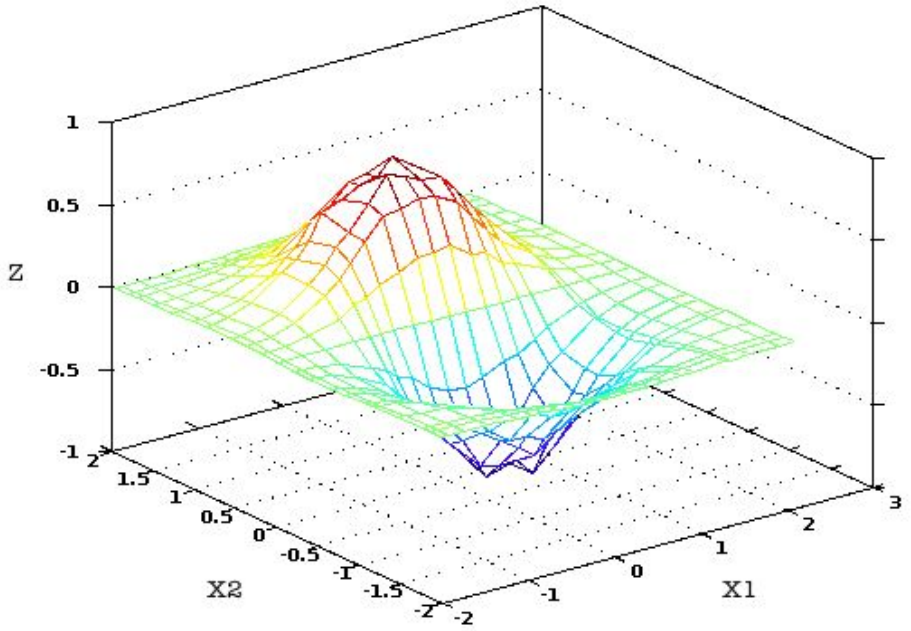
Redes Neuronales

Sistemas de Inteligencia Artificial

Introducción

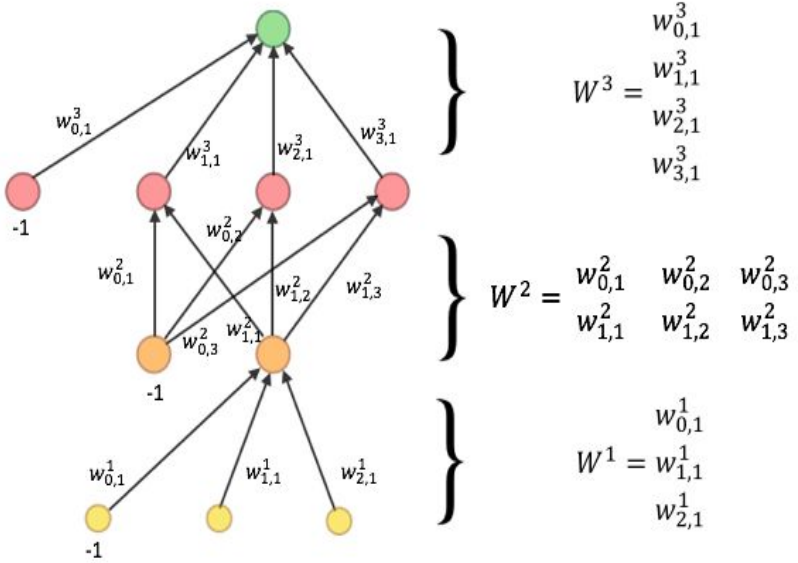
Objetivo

Desarrollar una red neuronal que permita simular un terreno del que se conocen un conjunto de puntos



Implementación

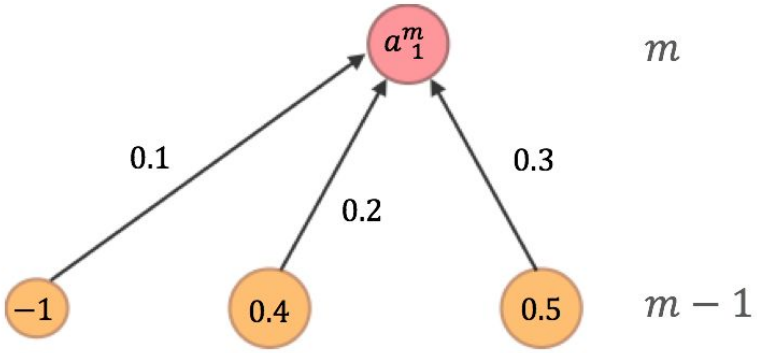
Perceptrón multicapa



La red neuronal es representada con un conjunto de matrices donde se almacenan los pesos de las conexiones entre las neuronas.

A cada capa de la red, le corresponde una matriz.

Funcionamiento de la red



$$a_1^m = g\big((-1) * 0.1 + 0.2 * 0.4 + 0.3 * 0.5\big)$$

La red neuronal recibe ciertos parámetros en su entrada y calcula el valor de activación de cada neurona.

Las neuronas en la capa de entrada reciben los parámetros iniciales como valor de activación

Las neuronas en las capas restantes calculan sus valores de activación a partir de los valores de las neuronas de la capa previa, los pesos de las aristas y una función de activación g

Aprendizaje

Feed Forward

Entrada ➡ Valor de activación ➡ Salida

Back Propagation

Salida ➡ Cálculo de deltas ➡ Corrección de los pesos

Normalización de los datos

Se normalizan tanto las entradas como las salidas

Dependiendo de la función de activación a utilizar, se normaliza entre 0 a 1 o entre -1 a 1

El error se calculó dependiendo la diferencia de lo que da la red y el valor normalizado de la función.

Mejoras del backpropagation

Momentum

Al hacer back propagation se obtiene un valor que representa la corrección que es necesario hacer en el peso de cada arista.

Momentum agrega a este valor una proporción de la variación obtenida para la misma arista en el paso anterior

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

Parámetros adaptativos

En el backpropagation, la variación que se debe aplicar al peso de cada arista depende de una constante **η**

Para evitar que la elección de este parámetro condicione el aprendizaje, **η** se actualiza paso a paso en base al error.

$$\Delta\eta = \begin{cases} +a & si \quad \Delta E < 0 \quad consistentemente \\ -b\eta & si \quad \Delta E \geq 0 \\ 0 & si\ no \end{cases}$$

Pruebas y resultados

Aclaraciones

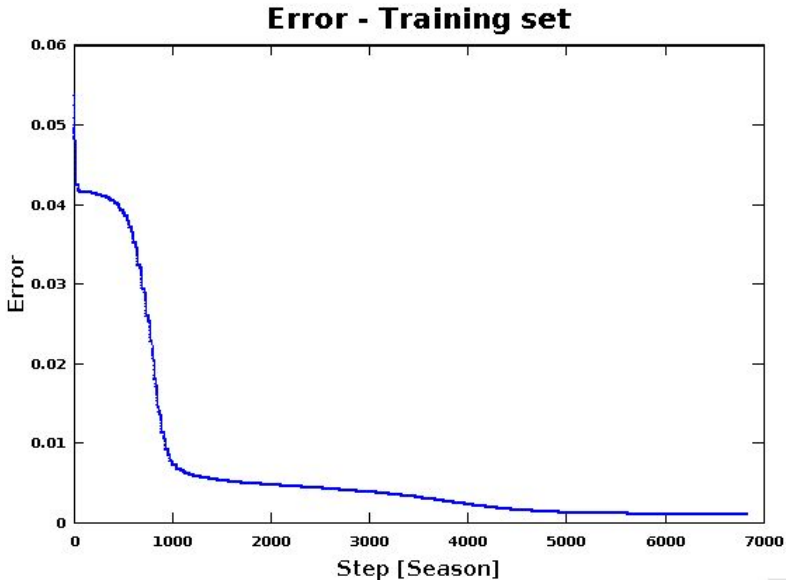
VALORES de parámetros:

Batch a = 0.002, b = 0.1, α=0.9, β=0.5, η=0.002.
Incremental a = 0.02, b = 0.1, α =0.9, β=0.5, η=0.02

Todas las pruebas salvo las de generalización se hicieron utilizando como conjunto de entrenamiento la totalidad de los datos

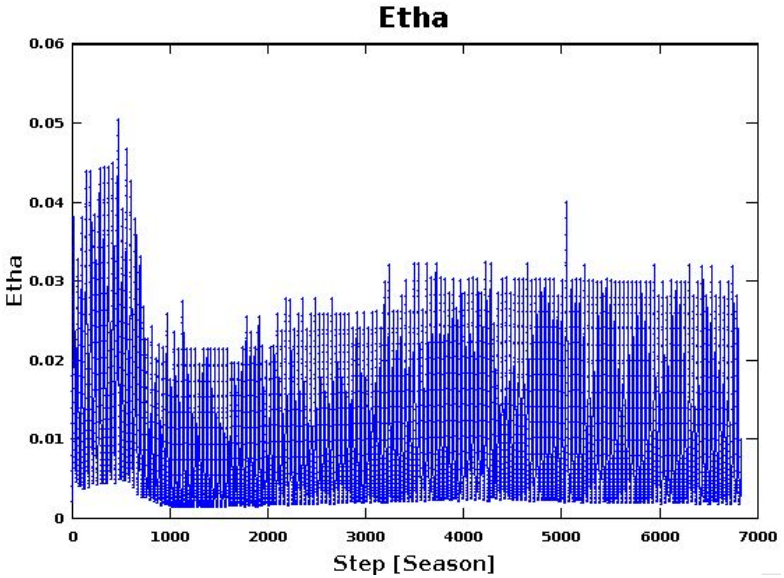
Salida

- Error del conjunto de entrenamiento en el tiempo.



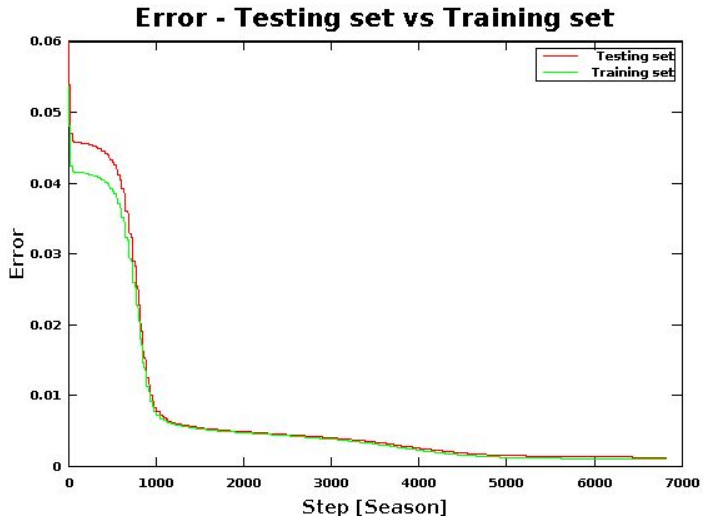
Salida

- Valor de η en el tiempo (η adaptativo).



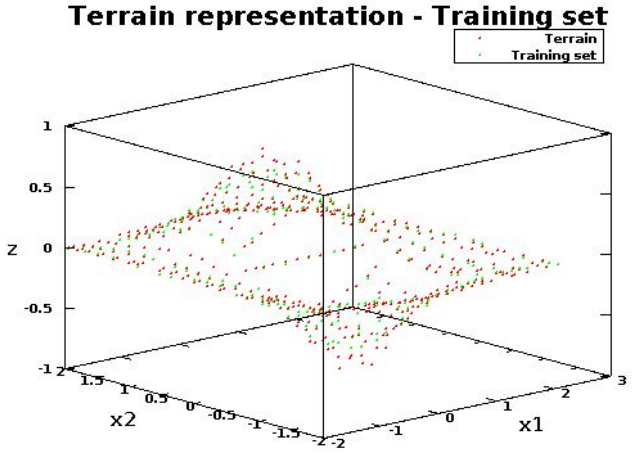
Salida

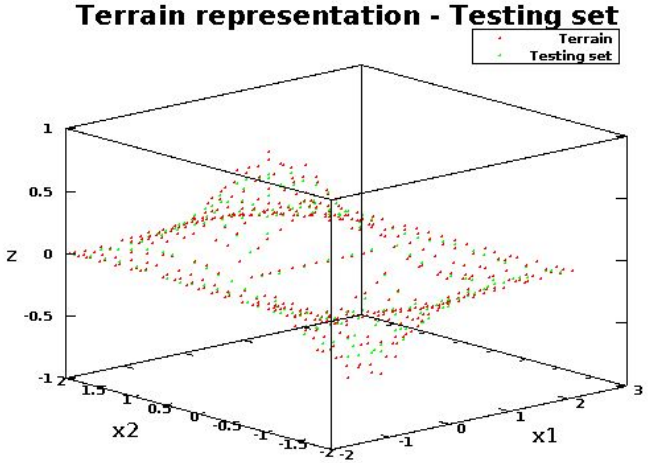
- Comparación del error del conjunto de entrenamiento con el de testeo en el tiempo.



Salida

- Visualización de la salida esperada comparada con la obtenida.





Comparación incremental - batch

Se quiere comparar la performance de los métodos incremental y batch con todas sus variantes (momentum, parámetros adaptativos, funciones de activación)

Las pruebas se realizan sobre una red de arquitectura 2 - 7 - 7 - 1

Comparación incremental - batch

	Error cuadrático medio	Épocas	Patrones	Tiempo[sec]	Función de activación	Momentum	Etha adaptativo	% aprendido
Batch	**0,001000**	27752	12238632	99,07	tanh	no	no	95,05
	0,001000	**9964**	**4394124**	**35,19**	tanh	si	no	**95,28**
	0,001000	23560	10389960	85,73	tanh	no	si	94,78
	0,001000	12378	5458698	**45,17**	tanh	si	si	95,23
	0,025972	500000	220500000	2269,76	exp	no	no	77,82
	0,025912	500000	220500000	2415,87	exp	si	no	78,91
	0,025781	500000	220500000	2465,16	exp	no	si	65,98
	0,025765	500000	220500000	2423,27	exp	si	si	79,36
Incremental	0,042989	1000	441000	2970,71	tanh	no	no	33,33
	0,040050	1000	441000	2985,45	tanh	si	no	29,93
	0,044187	1000	441000	2978,00	tanh	no	si	40,59
	0,043600	1000	441000	2984,91	tanh	si	si	32,43
	0,053550	1000	441000	3164,47	exp	no	no	54,2
	0,053168	1000	441000	3323,19	exp	si	no	55,1
	0,052949	1000	441000	3302,66	exp	no	si	55,1
	0,043171	1000	441000	3217,89	exp	si	si	42,4

Comparación de arquitecturas

Se quiere ver cómo influencia el cambio de arquitectura de la red a su performance para aprender

Se utilizó como método el batch con momentum, los valores de los parámetros son los usados anteriormente

Comparación de arquitecturas

	Error cuadrático medio	Épocas	Tiempo[sec]
[2 7 1]	0,002203	500000	1317,59
[2 10 1]	0,001000	319533	1227,03
[2 13 1]	0,001000	332673	1402,09
[2 16 1]	0,001000	133860	631,53
[2 7 7 1]	0,001000	7031	35,62
[2 10 10 1]	0,001000	7052	43,69
[2 13 13 1]	0,001000	12712	92,07
[2 16 16 1]	0,001000	6360	52,4
[2 7 7 7 1]	0,001000	7254	43,57

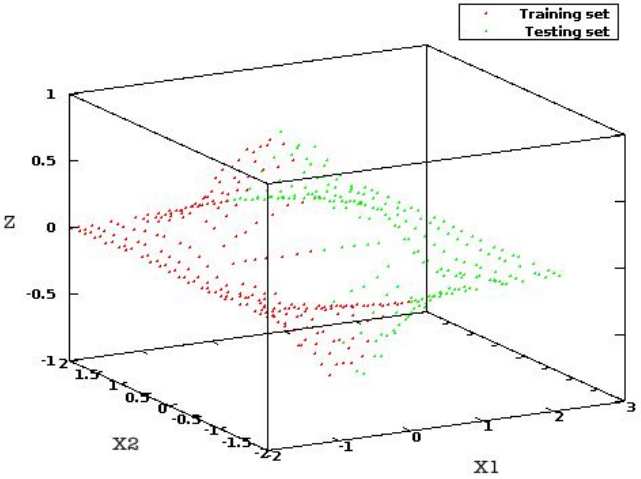
Pruebas de generalización

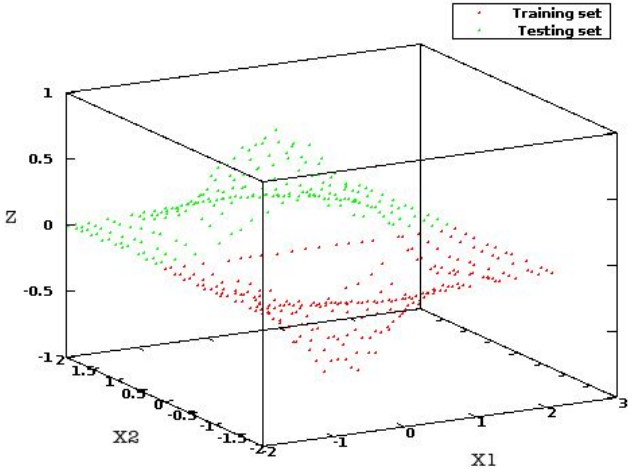
El propósito es ver qué porcentaje de un conjunto que no fue incluido en el entrenamiento, es correctamente generalizado

Para ver si un punto está bien generalizado se evalúa la diferencia entre el valor obtenido por la red y el esperado. La diferencia se eleva al cuadrado y se divide por dos.

Pruebas de generalización

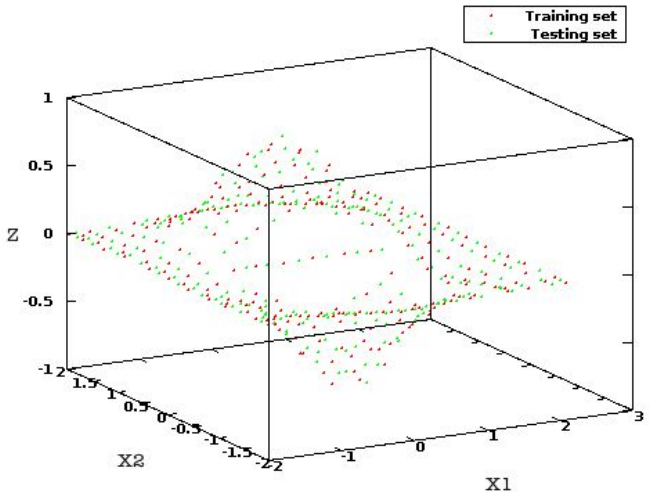
HALF

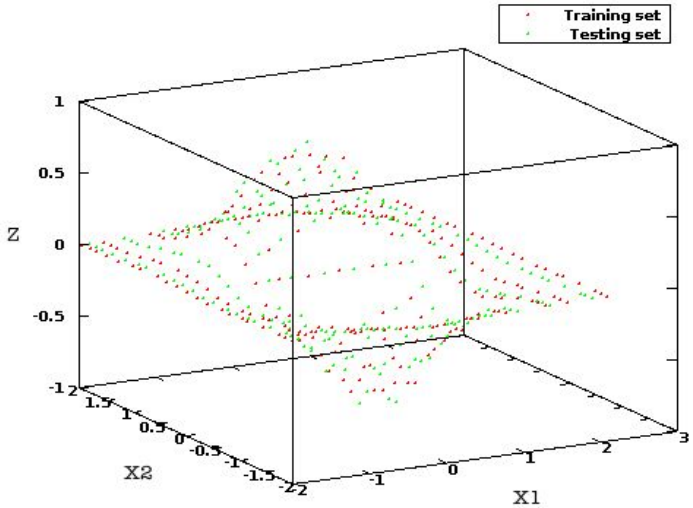




Pruebas de generalización

SKIPPING





Pruebas de generalización

Arquitectura [2 7 7 1]

	Error cuadrático medio del conj de entrenamiento	Error cuadrático medio del conj de testeo	Épocas	Patrones	Tiempo [sec]	% Aprendido del conj. de entrenamiento	% Aprendido del conj. de testeo
skipping1	0,001	0,001166	29828	6591988	75,24	95,02	94,55
skippingrows	0,001	0,001508	41168	9509808	109,37	94,81	92,86
half1	0,001	0,08	19051	4400781	52,21	94,37	32,38
half2	0,001	0,175209	6542	1511202	18,02	96,1	15,71

Arquitectura [2 16 16 1]

	Error cuadrático medio del conj de entrenamiento	Error cuadrático medio del conj de testeo	Épocas	Patrones	Tiempo [sec]	% Aprendido del conj. de entrenamiento	% Aprendido del conj. de testeo
skipping1	0,001	0,001134	15351	3392571	66,67	94,57	93,63
skippingrows	0,001	0,001489	17583	4061673	75,69	93,5	92,85
half1	0,001	0,079665	9289	2145759	37,02	93,93	32,38
half2	0,001	0,198496	9106	2103486	36,43	95,67	8,57

Conclusiones

Conclusiones

- Batch más rápido que Incremental por temas de implementación
- Como función de activación, la tangente hiperbólica fue superior
- La mejor combinación fue batch con momentum
- De las arquitecturas probadas, las de 2 capas lograban un error menor en menos tiempo
- Generalizó muy bien en las pruebas de skipping

Preguntas