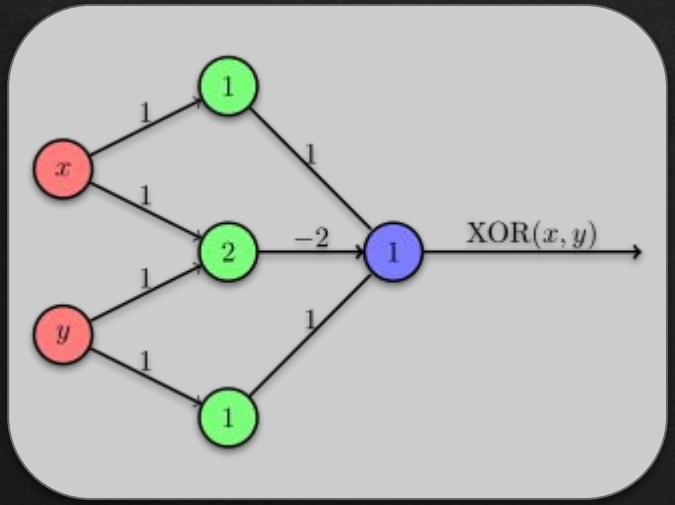
Redes Neuronales

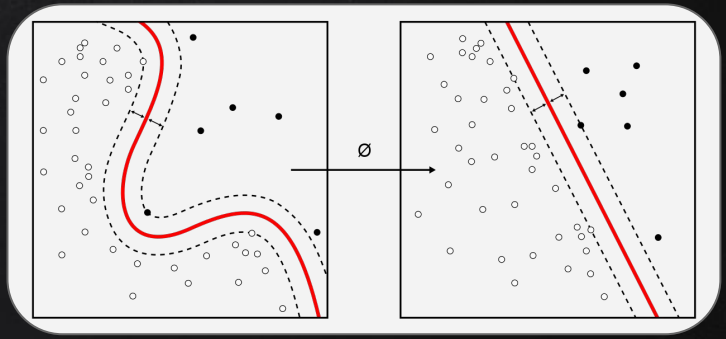
Grethe - Nagelberg - Grabina


Problema
ATT
99
DÉF
∞
Lil Uzi Vert
007
G×01
/ 021

Solución



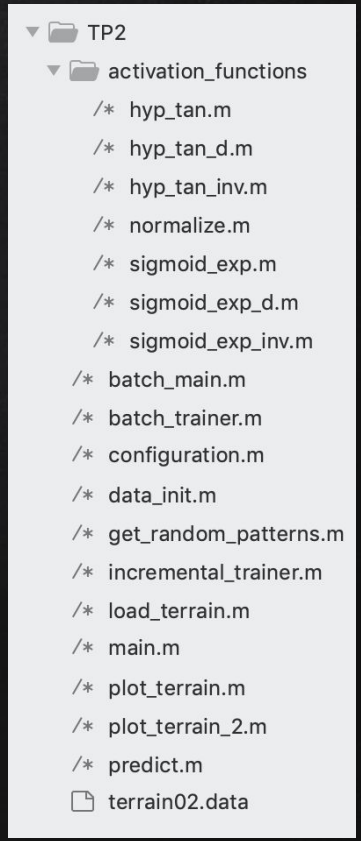
Entrada: 2 – Salida: 1
Entrenamiento supervisado



Implementacion

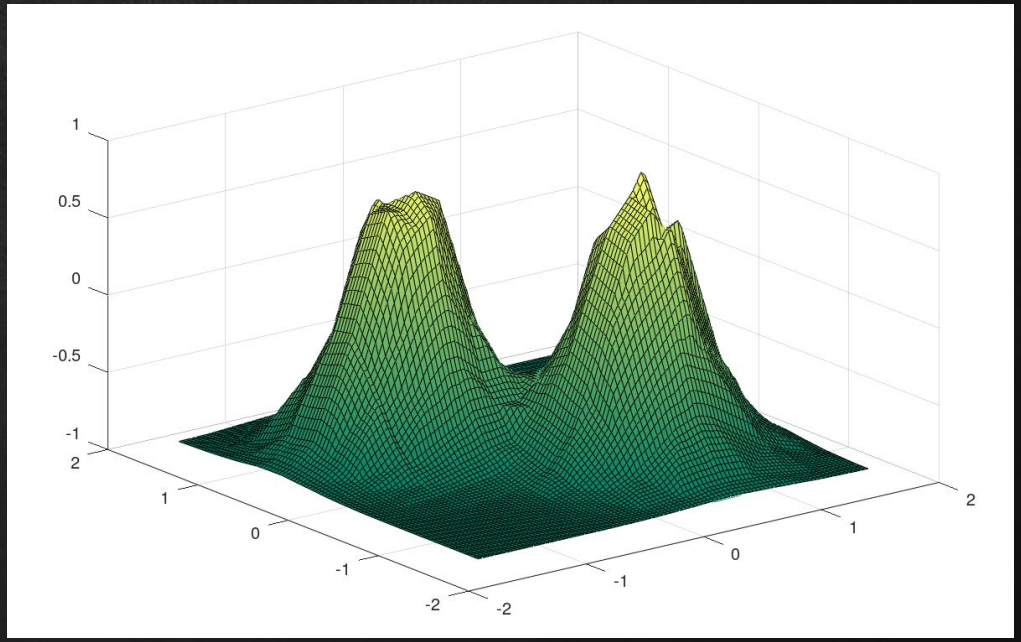
- ✘ Octave
- ✘ Git
- ✘ Teóricas de la cátedra

Perceptron simple (AND) -> Perceptron multicapa (XOR) -> Simulación del terreno

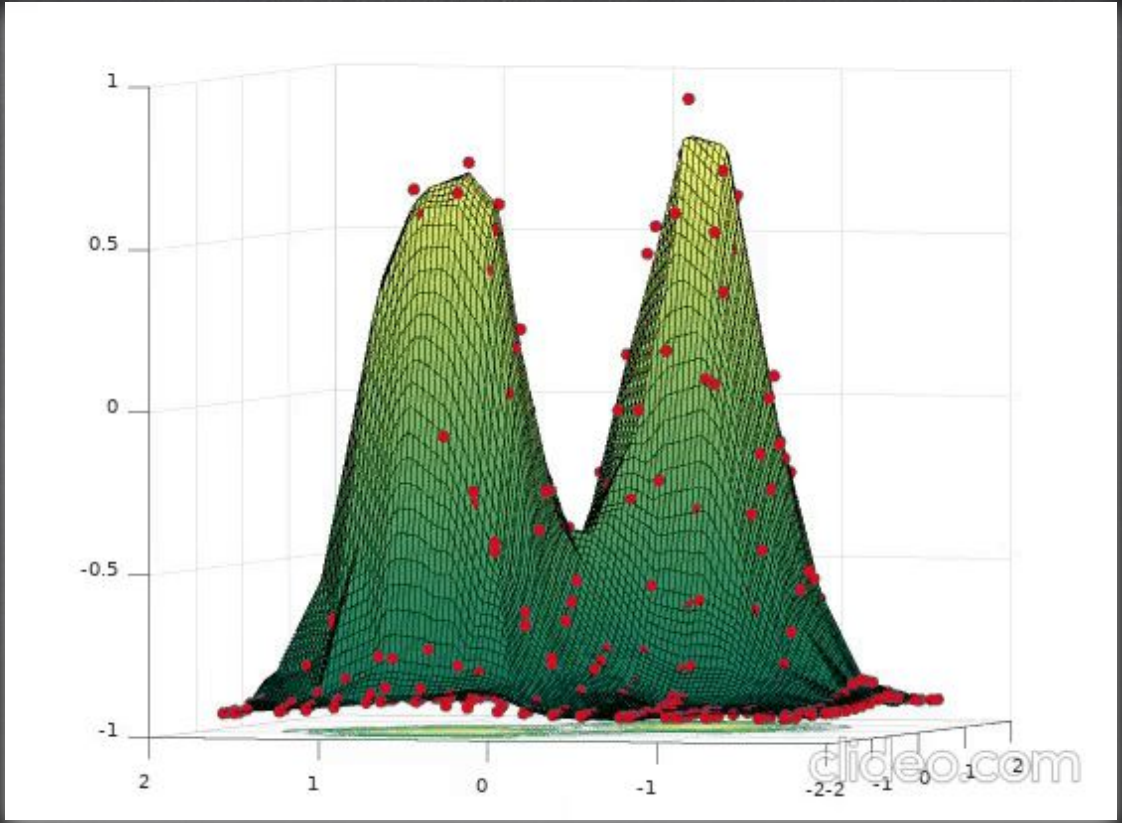


Terreno real

Mordor



Terreno Generado



60% datos
Arq. 50–50–50–50
E < 0.0002
7000 Epochs

¿Como lo hicimos?

Entrenamos la red con

Batch

Error de corte: 0.0002 (o largo plazo)

Función Tangencial

Beta = 1.0

ETA = 0.0005

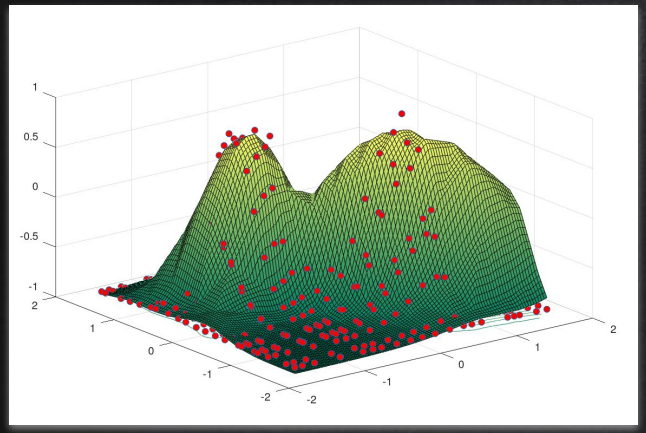
Puntos específicos añadidos para precisión (los mas elevados)

Sin optimizaciones (ETA adaptativo, Momentum)

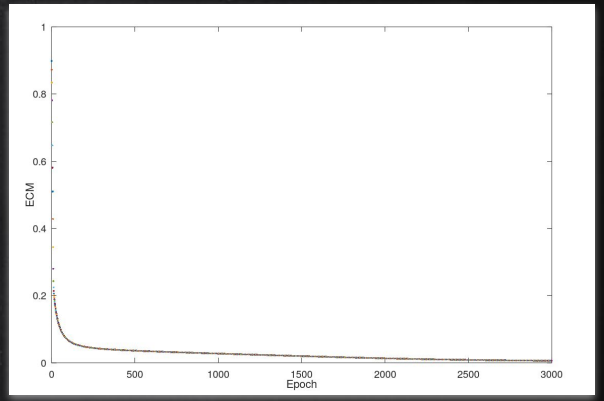
Arquitectura (50 50 50 50)

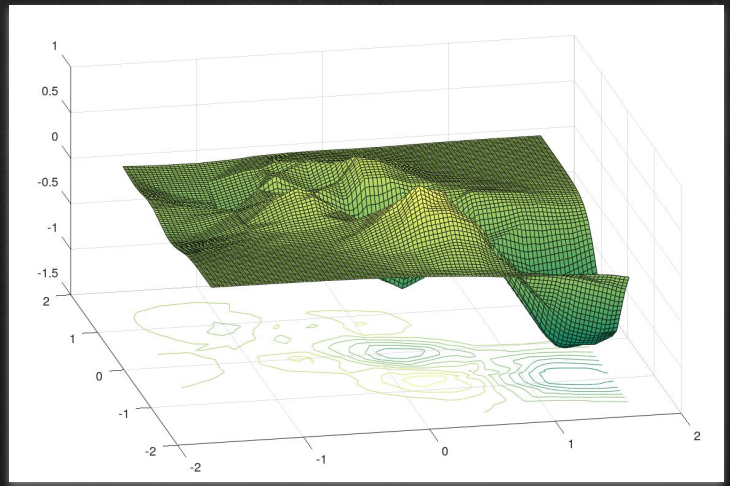
Resultados de las variaciones de parámetros

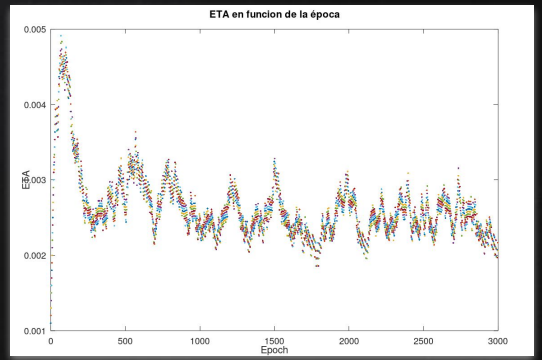
METRICAS



ECM, EPOCHS, GENERALIZACION







Parametros fijos

Batch

Error de corte: 0.0005

Función Tangencial

Beta = 1.0

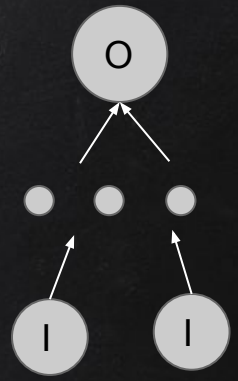
ETA = 0.0005

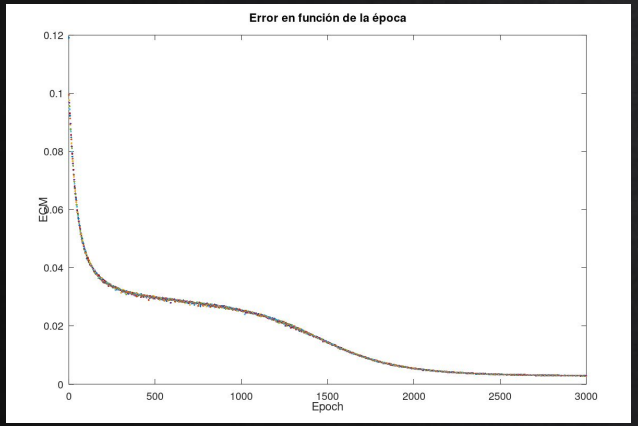
Sin puntos específicos añadidos para precisión

Sin optimizaciones (ETA adaptativo, Momentum)

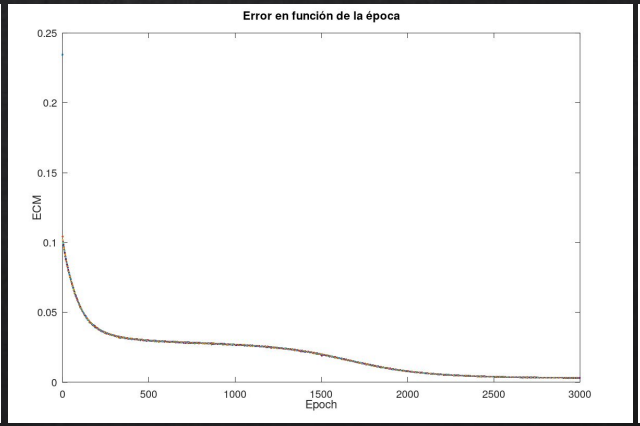
Arquitectura (50 50 50 50)

Comparativa: ARQUITECTURA Nodos ~ 1 CAPA

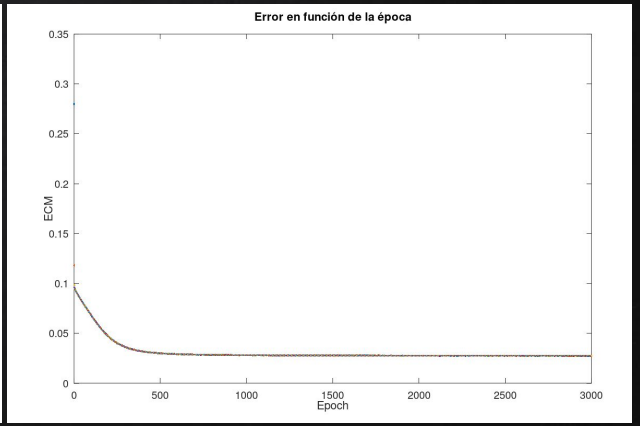




200 neuronas

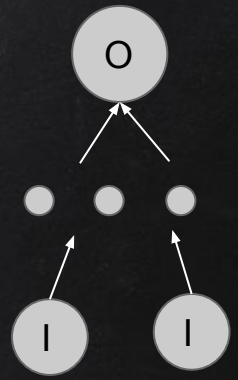


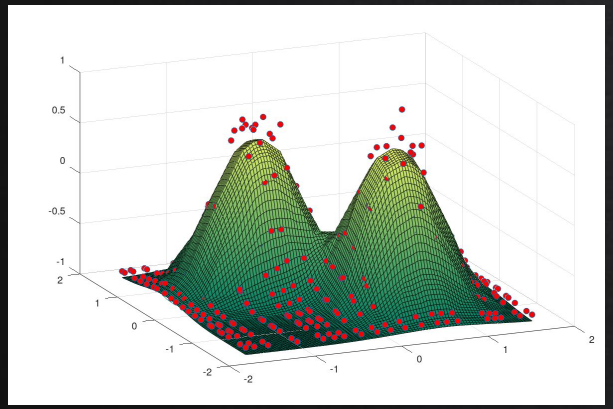
120 neuronas



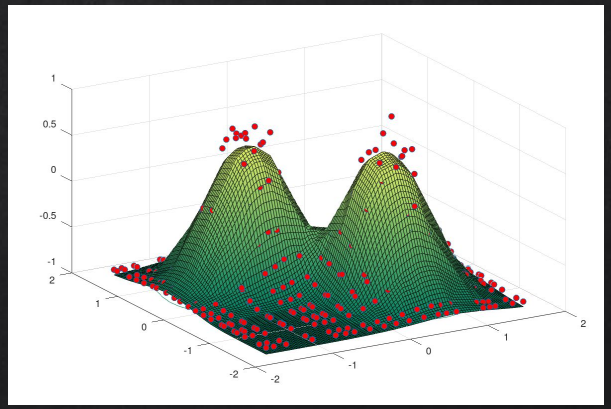
60 neuronas

Comparativa: ARQUITECTURA Nodos ~ 1 CAPA

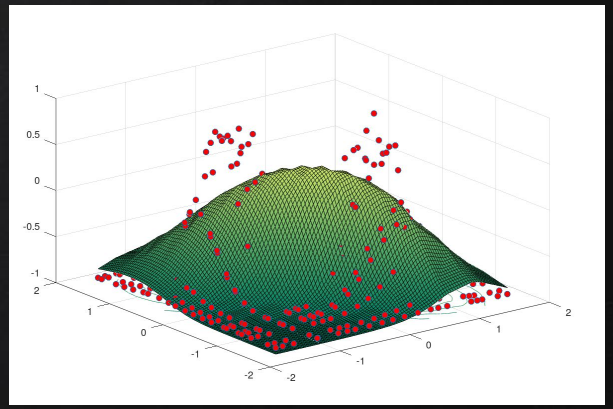




200 neuronas

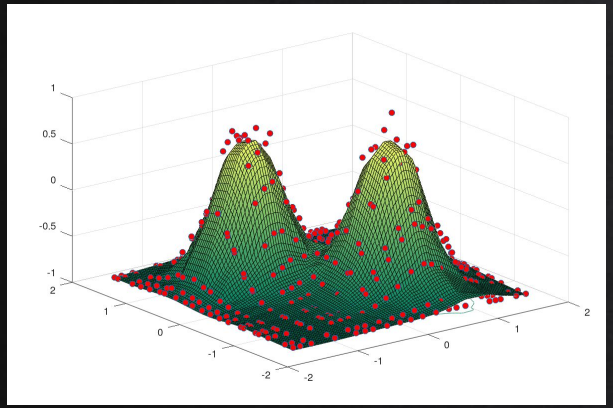


120 neuronas

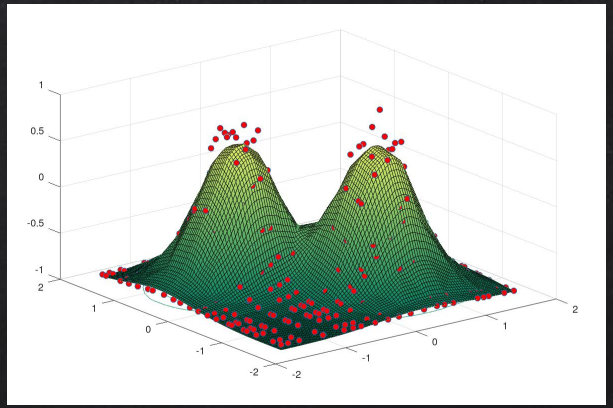


60 neuronas

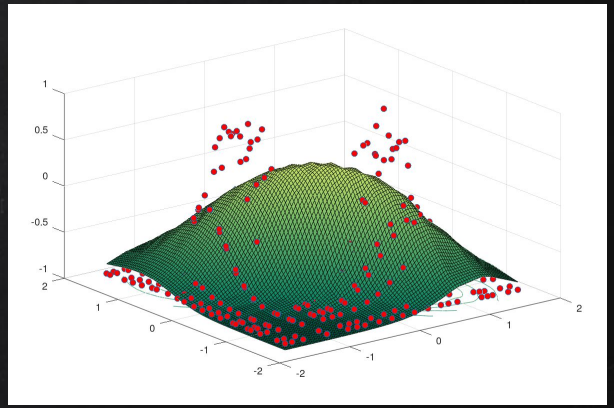
Comparativa: ARQUITECTURA Capas Ocultas 3000 EPOCH



[50 50 50 50 50]
5 capas



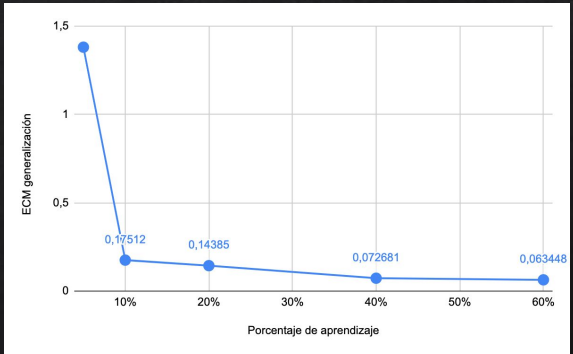
[50 50 50]
3 capas



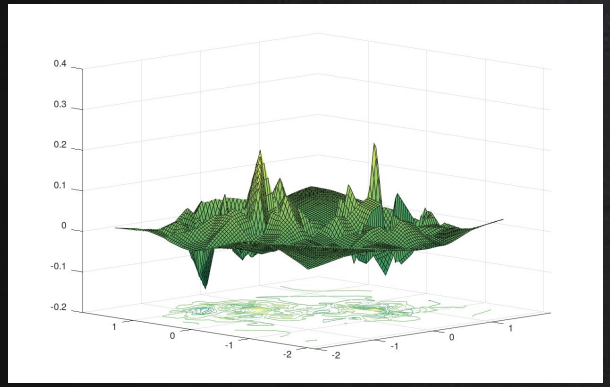
[50]
1 capa

Comparativa:
% de datos aprendidos

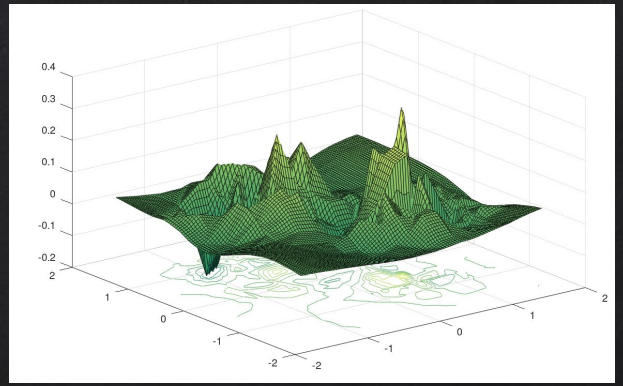
Porcentaje de aprendizaje	ECM generalización
5%	1.3813
10%	0.17512
20%	0.14385
40%	0.072681
60%	0.063448



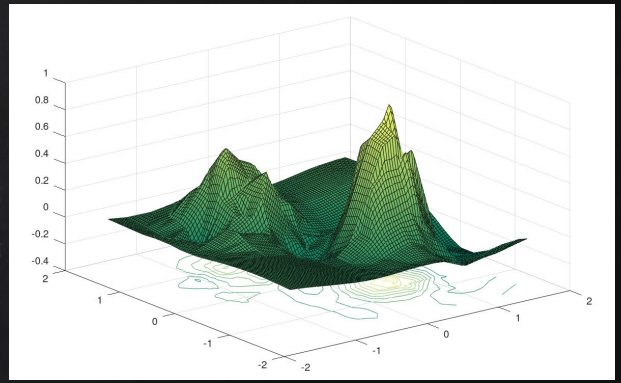
Comparativa:
% de datos aprendidos
Error Tridimensional



60%



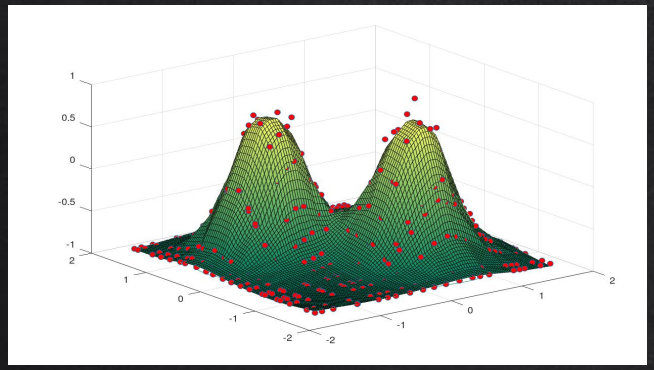
20%



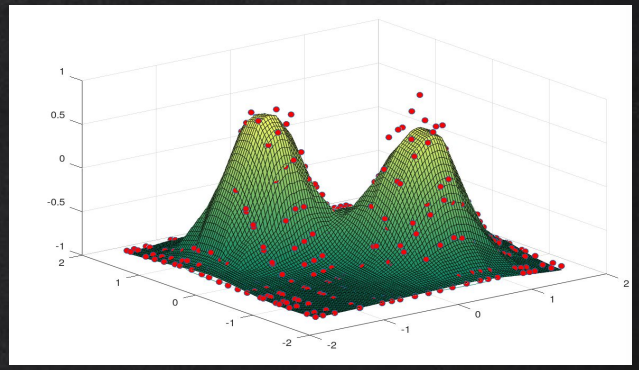
5%

Comparativa:
% de datos aprendidos: generación

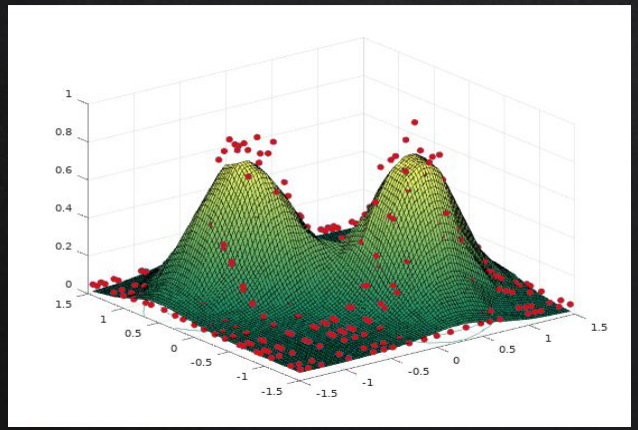
60%



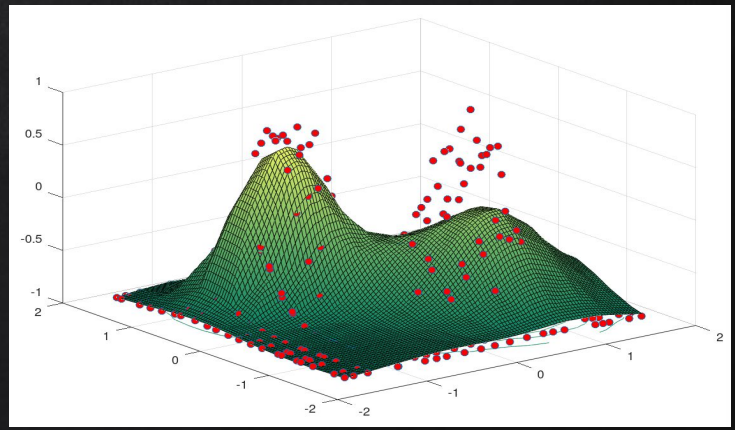
20%



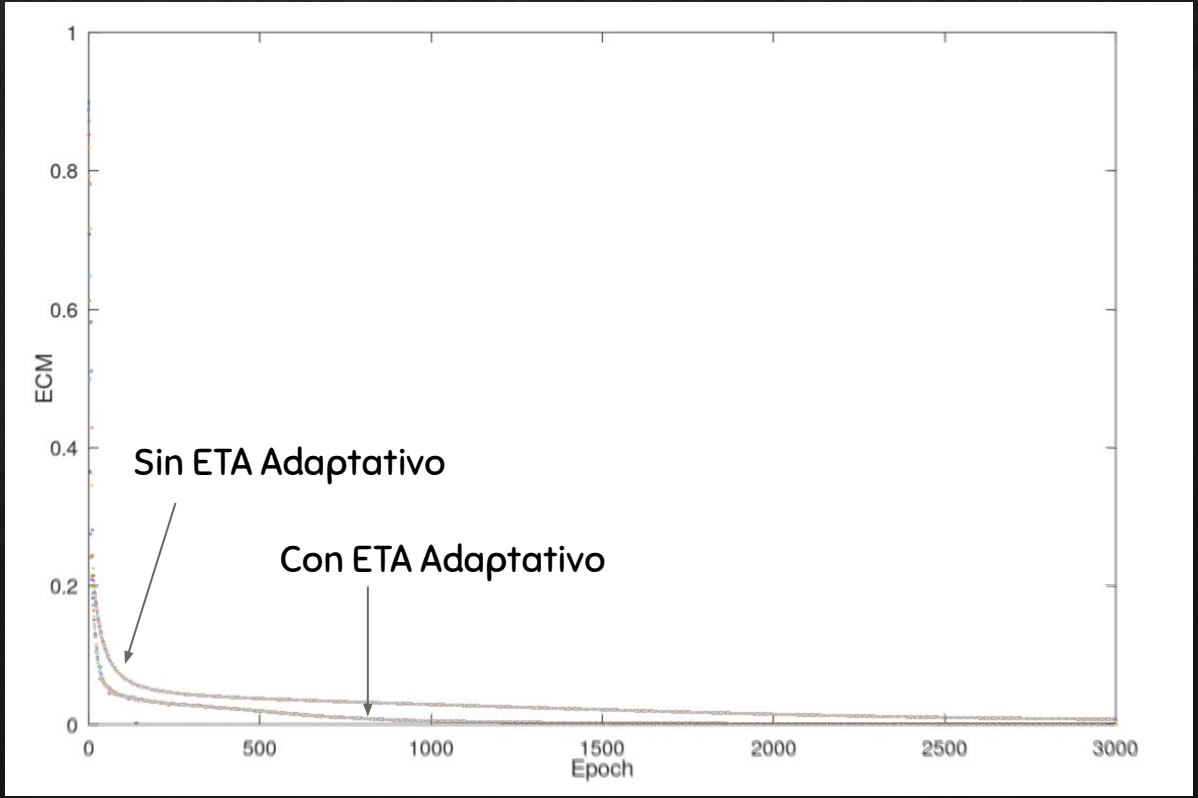
10%



5%



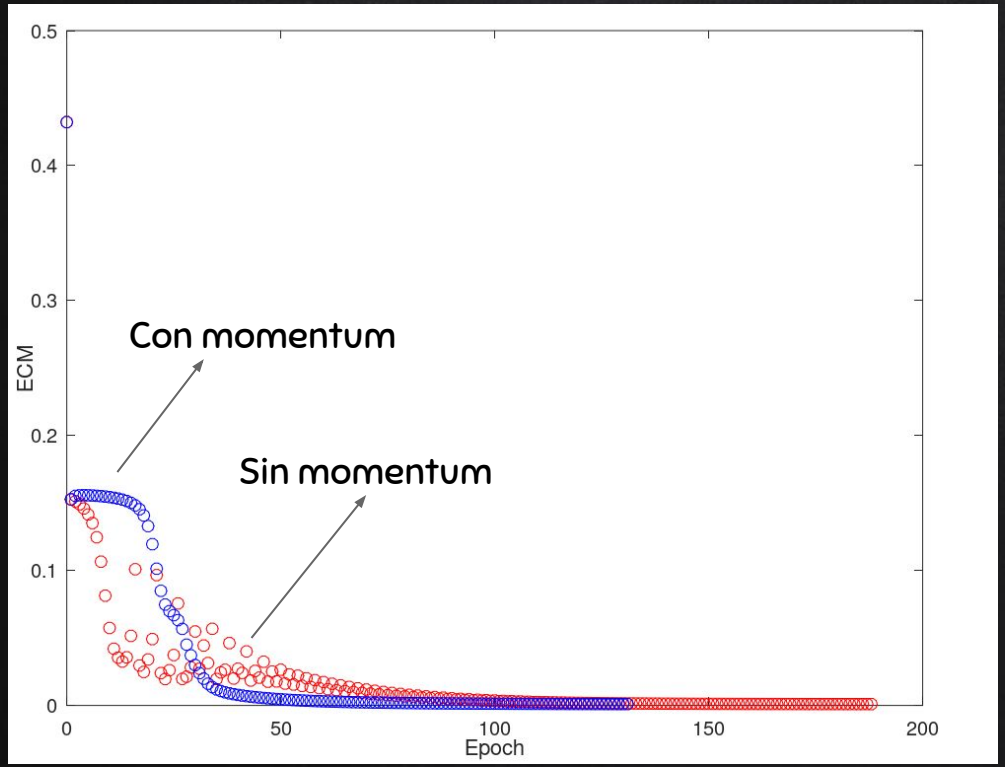
Comparativa:
ETA ADAPTATIVO

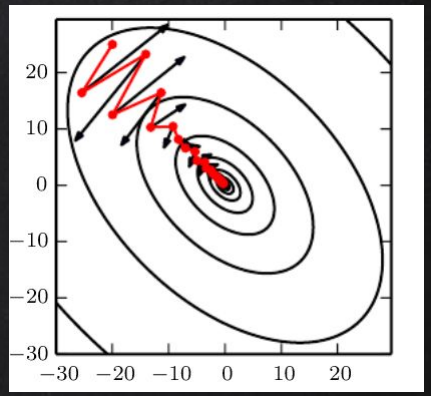


Batch
arquitectura [50 50]
ETA inicial de 0.001
Constante A = 0.0001
Constante B = 0.05
ETA Mínimo = 0.0001
ETA máximo = 0.005

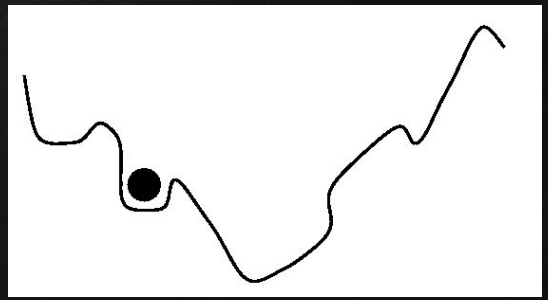
Comparativa: MOMENTUM

Batch
Arquitectura [50 50 50 50]
Momentum value 0.5

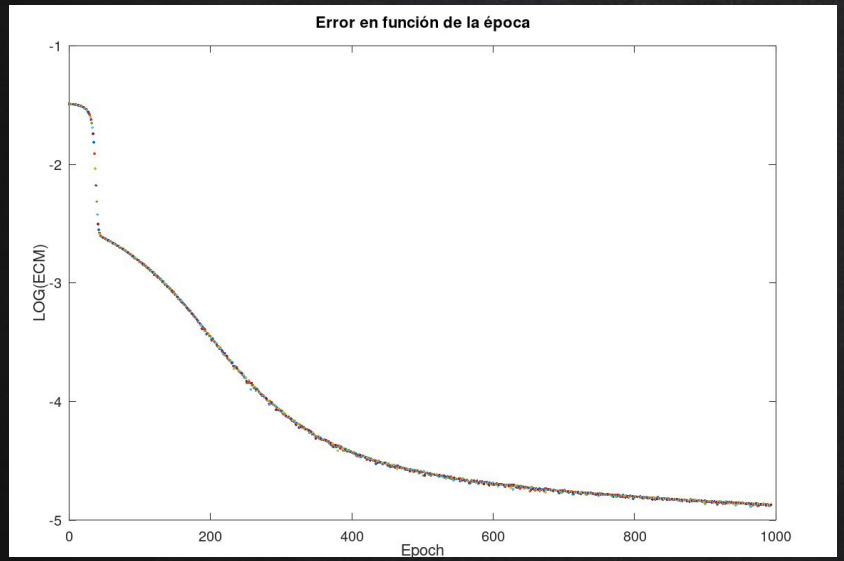




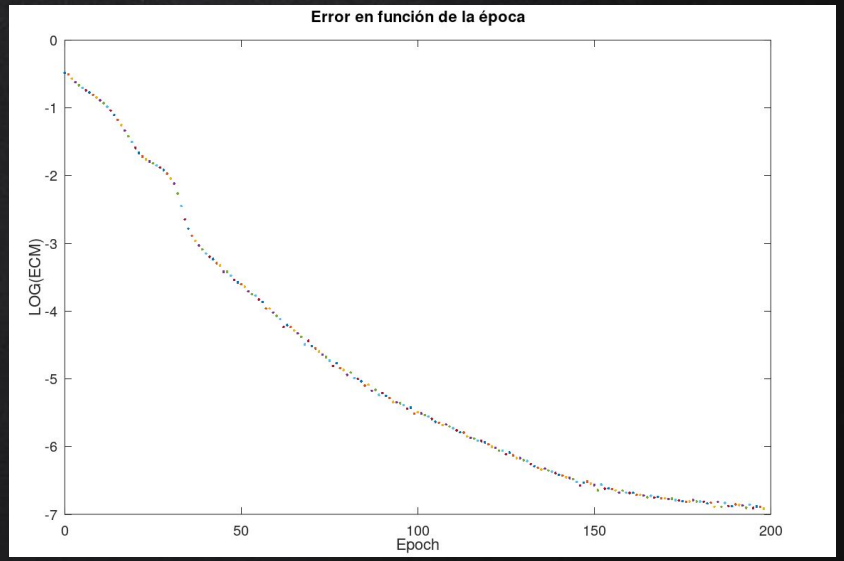
Minimo local



Comparativa:
EXP vs TAN



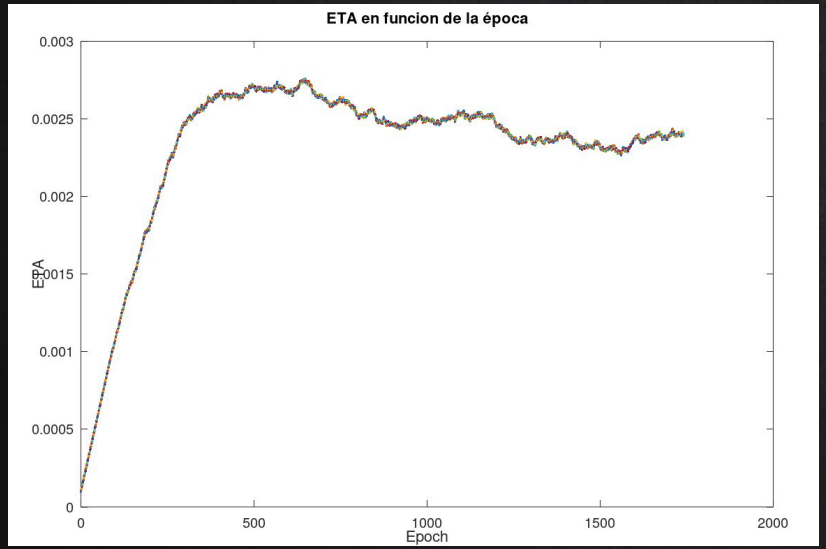
Exponencial



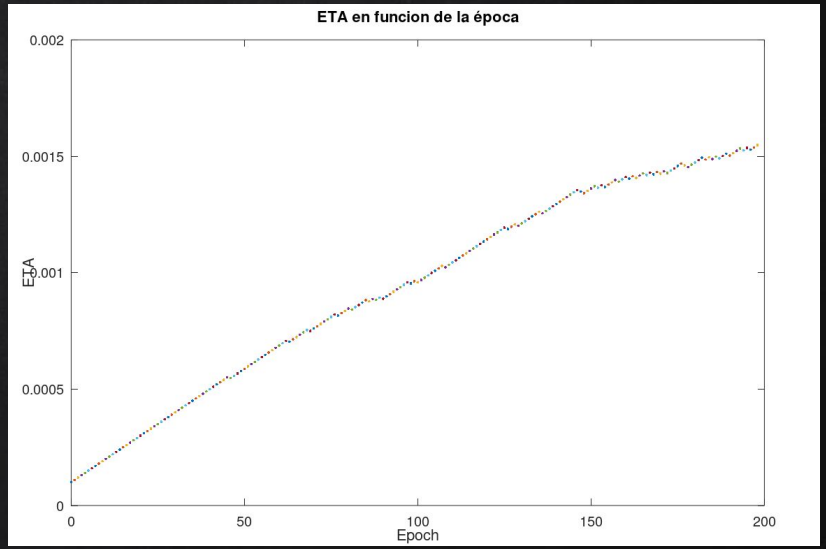
Tangencial

Beta = 1.5

Comparativa:
EXP vs TAN
Evolución del aprendizaje (ETA)



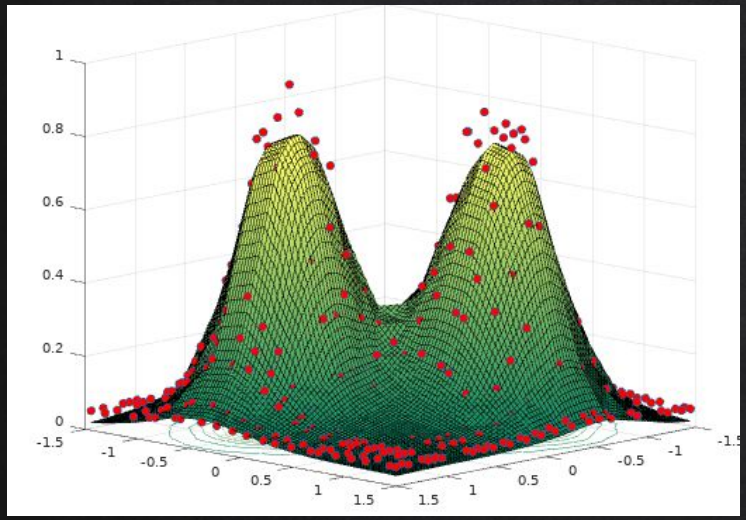
Exponencial



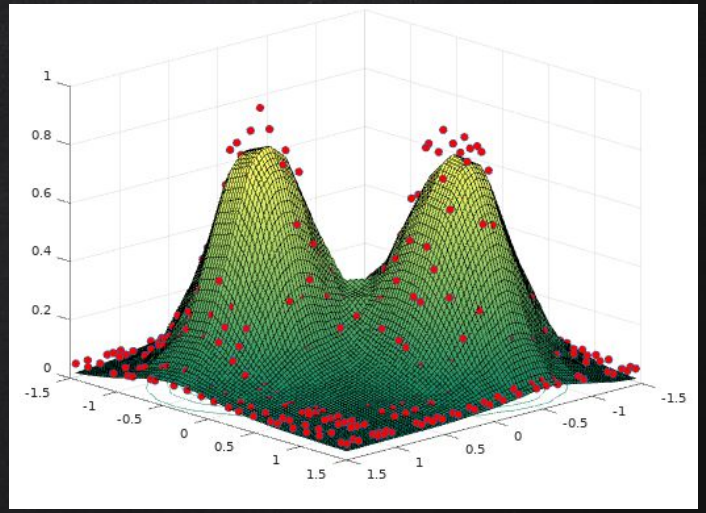
Tangencial

Beta = 1.5

Comparativa:
BATCH vs INCREMENTAL



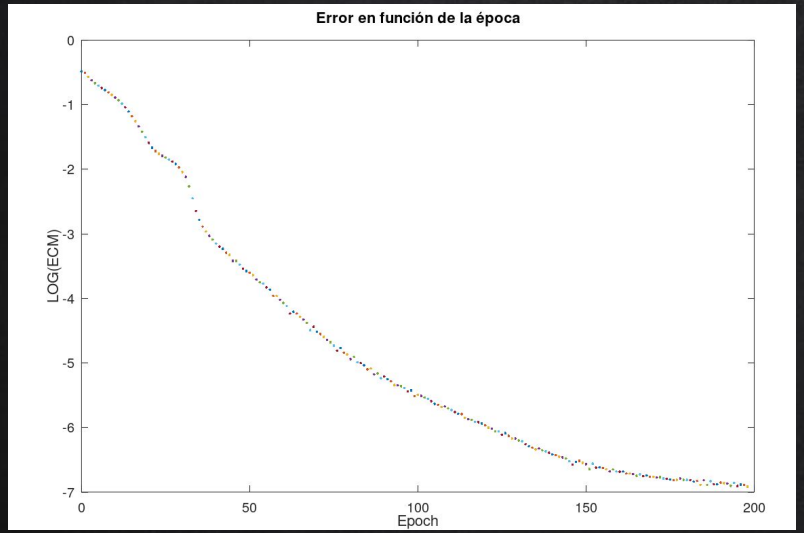
ECM = 0.074



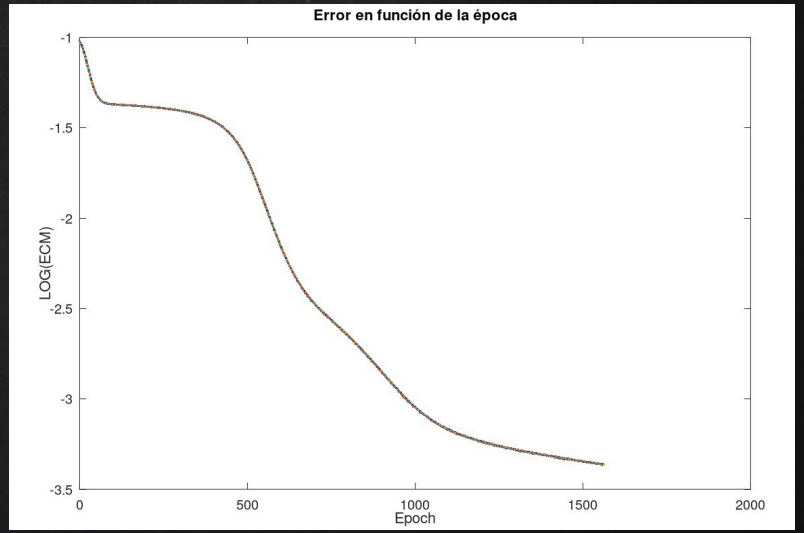
ECM = 0.072

BATCH	INCREMENTAL
• Full Data • Mejor Performance	• Real Time • Mas eficaz por teoria

Comparativa:
BETA con F. Tan.

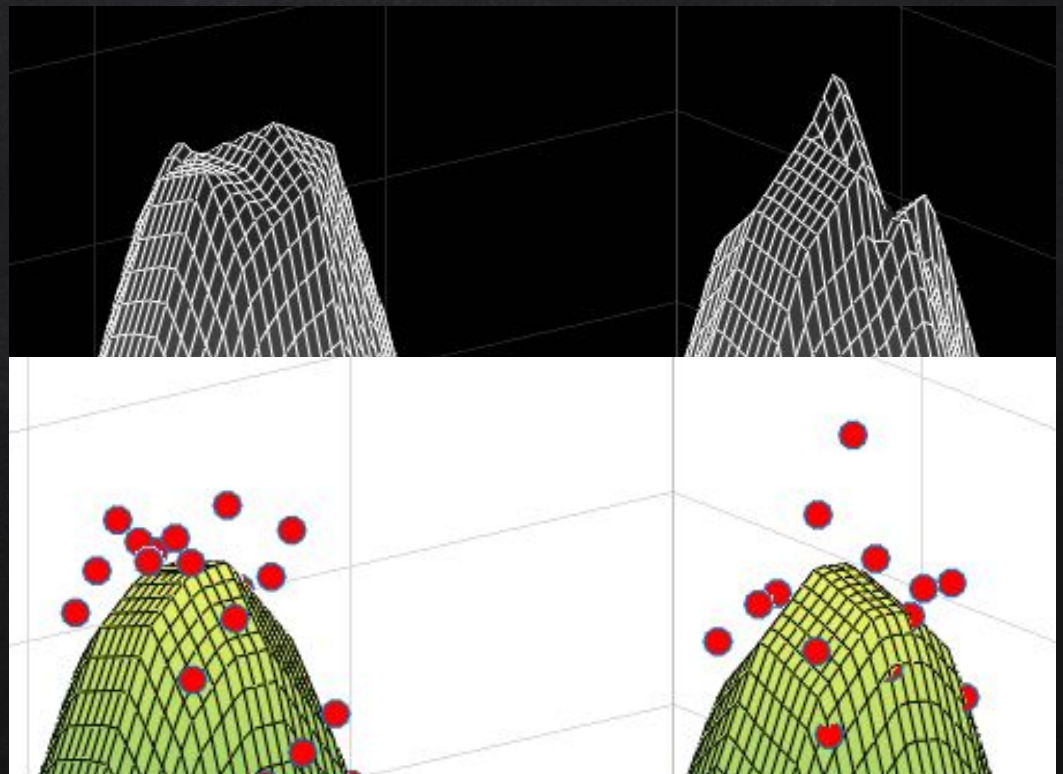


BETA = 1.5

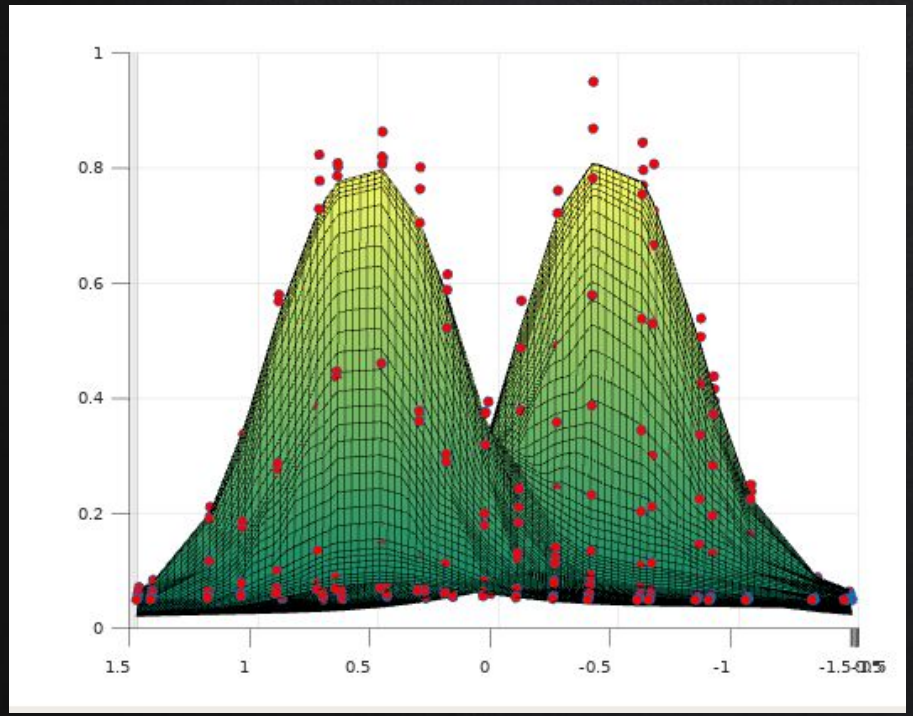


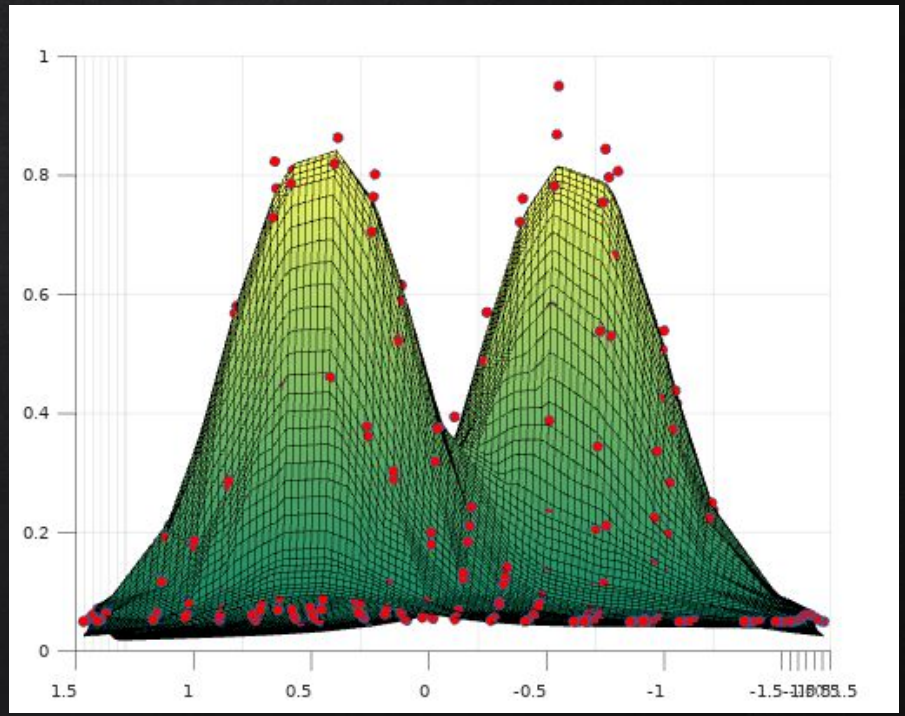
BETA = 0.5

El problema de las puntas



Comparativa:
puntos random vs puntos random + las puntas





Conclusiones

- El comportamiento de la red varía mucho en función a todos los parámetros mencionados, y los valores óptimos de los mismos también dependen de los demás para acelerar la convergencia del error.
- Siguiendo el teorema de aproximación universal (Hornik, Stinchcombe y White), pudimos ver que es posible tener una buena generalización con una única capa oculta de varias neuronas. La cantidad de capas y neuronas por capa fueron determinadas mediante la realización de pruebas empíricas
- Es importante una buena elección de patrones de entrenamiento en zonas donde la función presenta picos o valles porque es la zona donde más error puede ocurrir. En los segmentos lineales, la red aprende más fácilmente que en los picos/valles.

- Las optimizaciones propuestas, lograron mejorar los tiempos de convergencia.
- Respecto a la arquitectura en general, mientras más capas y neuronas utilizamos, obtuvimos mejores resultados.
- A mayor porcentaje de datos aprendidos, mejor nivel de generalización.

Gracias!

Preguntas?