Sistemas de Inteligencia Artificial

Trabajo Practico 1

Redes Neuronales

Grupo 7:

Alejo Hegouaburu – Legajo: 47279
Daniel Azar – Legajo: 48317
Federico Santos – Legajo: 44228

Introducción:

Las redes neuronales constituyen una abstracción del funcionamiento del cerebro humano, con el objetivo es resolver problemas para los cuales no se conoce un algoritmo que los permita obtener una solución exacta en tiempo razonable.

Estas redes se basan en los pesos asignados a las conexiones entre las neuronas, permitiendo así aproximar una salida para una entrada dada.

Los pesos de dichas conexiones se obtienen a partir de un proceso de entrenamiento de la red. El entrenamiento consiste en modificar los pesos en base a un patrón de entrada para el cual se conoce la respuesta.

Luego de presentarle varios patrones a la red, los pesos tomaran valores de manera que se pueda reproducir la respuesta a dichos patrones (aprendizaje) y estimar la salida para entradas desconocidas (generalización).

En este informe se describirá el funcionamiento y entrenamiento de una red neuronal destinada a representar una función matemática en el espacio. Se analizaran los resultados obtenidos con distintas configuraciones de la red y distintas estrategias de entrenamiento. El objetivo consiste determinar cual es la configuración que brinda mayores resultados.

Descripción del problema:

El problema que da el origen a este informe es el reconocimiento por medio de una red neuronal de la siguiente función matemática:

$$F(x,y) = 3*((1-x)^2)*\exp(-x^2-(y+1)^2)$$

En la Figura 1 se grafica dicha función para los valores que se encuentran dentro de su dominio. Se puede observar que consiste en una región plana con una protuberancia similar a un "cono redondeado" con centro en el punto (-0.5, 1). La imagen de la función se encuentra acotada entre 0 y 6.

La protuberancia señalada anteriormente constituye una región crítica para el aprendizaje de la red neuronal, ya que es ahí donde los valores varían más abruptamente. Se podría considerar que es ahí donde hay mayor densidad de información en la función.

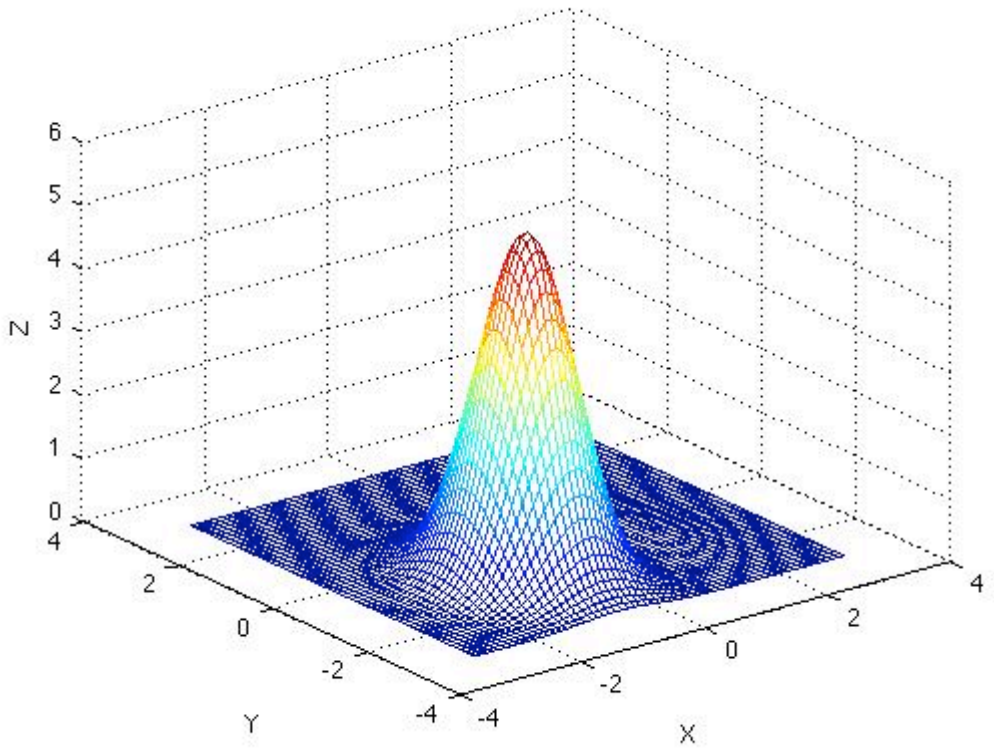

Figura 1: Grafico de la función a representar.

Implementación y arquitectura de la red:

La primera caracterización que se puede hacer de una red neuronal hace referencia a su tipo de aprendizaje o entrenamiento. En este informe solo se utilizaran redes con aprendizaje supervisado. Esto significa que la red ajusta los pesos a partir del conjunto de datos y respuestas conocidas que se le proporciona. Existen otros tipos de entrenamiento, pero exceden el alcance de este informe.

El otro aspecto que caracteriza a una red neuronal es la cantidad de capas y de neuronas que tiene la misma. Toda red tiene al menos una capa de entrada y una capa de salida. Según la cantidad de capas definimos a las redes como redes simples o multicapa. Se consideran redes simples a aquellas que solo tienen la capa de entrada y salida.

Las redes multicapa son las que tienen capas internas. Es decir, tienen nodos ubicados en capas que se encuentran entre la entrada y la salida. Estas redes sirven para resolver problemas no linealmente separables, problemas que no son aprendibles por una red simple.

Por esta razón se usaran redes multicapa para la representación de la función presentada anteriormente. En el desarrollo de este informe de analizaran los resultados obtenidos para redes con distintas cantidades de capas y distinta cantidad de nodos por capa. El objetivo será determinar cual es la combinación que permite reducir el error de la aproximación y el tiempo de entrenamiento.

Para lograr un buen aprendizaje de la red, es necesario presentarle los patrones de entrenamiento reiteradas veces. De esta forma, se reduce la diferencia entre las salidas esperadas y las aproximadas por la red. Cada vez que se presenta el lote completo de patrones de denomina época. El tiempo de entrenamiento de encuentra íntimamente relacionado con la cantidad de épocas necesarias para lograr el error esperado.

1. Funciones de activación:

Se utilizan dos funciones de activación distintas: la exponencial (sigmoidea) y la tangente hiperbólica. Ambas funciones incorporan una constante que llamaremos beta, que modifica la "pendiente" de las mismas. Los valores típicos de beta son 0,5 o 1.

La función exponencial viene dada por la expresión:

```
F(x) = 1 / (1 + exp(-2*x*beta));
```

La imagen de la misma esta acotada en el intervalo (0, 1). En la Figura 2 se muestra la grafica para dos valores distintos de beta.

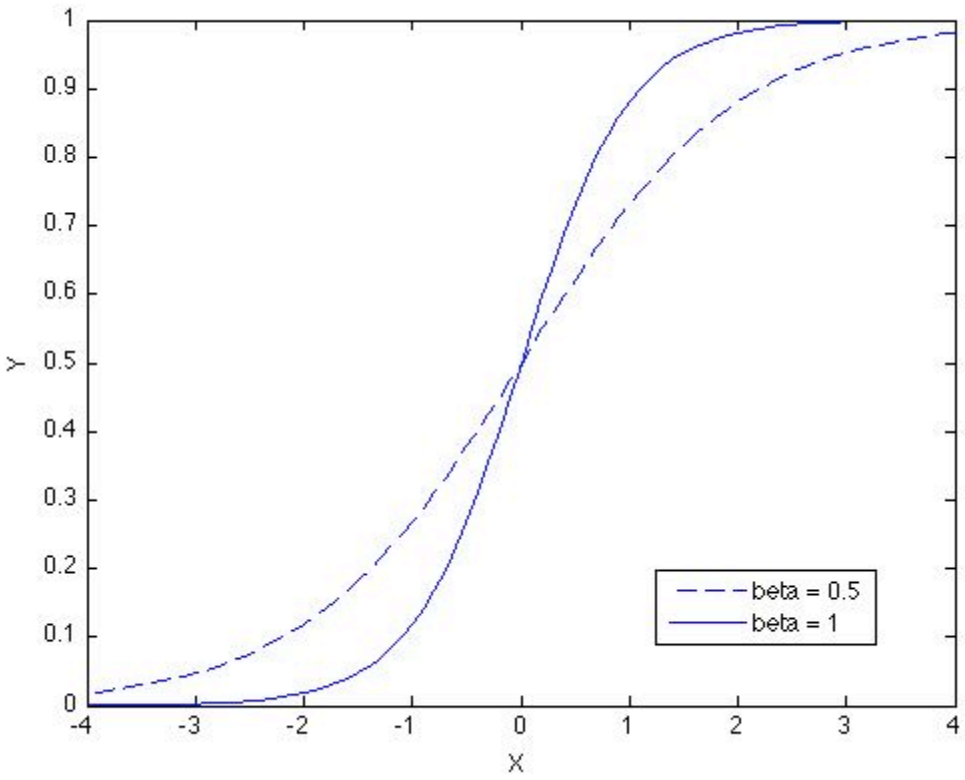

Figura 2: Función exponencial.

Por otra parte, la tangente hiperbólica es una función cuya imagen se encuentra acotada en el intervalo (-1, 1). La grafica se muestra en la Figura 3, para los dos valores de beta.

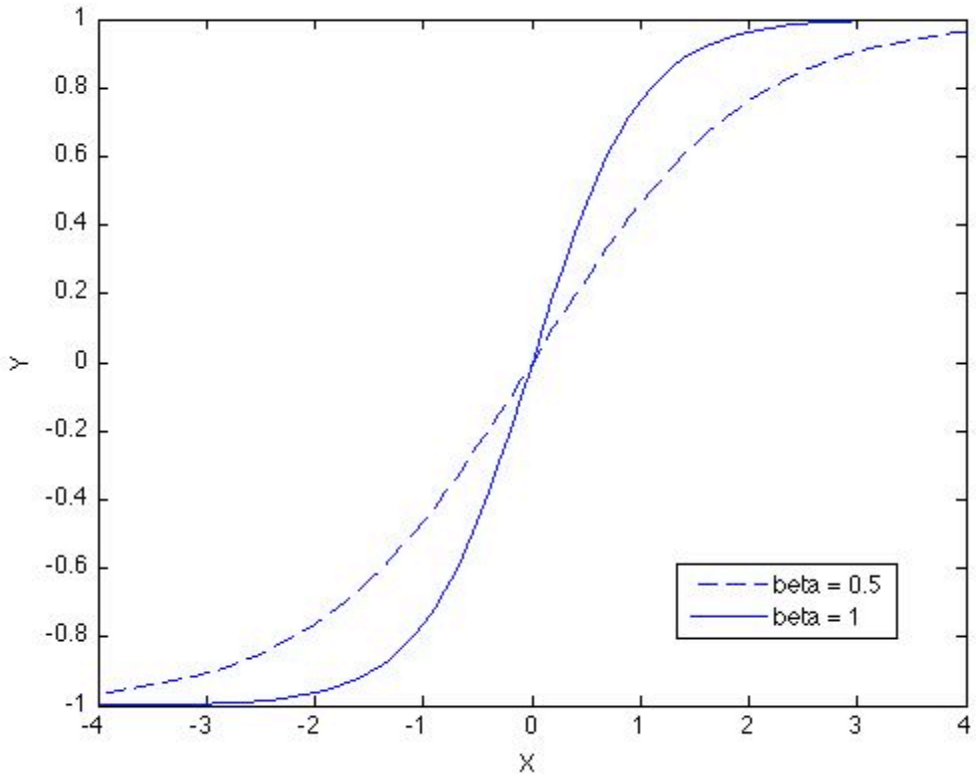

Figura 3: Función tangencial.

2. Normalización de la entrada:

Debido a que la imagen de la función que debe aprender la red es diferente a la imagen de la función de transferencia, el error al comparar la respuesta de la red y los valores esperados interfiere en el aprendizaje de la red.

Como solución a este problema, se decidió normalizar la entrada. Para esto se aplicaron dos transformaciones. La primera adapta el intervalo de la imagen de la función de entrada para que tenga la misma longitud que la función de transferencia.

La segunda adapta los limites de la imagen de la entrada a los de la función a los limites de la función de transferencia, de forma tal que la media de ambas sea la misma.

Por ultimo, la respuesta de la red es desnormalizada para que sea equivalente a la función de entrada. Este proceso tiene como desventaja que se aumenta el error de los datos calculados, siendo el mismo superior a la cota establecida para el aprendizaje.

3. Mejoras realizadas:

Con el fin de acelerar el proceso de aprendizaje, se implementaron dos mejoras al algoritmo back-propagation. Estas son: Factor de aprendizaje (Eta) adaptativo y Momentum.

La primera consiste en modificar el factor de aprendizaje al finalizar cada una de las épocas del entrenamiento. Esta modificación se efectúa según los resultados obtenidos en los pasos anteriores.

Si el error global aumenta, se disminuye el valor de eta en un 5%. En el caso donde el error global disminuye consistentemente en las ultimas 4 épocas se aumenta el factor de aprendizaje en un 10%.

Se decidió incrementar el parámetro en porcentaje, ya que si se aumenta en una constante se debería seleccionar la misma dependiendo del eta inicial. Si eta es chico y se aumenta en una constante fija durante varios pasos seguidos, el error se vuelve inestable.

En el caso de que el paso sea malo (el error aumenta), se desecha ese paso volviendo al estado anterior y se ejecuta el algoritmo con el nuevo valor de eta.

La segunda modificación corresponde a la implementación de "momentum" en el algoritmo back-propagation. Esta modificación consiste en computar la variación del peso de una conexión teniendo en cuenta la variación de ese peso obtenida en el paso anterior.

Por último, se modificaron las derivadas de las funciones de transferencia, agregándoles una constante cercana pero distinta de cero. Si bien esta modificación resulta trivial, nos asegura que la derivada nunca de cero, anulando el aprendizaje.

4. Patrones de entrenamiento:

Se proponen dos métodos para generar los patones para entrenar la red. El primero consiste en la división uniforme del plano XY, generando una grilla de puntos para los cuales se calcula la componente Z aplicando la función dada. Este método nos garantiza que se cubre todo el dominio de la función. Si la cantidad de puntos es lo suficientemente grande, debería representarse la región crítica de la función.

La otra alternativa consiste en generar puntos de forma aleatoria en el plano XY, incrementando la probabilidad en el área critica de la función. Esto nos asegura tener más puntos en la región donde se encuentra la mayor cantidad de información. De esta forma, la red aprenderá mejor la forma de la protuberancia pero perderá precisión en la región plana.

La Figura 4 muestra la distribución aleatoria de puntos en el plano XY que se explico anteriormente. Se puede observar que la cantidad de puntos es mayor en la zona cercana al punto (-0.5, -1), región donde esta ubicada la protuberancia.

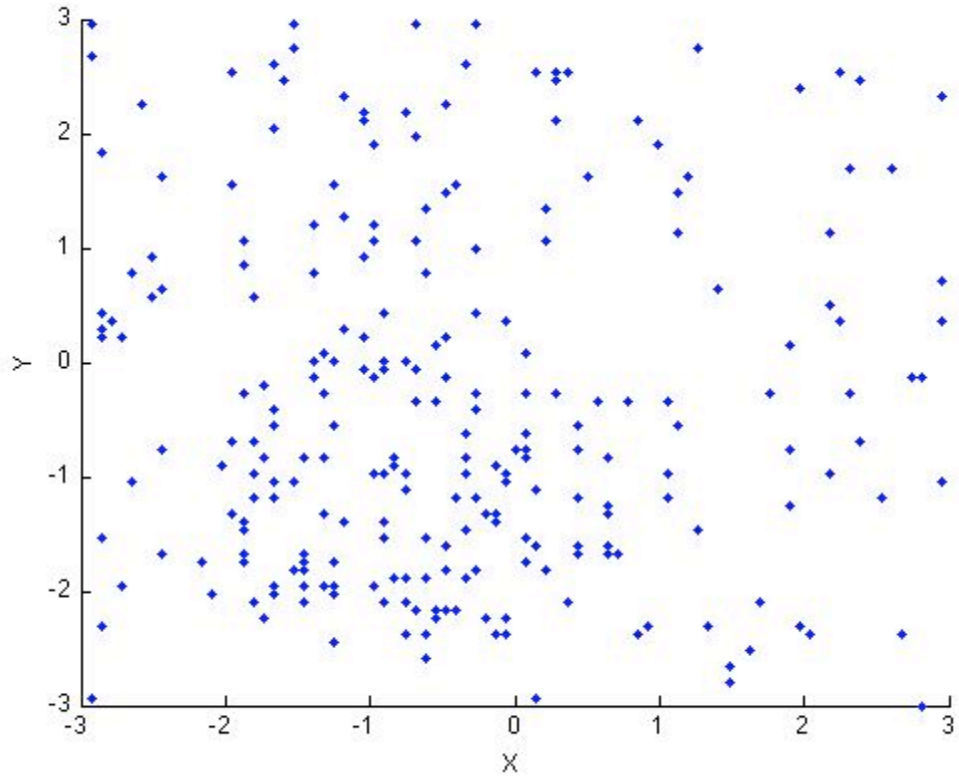

Figura 4: Distribución de patrones aleatoria.

Resultados:

Dado que la red debe aprender una función de dos variables, la misma deberá tener dos entradas y una neurona de salida. En este apartado se comentaran los resultados obtenidos a partir de distintos ensayos con diferente cantidad de capas y neuronas por cada capa, con el objetivo de determinar cual es la arquitectura optima para este problema.

Para el entrenamiento de la red se usaron las dos funciones de transferencia comentadas en los apartados anteriores, y de eligió un valor de beta = 0.3 en ambos casos y se uso en todos los ensayos (salvo que se indique lo contrario).

Se eligió una cota para el error global igual a 0.03. Esto indica que se entrenara la red hasta que el error global en la ultima época se menor que dicha cota. Si bien este valor puede parecer alto, permite que el entrenamiento sea mas corto. De esta forma, se puede ejecutar una mayor cantidad de pruebas para diferentes parámetros.

Se uso la misma cota para decidir si un patrón fue aprendido correctamente. De esta forma se calcula el porcentaje de aprendizaje. Para calcular el porcentaje de generalización se calcula la salida de la red para una cantidad de valores considerablemente mas grande que la cantidad de patrones de entrenamiento. Estos resultados se comparan con los valores esperados, usando una cota con un valor del doble que la cota de aprendizaje. En este caso, el valor será 0.06.

Para calcular este porcentaje de generalización, los valores utilizados corresponden a un grillado uniforme del plano XY, tomando valores entre -3 y 3 en ambos ejes, con un paso de 0.01.

Los resultados que se muestran en la Tabla 1 se computaron a partir de un set de patrones generado aleatoriamente, tal como se explico en los apartados anteriores. El set consiste en un conjunto de aproximadamente 140 entradas.

Fun. Act.	Arquitectura	Aprendizaje	Generalización	Épocas	Momento
Tangencial	[8 2 1]	90.07	89.37	3033	0.5
Exponencial	[8 2 1]	86.52	92.83	436	0.5
Tangencial	[10 6 1]	85.81	88.62	813	0.5
Exponencial	[10 6 1]	87.23	91.55	148	0.5
Exponencial	[10 6 1]	87.94	92.16	156	0
Exponencial	[10 6 1]	87.94	94.43	667	0.8
Exponencial	[15 3 1]	87.93	92.5	363	0.5

Tabla 1: Entrenamiento aleatorio.

La primera conclusión que se puede obtener de la Tabla 1, es que usando una función de transferencia exponencial se requiere una menor cantidad de épocas para completar el entrenamiento de una red de iguales características. En ambos casos se obtiene una mejor generalización con la función exponencial.

Cuando se entrena una red con 10 neuronas en la primer capa oculta y 6 en la segunda sin usar “momentum” se incrementa el numero de épocas. Lo mismo sucede si se aumenta el valor del mismo. Con un valor de 0.8, el numero de épocas aumento considerablemente. Con un valor para el “momentum” de 0.5 se obtienen los resultados óptimos.

Cuando se aumenta la cantidad de neuronas en cada una de las capas, pasando de 8 y 2 a 10 y 6 respectivamente, se obtienen mejores resultados tanto en la cantidad de épocas necesarias para terminar el entrenamiento y en los porcentajes de generalización y aprendizaje.

No obstante, si se continua aumentando el numero de neuronas, se mantienen los porcentajes de generalización y aprendizaje pero aumenta la cantidad de épocas necesarias para entrenar la red.

En la Tabla se muestran los resultados obtenidos usando un grillado uniforme de 169 puntos para generar los patrones de entrenamiento. Se uso un momento de 0.5 y el resto de los parámetros se mantuvieron iguales.

Fun. Act.	Arquit.	Aprend.	Generaliz.	Epocas	Grillado	Puntos
Exp.	[10 6 1]	85.79	97.33	392	0.5	169

Tabla 2: Entrenamiento uniforme.

Con este tipo de entrenamiento se necesitaron mas épocas para entrenar la red, comparada con la cantidad necesaria usando un entrenamiento aleatorio.

En cuanto a los resultados, el porcentaje de aprendizaje disminuyo mientras que el porcentaje de generalización mejoro considerablemente. Esto se debe a que este tipo de

grillado tiene mas puntos en las regiones planas, ayudando a la generalización en esa zona donde los resultados obtenidos con patrones aleatorios es mas pobre.

Conclusiones:

En base a los resultados obtenidos en la sección anterior, podemos decir que la arquitectura que mejor aprende la función en cuestión es aquella que tiene 10 unidades en la primer capa y 6 en la segunda, con una neurona de salida. El valor de momento mas adecuado para el problema es 0,5 , según lo observado el los distintos experimentos.