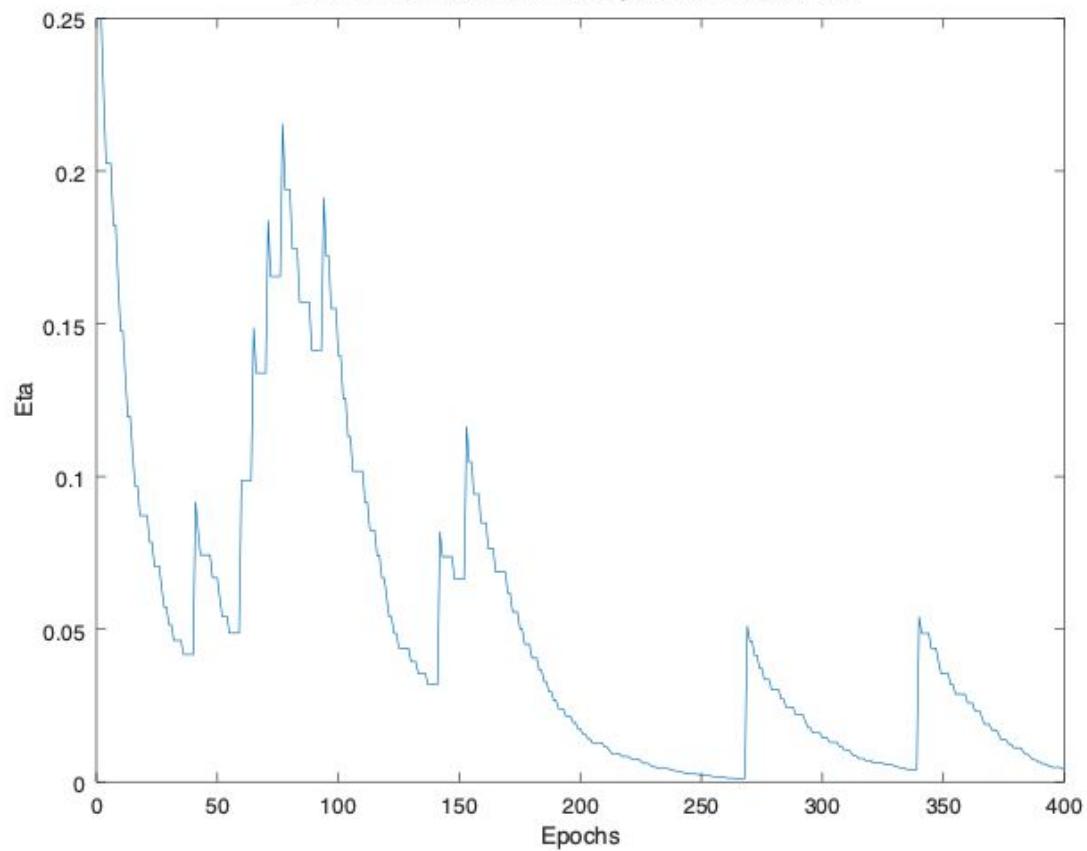
**Both terrain positions - Increase Constant 0.125, Decrease Factor 0.25**



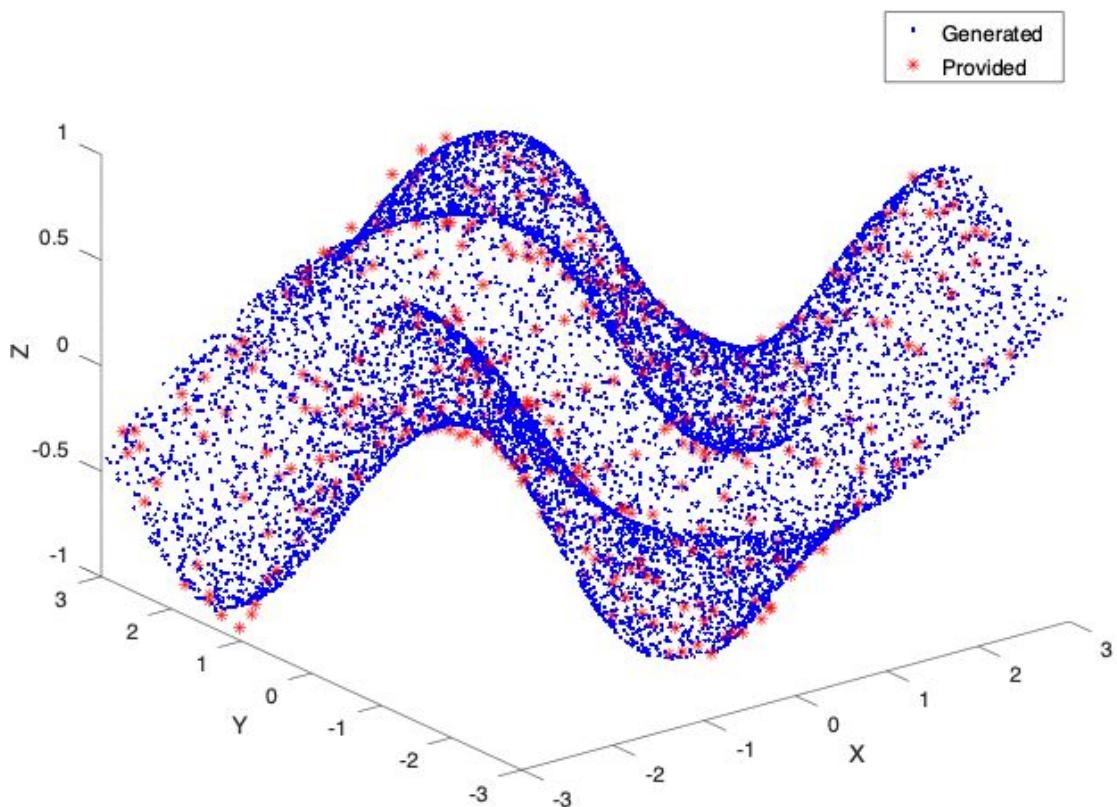
**Mean Squared Error - Increase Constant 0.05, Decrease Factor 0.1**



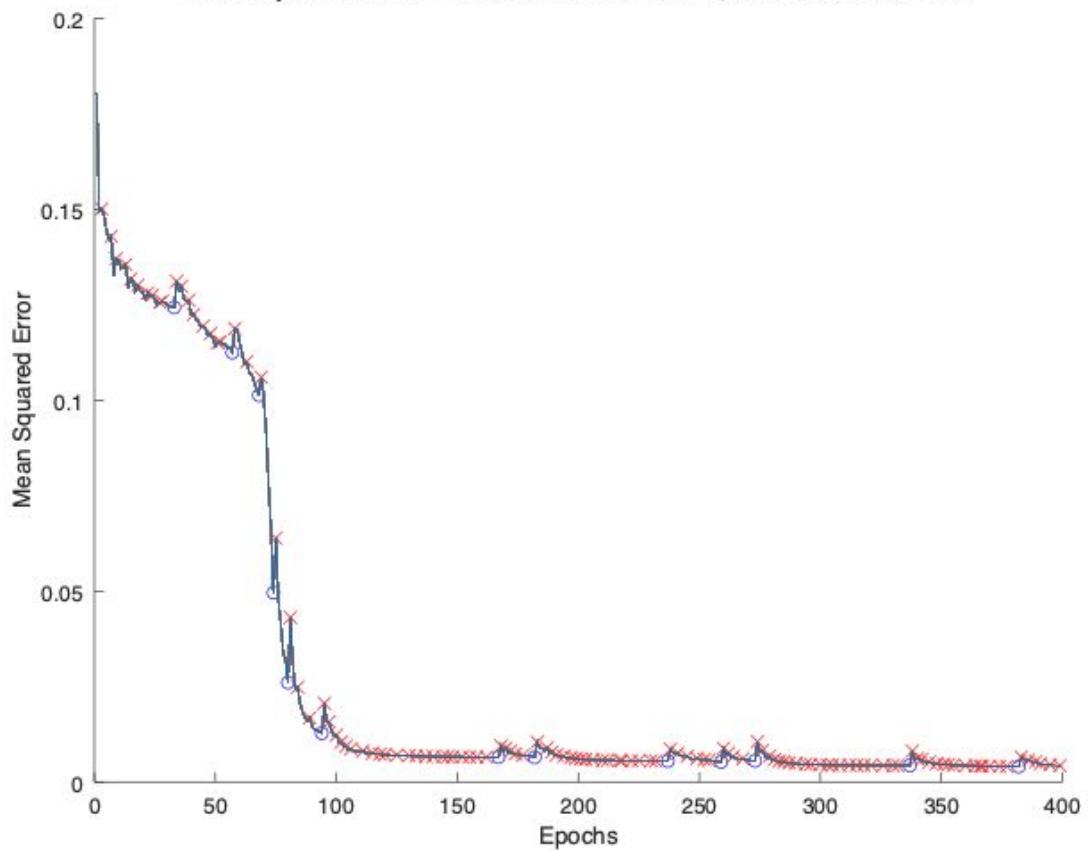
**Eta - Increase Constant 0.05, Decrease Factor 0.1**

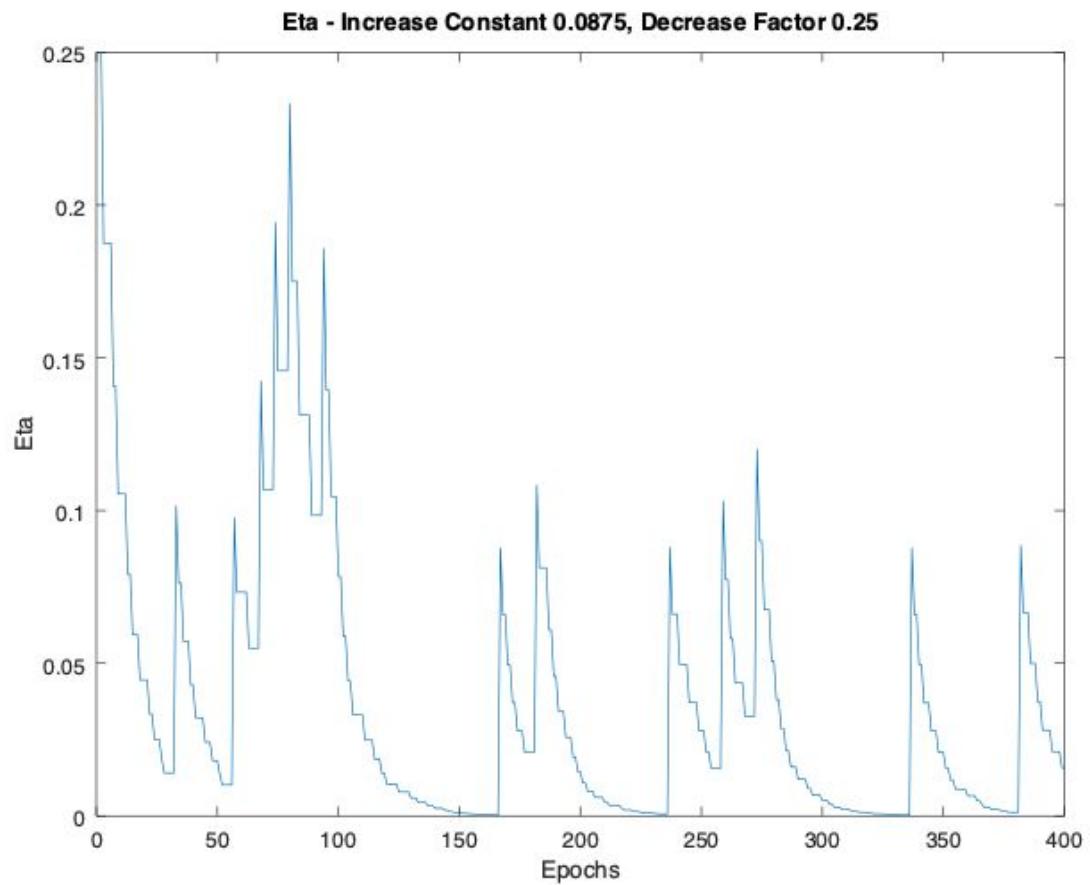


**Both terrain positions - Increase Constant 0.05, Decrease Factor 0.1**

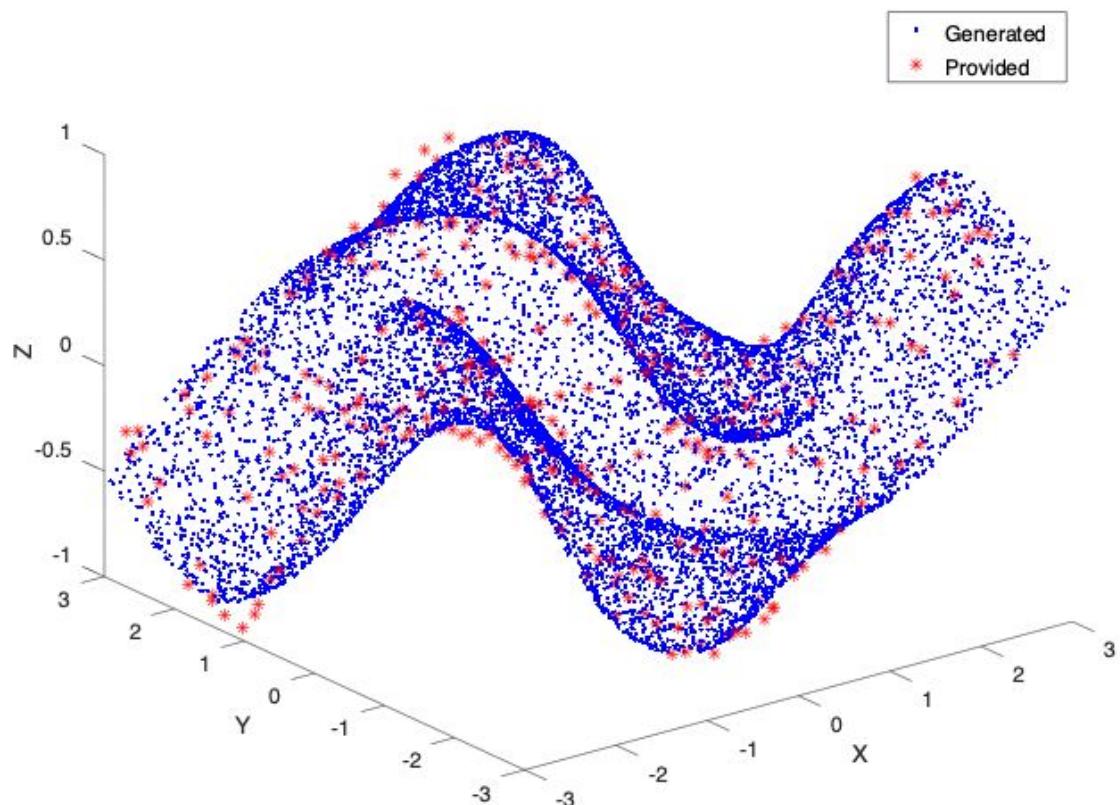


**Mean Squared Error - Increase Constant 0.0875, Decrease Factor 0.25**



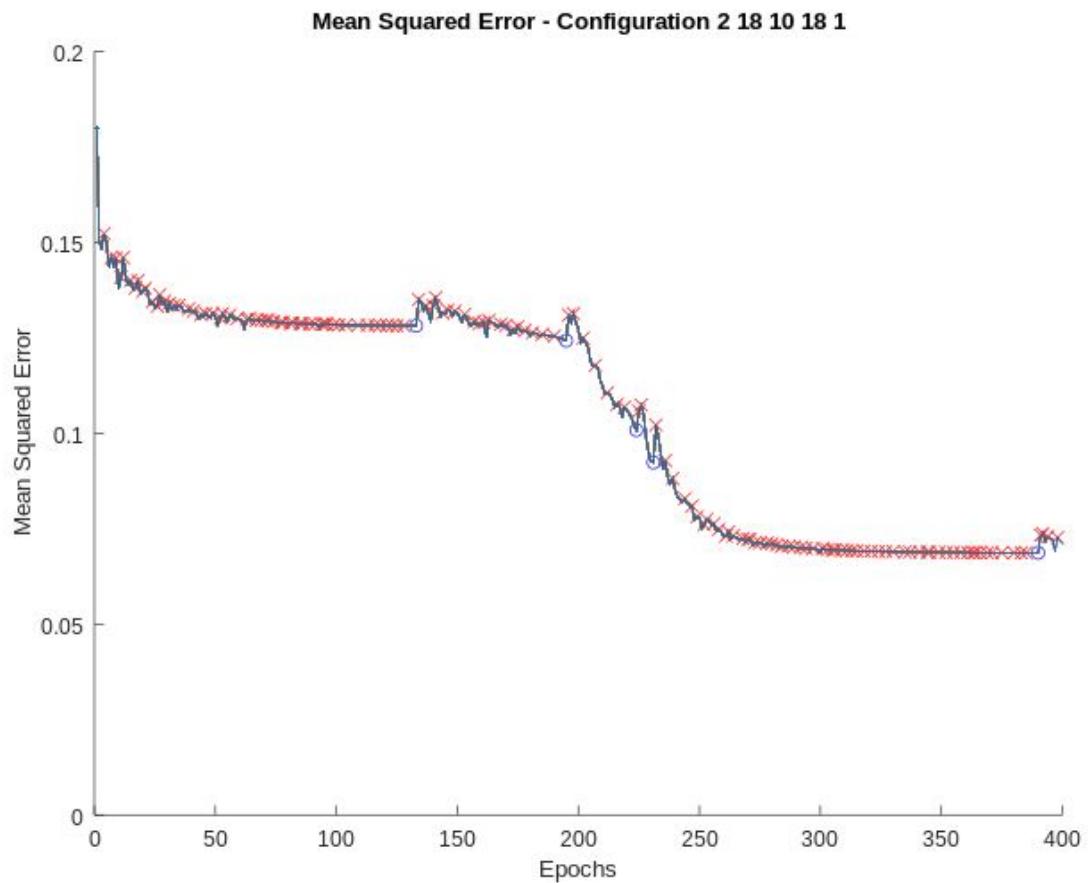


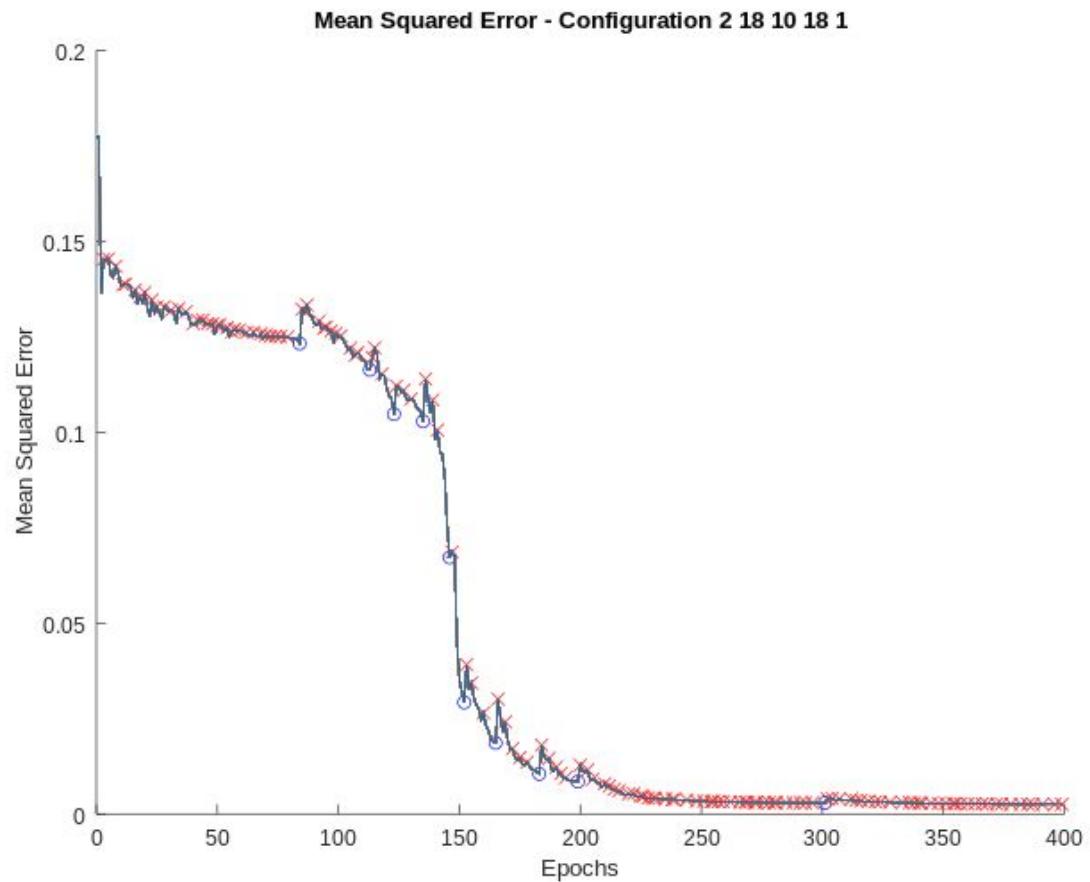
**Both terrain positions - Increase Constant 0.0875, Decrease Factor 0.25**



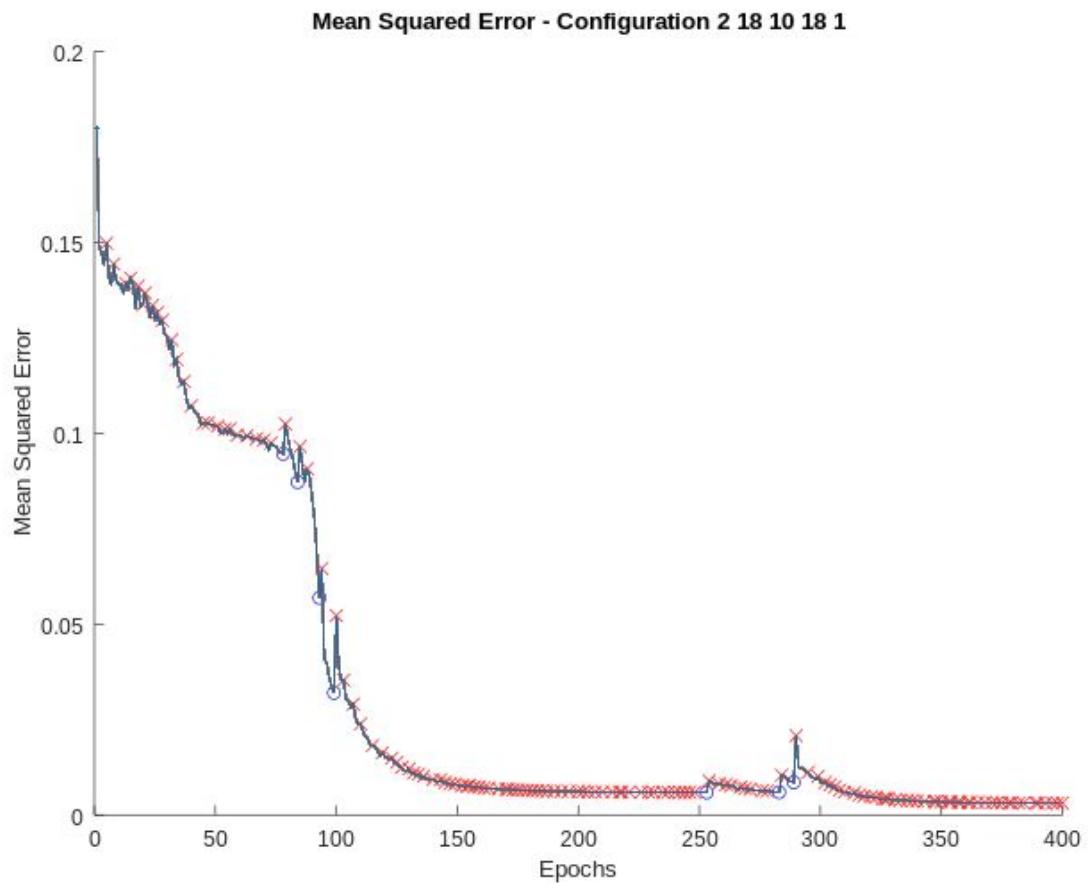
## Eta Adaptativo con Memoria vs. Sin Memoria

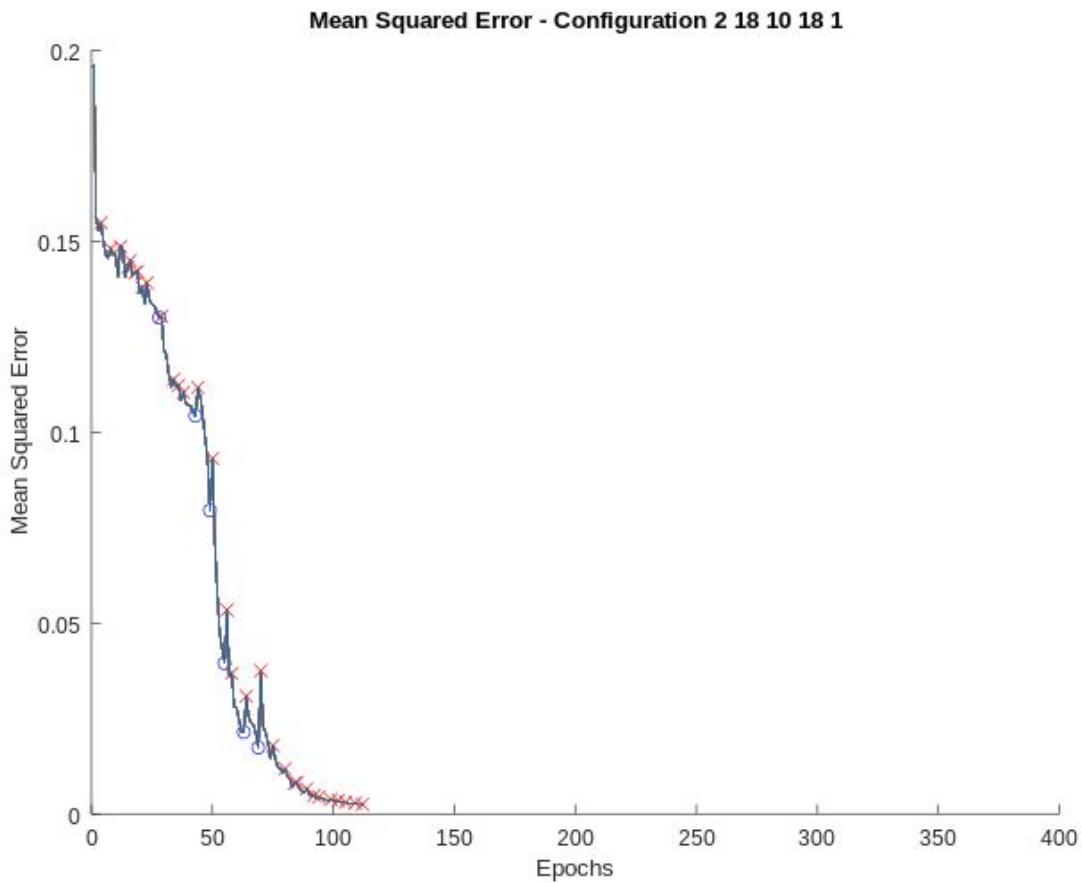
Memoria Activada





## Memoria Desactivada



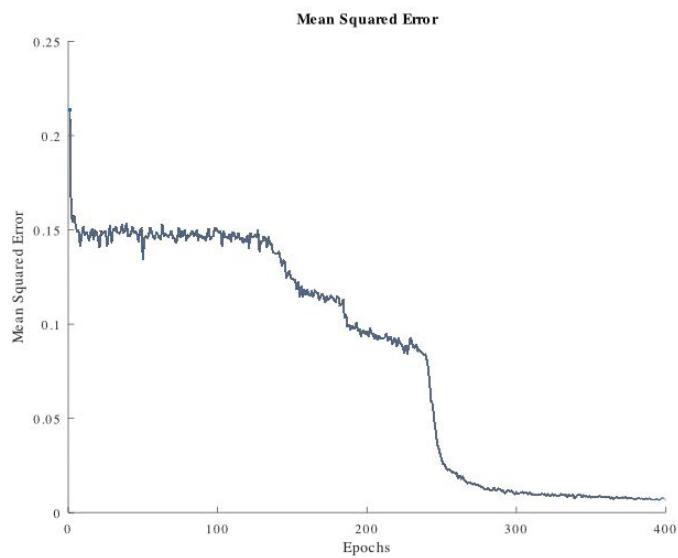


Se observa que con la memoria desactivada el decrecimiento inicial es mayor, aunque en general esto no implica que el error termine siendo menor que con la memoria activada pasado el límite de épocas.

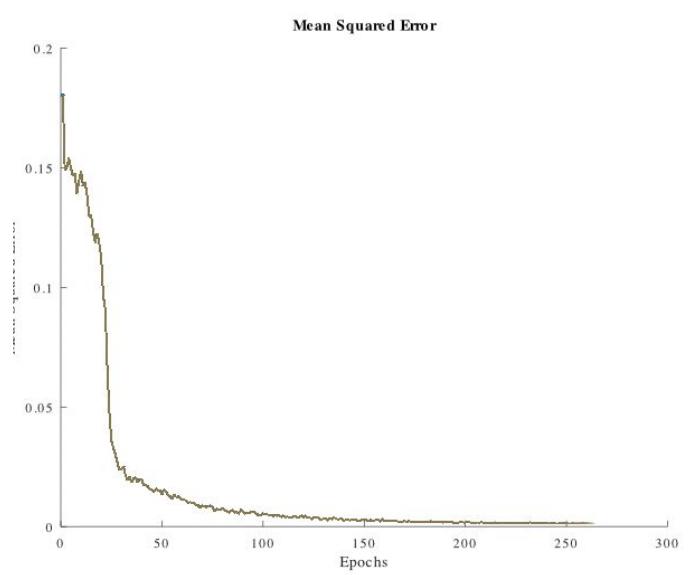
## Comparación de funciones de activación y factor beta

Función exponencial:

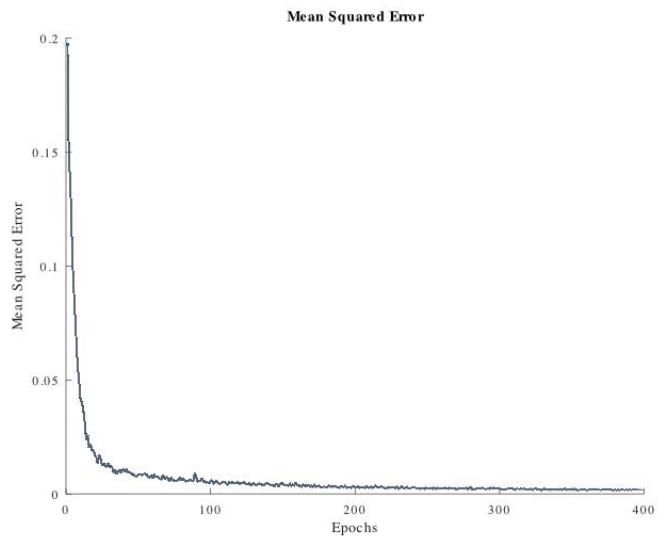
Beta	Épocas	Error
0.25	400	0.137400
0.5	400	0.006709
1	264	0.001237
2	400	0.002074



**Beta = 0.5**



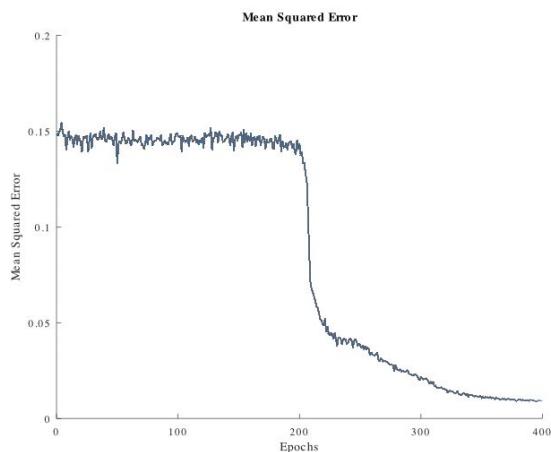
**Beta = 1**



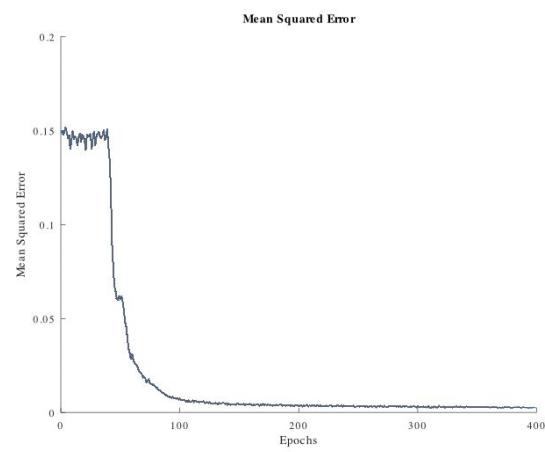
**Beta = 2**

Función tangente hiperbólica:

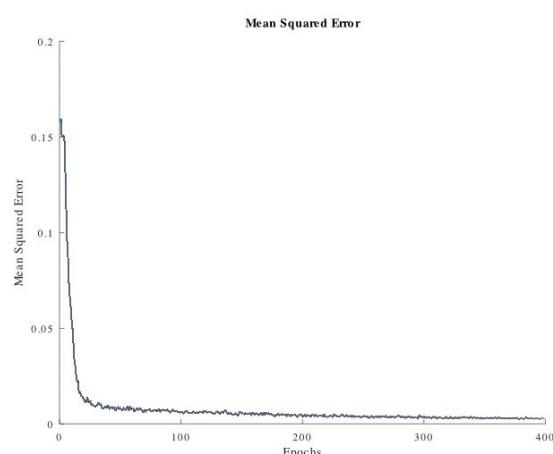
Beta	Épocas	Error
0.25	400	0.009320
0.5	400	0.002442
1	400	0.002837
2	-	-



Beta = 0.25



Beta = 0.5



Beta = 1

70 de 90

## Determinación de Método de Inicialización Óptimo de Pesos

	1er Corrida	2da Corrida	3er Corrida	4ta Corrida	5ta Corrida
Error con Inicialización random [-0.5 , 0.5]	0.001477749 70116330	0.001586120 6137081	0.001899601 71157504	0.002387545 6431415	0.001397163 806503224
Error con Inicialización con fan-in	0.001477749 70116335	0.001586120 6137082	0.001899601 71157502	0.002387545 6431416	0.001397163 806503228

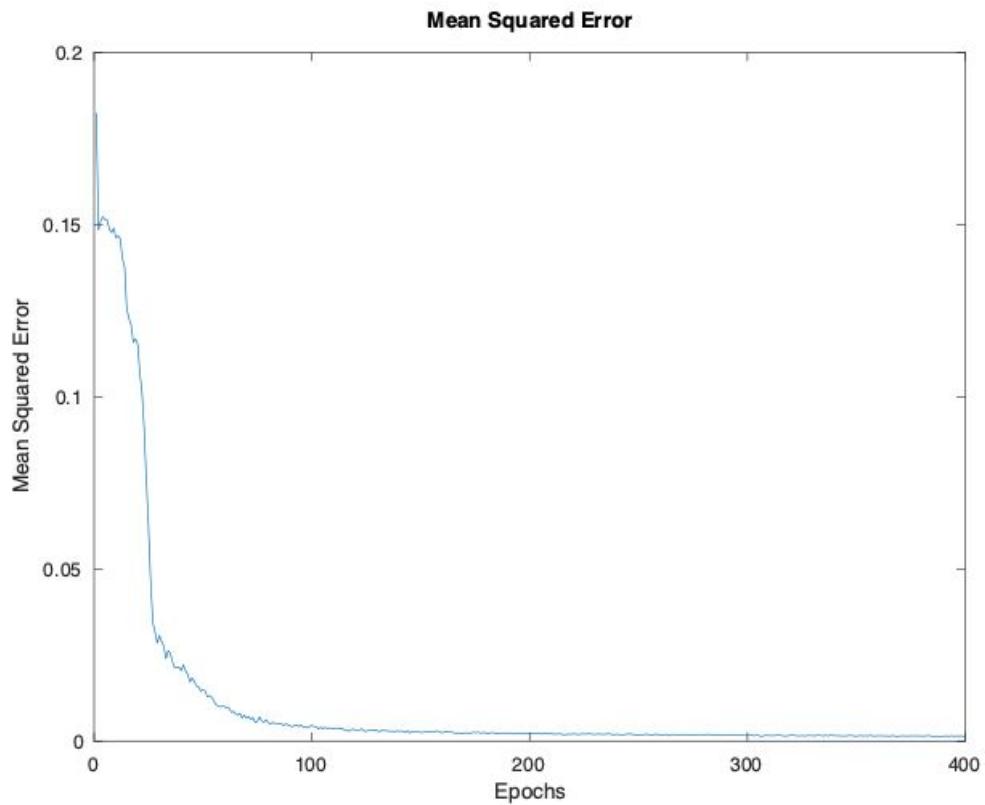
Como se puede ver en la tabla anterior se requiere de mucha precisión en el error para poder ver una diferencia entre ambos métodos.

En todas las corridas salvo en la tercera es mejor la inicialización con pesos random.

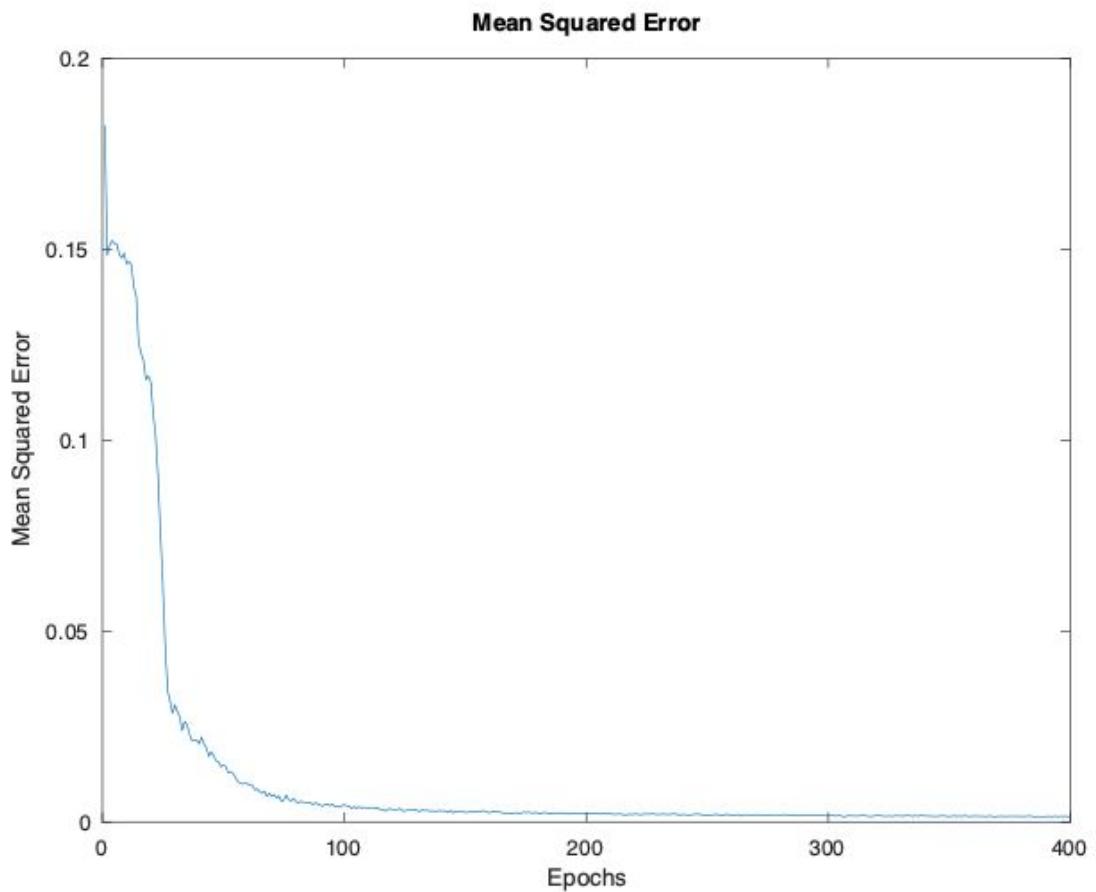
Veamos ahora los gráficos del error para ver si puede verse alguna diferencia considerable en la cantidad de épocas que tardan en llegar al error.

## 1er Corrida

Inicialización Random

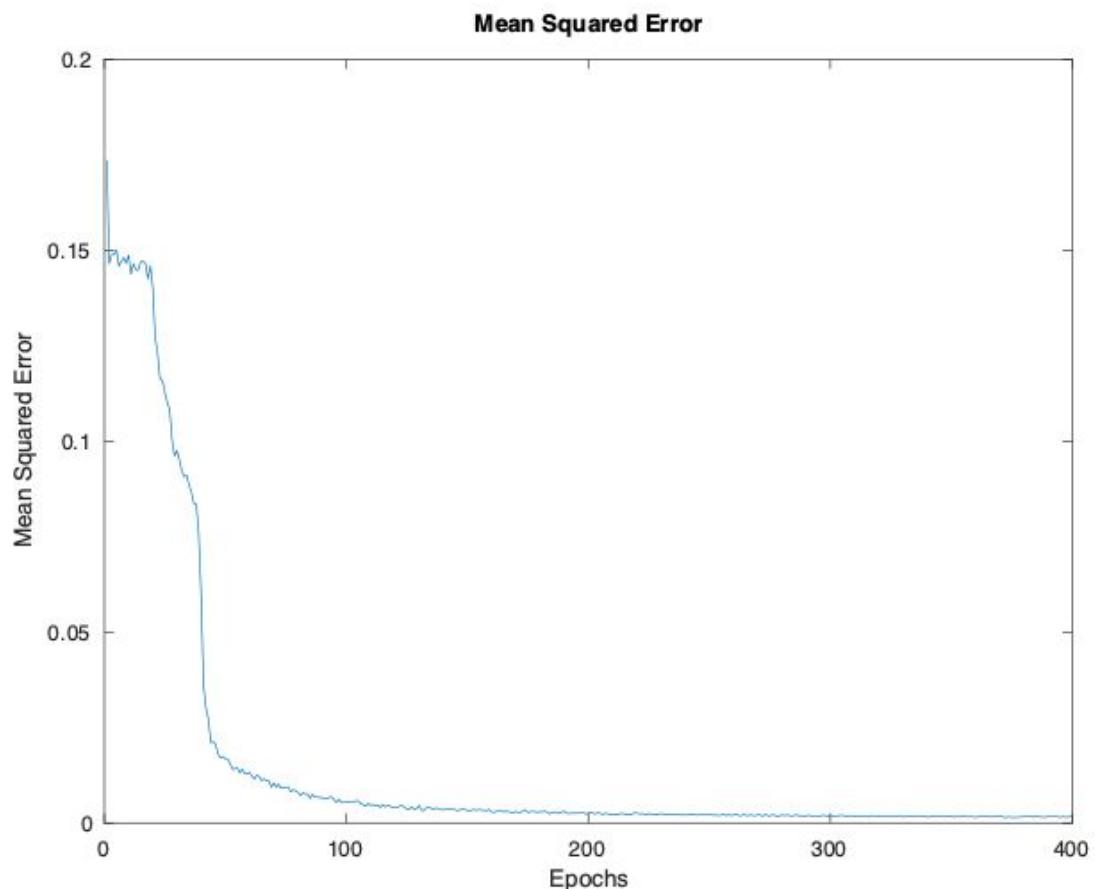


Inicialización fan-in

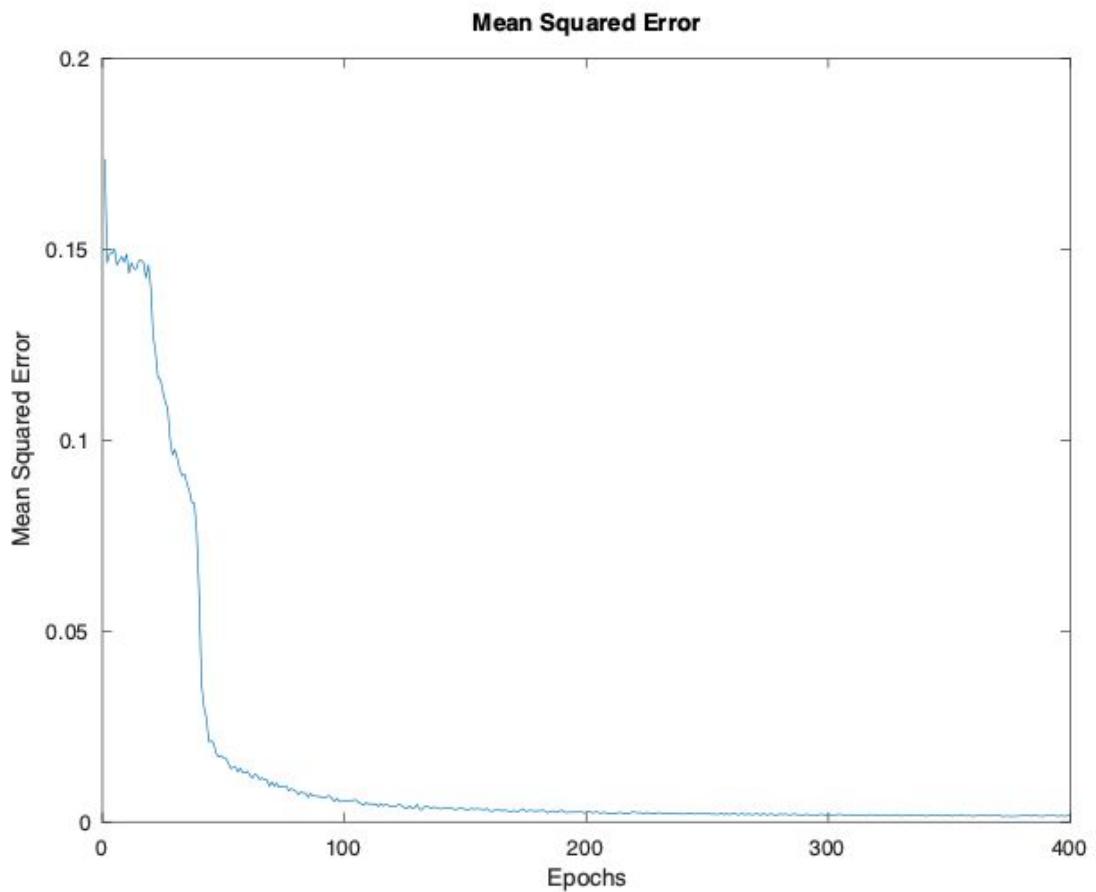


2ds Corrida

Inicialización Random

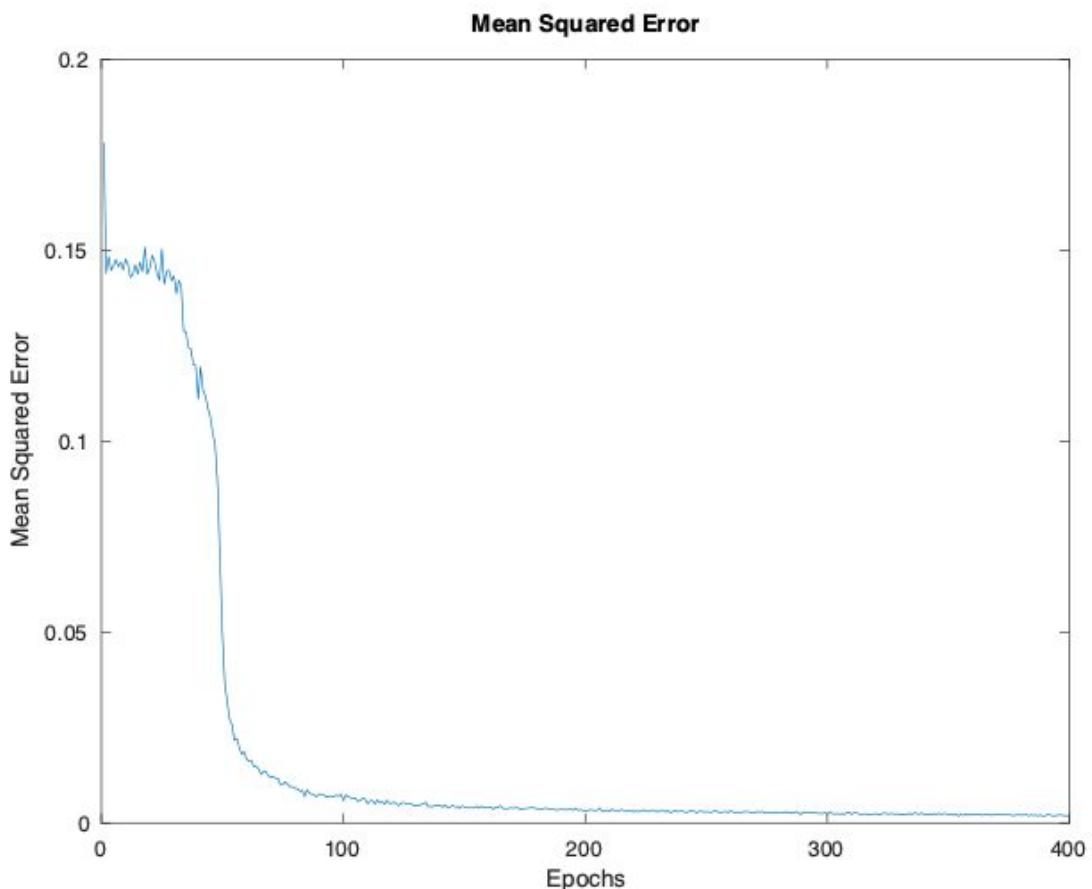


Inicialización fan-in

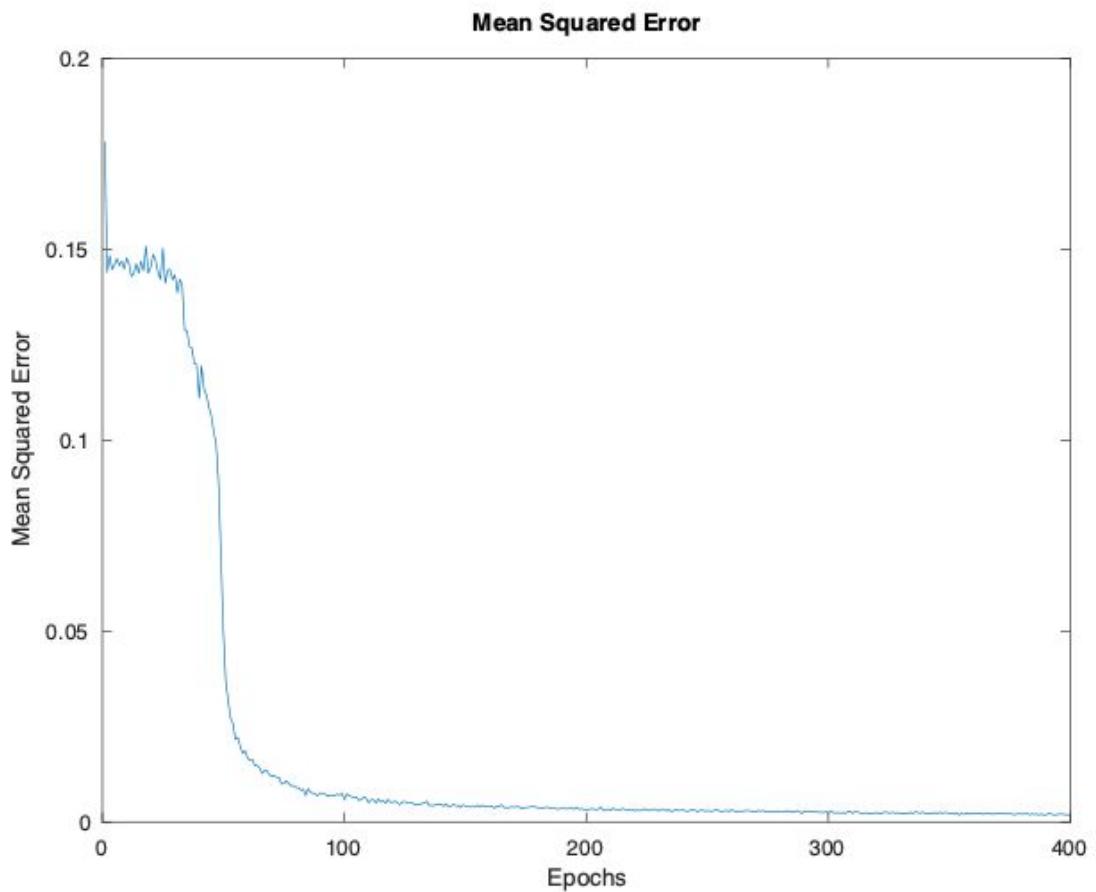


3er Corrida

Inicialización Random

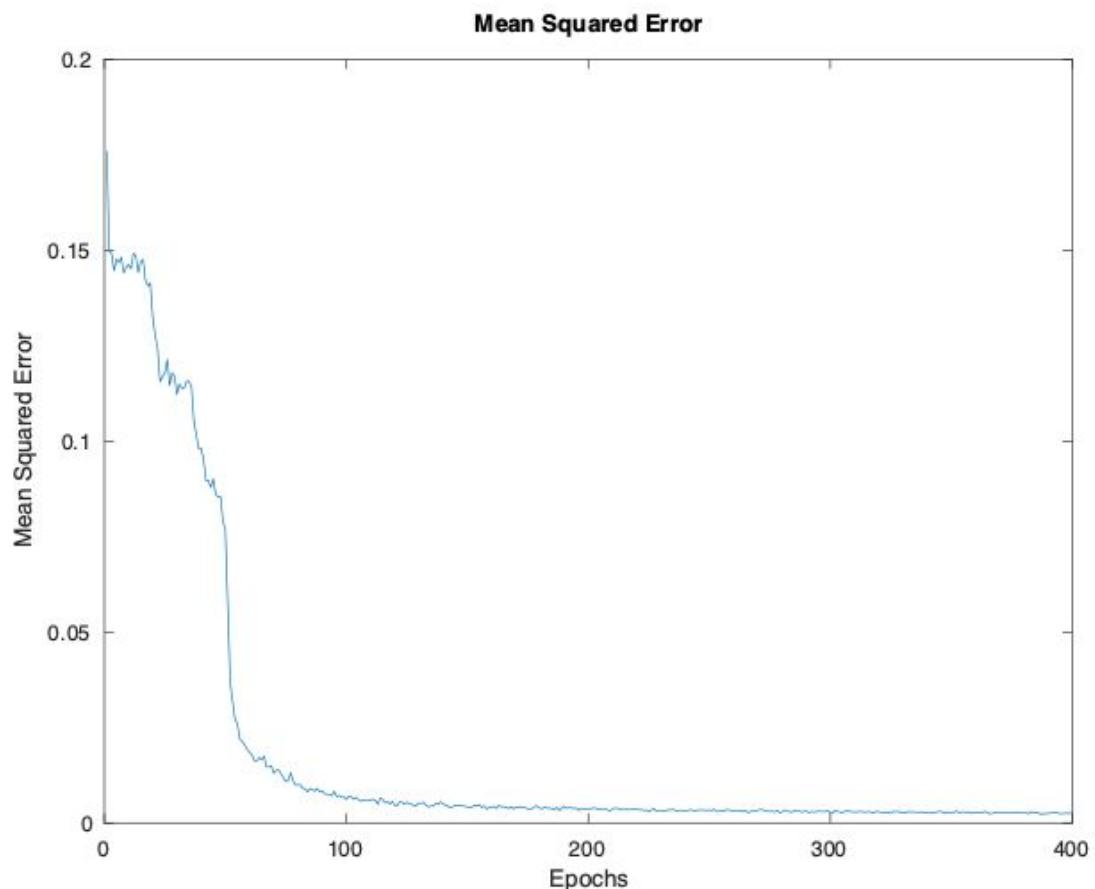


Inicialización fan-in

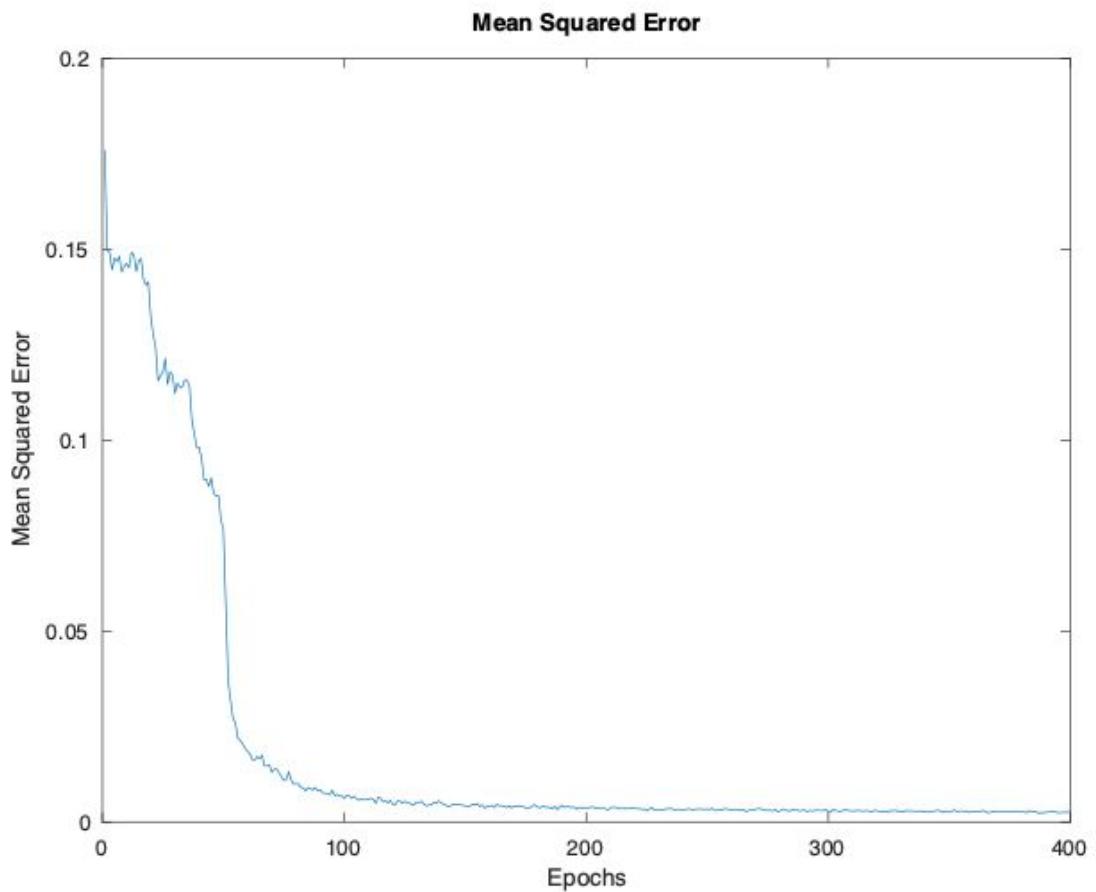


## 4ta Corrida

Inicialización Random

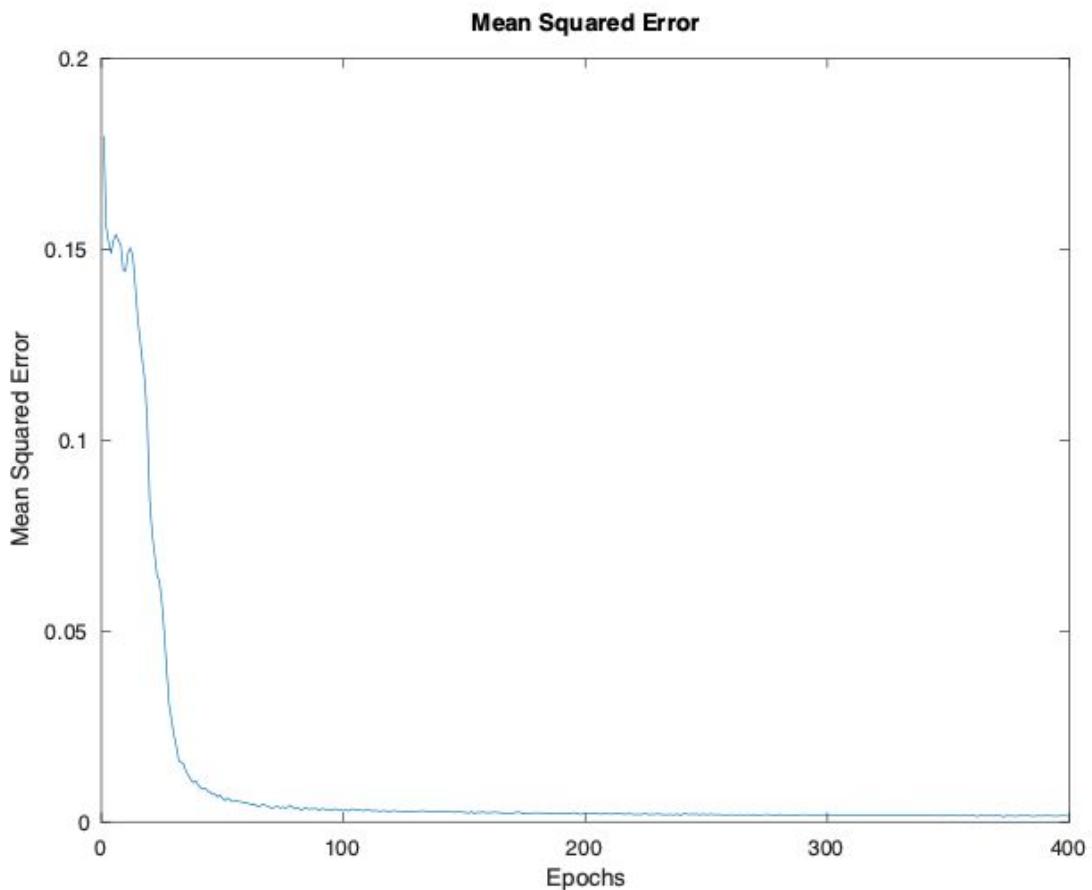


Inicialización fan-in

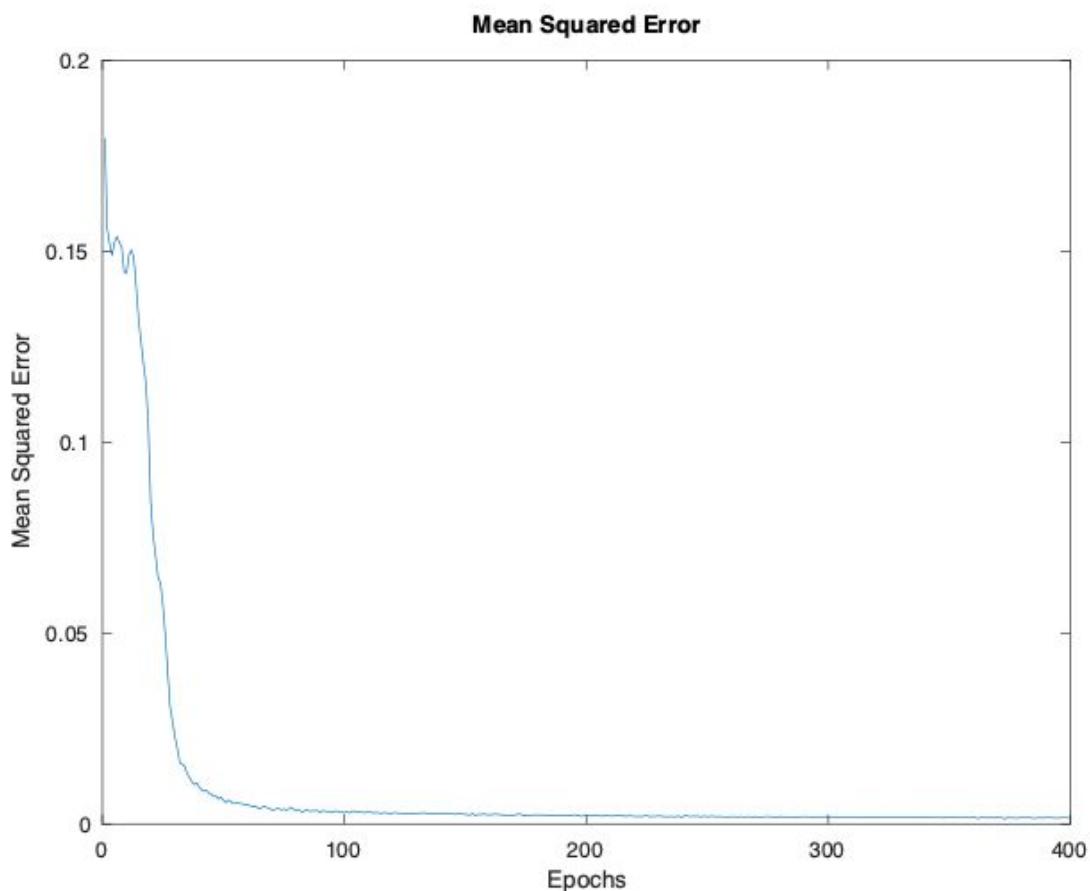


## 5ta Corrida

Inicialización Random



## Inicialización fan-in



Como se puede ver en los gráficos no hay ninguna diferencia apreciable en los gráficos y a pesar de que la diferencia es muy pequeña podemos concluir en base al error que la inicialización con pesos random es mejor que con fan-in.

## Determinación del Porcentaje de Entrenamiento

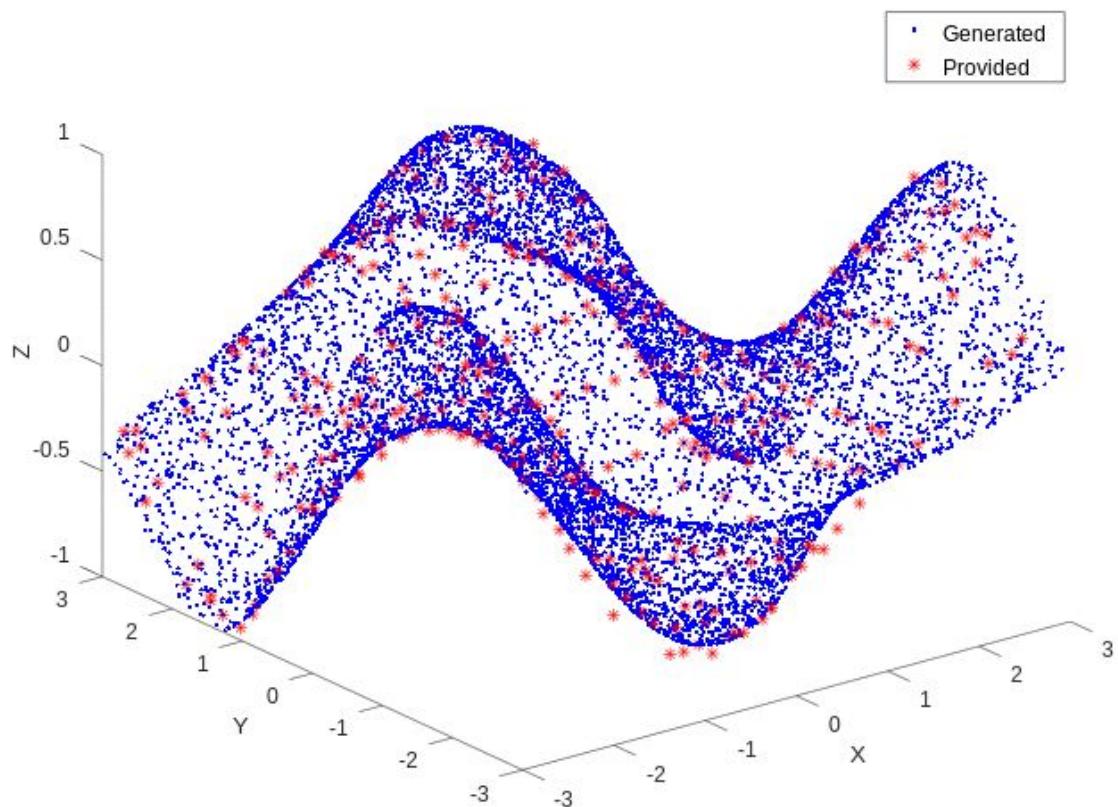
Porcentaje de Entrenamiento (%)	Error
10	0.070233
20	0.003167
30	0.001745
40	0.002406

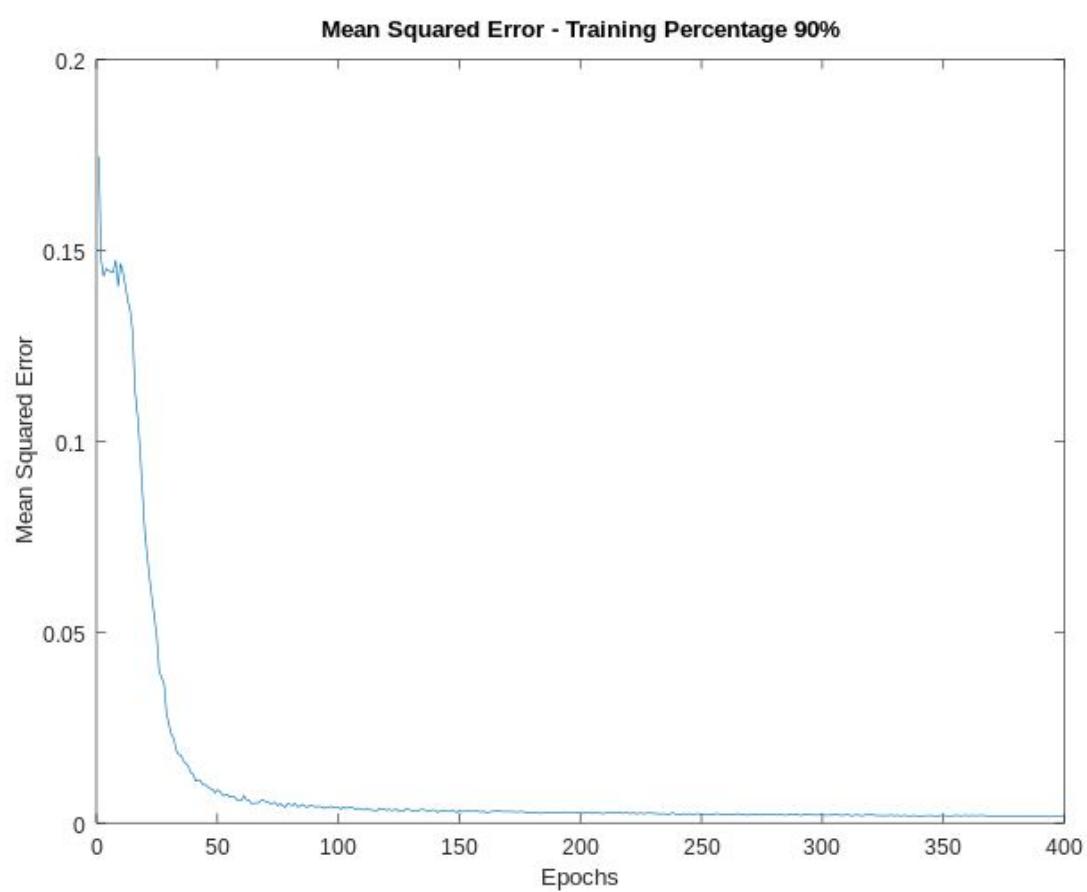
50	0.002066
60	0.001959
70	0.002308
80	0.001250
90	0.001642

Como se puede ver en la tabla, el mejor porcentaje de entrenamiento es el de 80% y el siguiente mejor es el de 90%

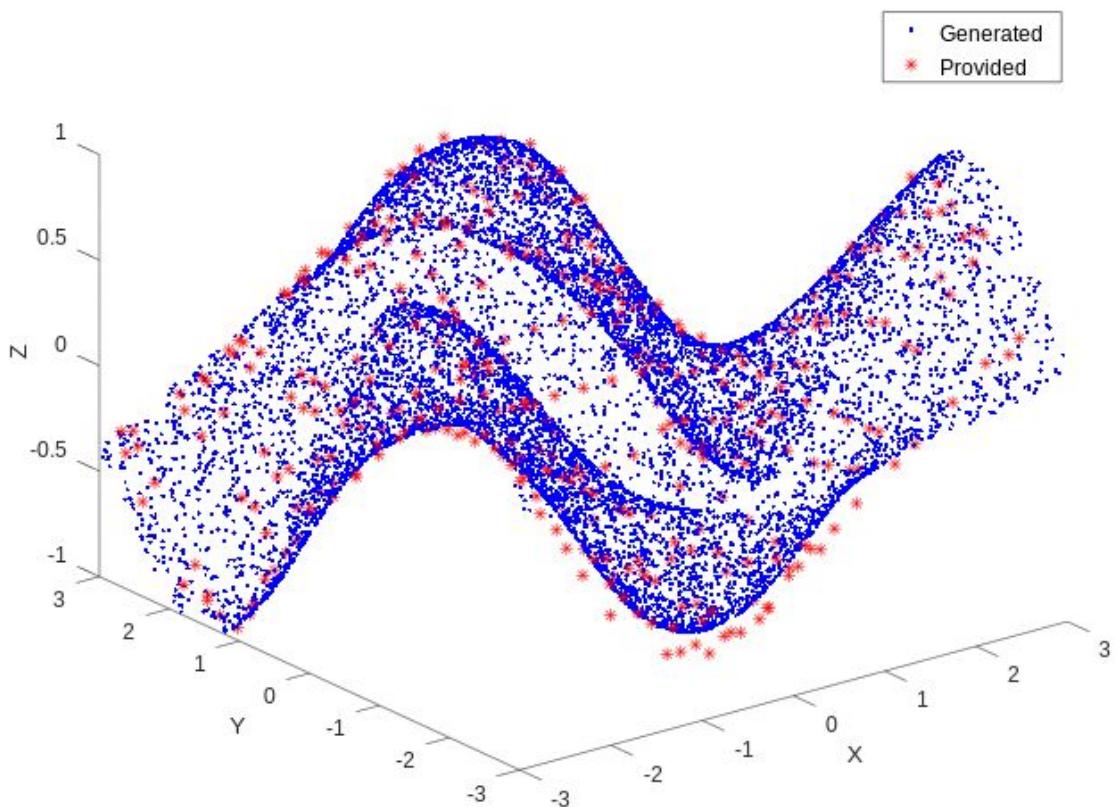
Veamos los gráficos de error y de la superficie que aprenden ambos

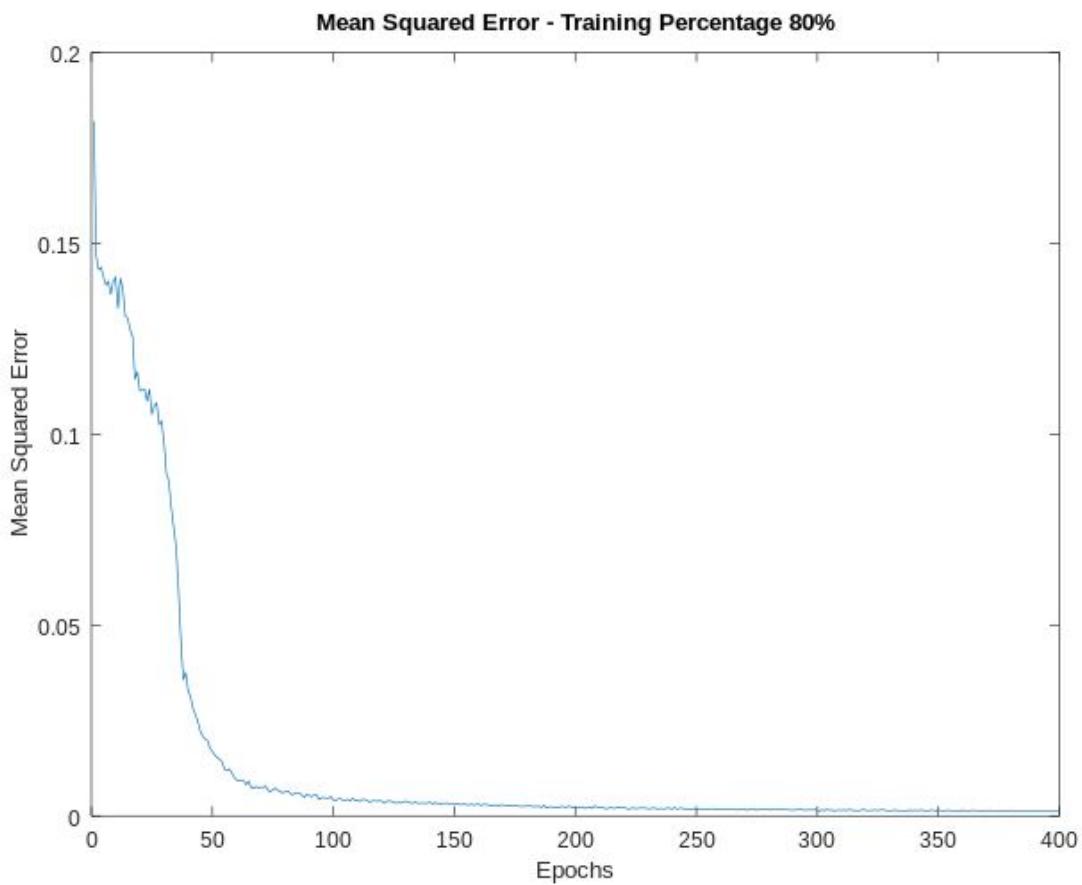
**Both terrain positions - Training Percentage 90%**





Both terrain positions - Training Percentage 80%





Como sucedió con la tabla los gráficos son muy similares, para obtener el mejor variaremos el porcentaje entre el 80 y el 90 con un paso de 1

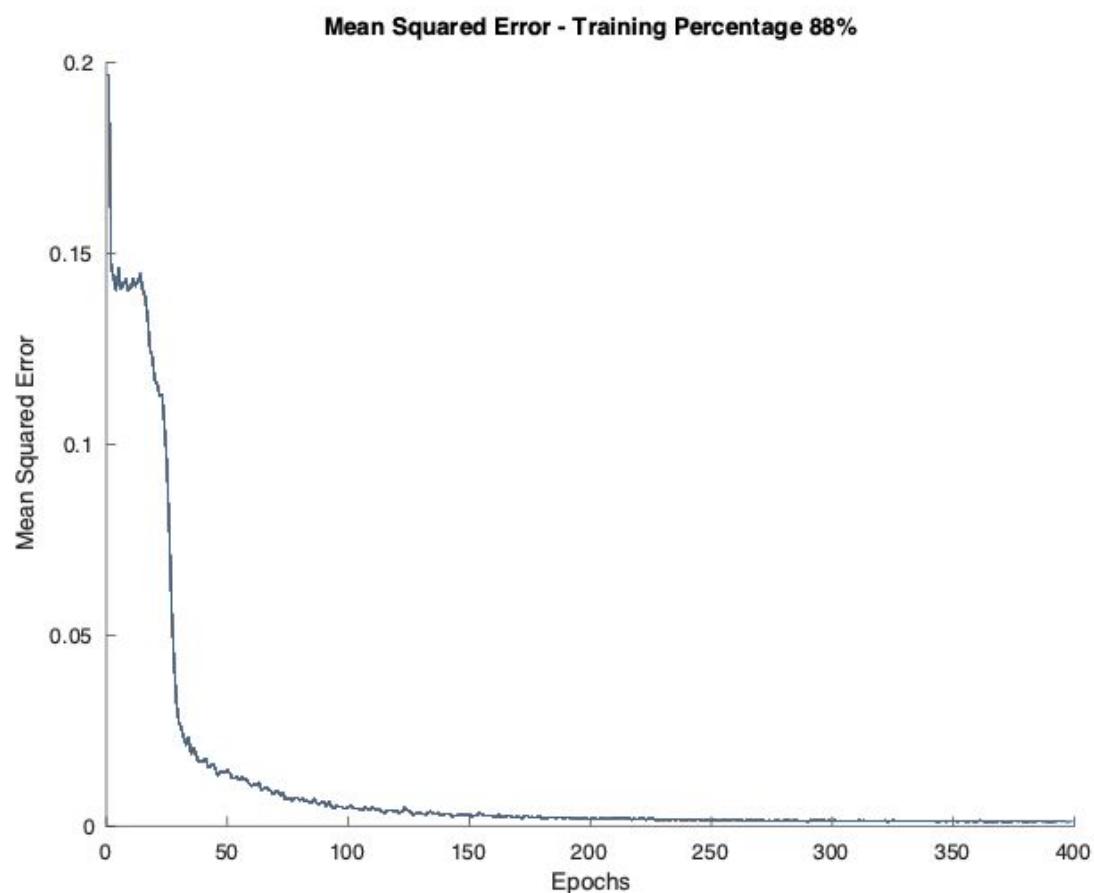
Porcentaje de Entrenamiento (%)	Error
80	0.001250
81	0.001238
82	0.001309
83	0.001427
84	0.001182
85	0.001539
86	0.001324
87	0.001345

88	0.001114
89	0.001129

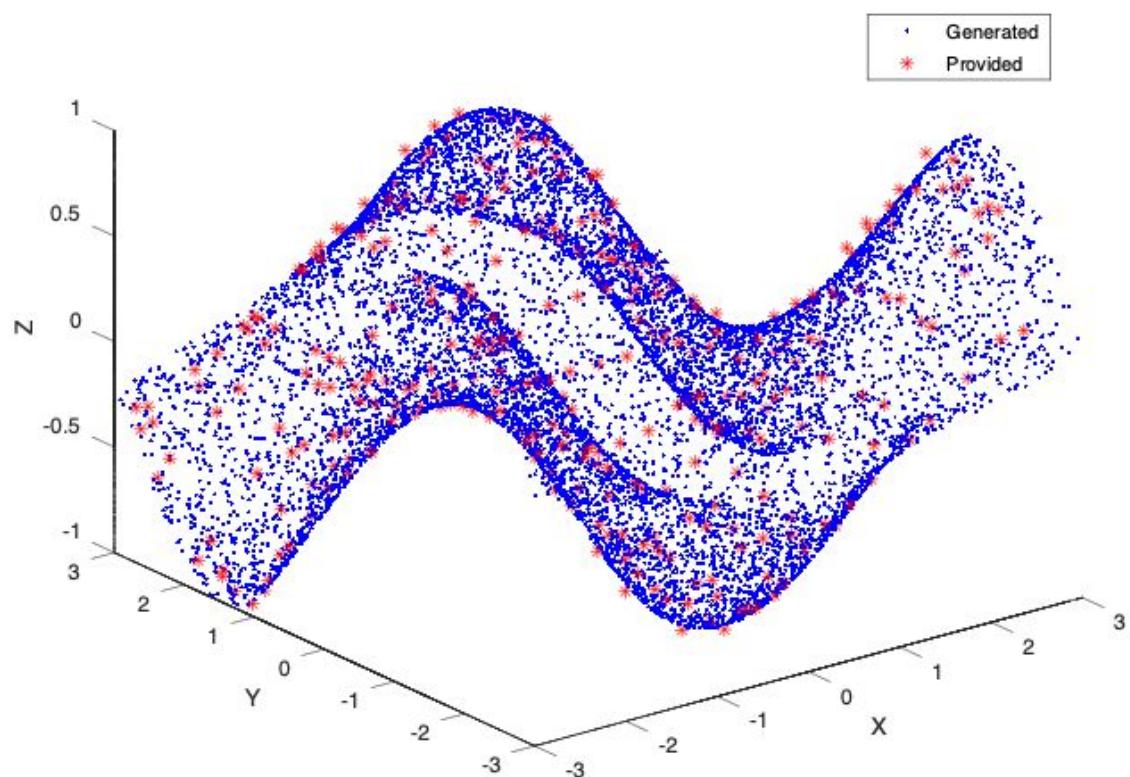
Como se puede ver en la tabla, el mejor porcentaje de entrenamiento para reducir el error es de 88% seguido de 89%

Analicemos los gráficos de error y de la superficie en ambos casos.

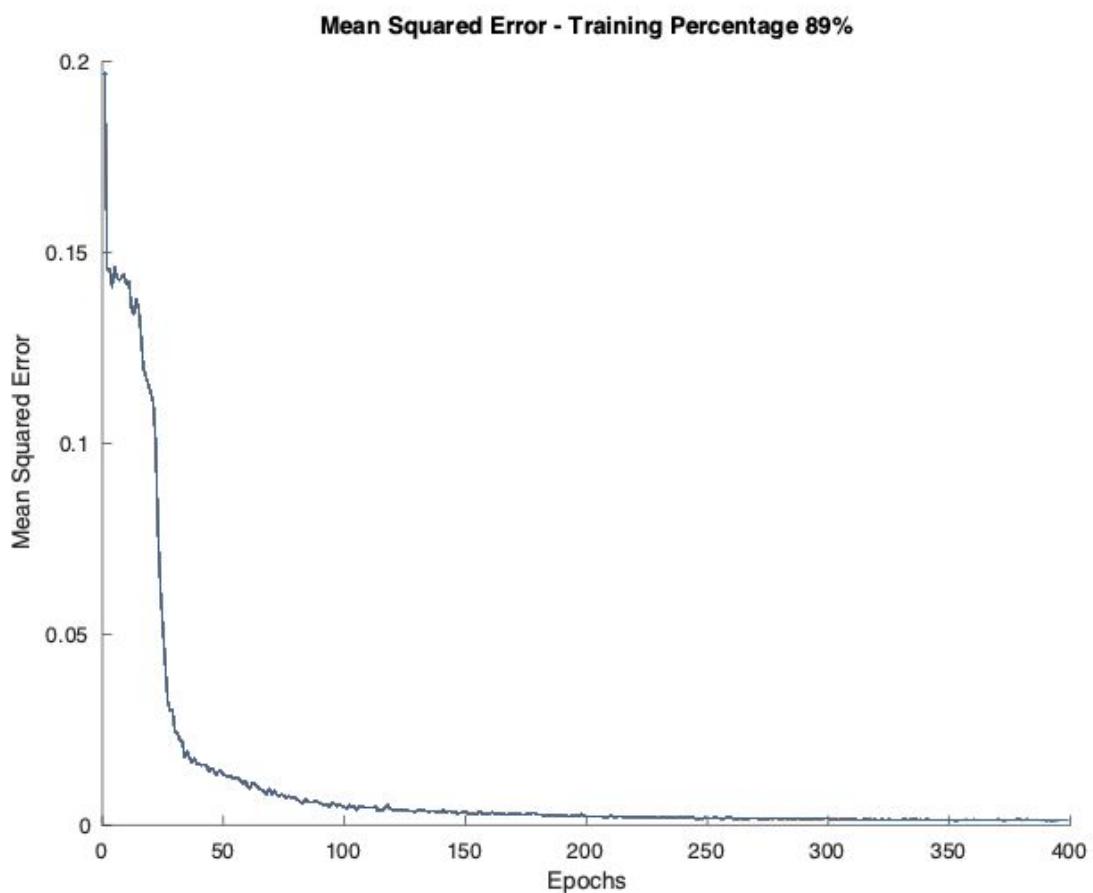
Porcentaje de Entrenamiento = 88%



**Both terrain positions - Training Percentage 88%**



Porcentaje de Entrenamiento = 89%



**Both terrain positions - Training Percentage 89%**

