Sistemas de Inteligencia Artificial

Redes Neuronales

Ducret, Argentino - 52194
Gutierrez, Ignacio - 54293
Prudhomme, Franco - 54263

Índice

- Índice
- Introducción
- La función
- Implementación de la red
- Funciones de activación
- Conjuntos de entrenamiento y testeo
- Mejoras implementadas
  - Momentum
  - Eta adaptativo
- Resultados de las diferentes arquitecturas
  - Incremental (sin mejoras)
  - Momentum
  - Eta adaptativo
- Conclusiones
- Anexo

Introducción

El objetivo del trabajo práctico fue implementar una red neuronal con **aprendizaje supervisado** la cual aproxime la función desconocida *f(x, y)* con la muestra provista por la cátedra.

En nuestro caso, la muestra analizada fue la del documento ***terrain8.txt***.

Tal como se pide en el enunciado del trabajo, únicamente la "Parte 2" del enunciado va a estar detallada en el informe.

La función

En nuestro caso, tal como se mencionó en la introducción, el terreno a analizar fue el que se encontraba en el documento ***terrain8.txt***. El mismo, tal como se puede observar en las fotos obtenidas al graficarla [*Figura 1, 2, 3 y 4*] se observa que presenta tres (3) máximos y tres (3) mínimos, y la misma está comprendida en su eje x entre [-2.7931, 3.0094], en su eje y entre [-2.7281, 2.7949] y en su eje z entre [-1.0567, 1.0845].

Implementación de la red

Para la implementación de la red se utilizó el lenguaje de programación **Octave**.

Se representó a la red neuronal como un **cell array** de pesos de las conexiones entre cada neurona, siendo cada índice de dicha estructura una matriz que representa cada "capa" de conexiones, habiendo la cantidad de capas de conexiones como capas de neuronas ocultas más uno.

Luego también se utilizó un **cell array** para representar cada salida de cada neurona en las capas ocultas, teniendo la misma dimensión que la estructura de conexiones mencionada anteriormente. Siendo el último índice el resultado de la red evaluada en el input de entrada.

Para representar los inputs y outputs se utilizó un **cell array** en el cual su primer índice contiene una matriz cuyas dimensiones de columna son la longitud de cada pattern, mientras que las dimensiones de filas son la cantidad de patterns que se le da a la red neuronal para que aprenda. Mientras que su segundo índice contiene una matriz cuya cantidad de columnas es la longitud de la salida esperada y la cantidad de filas es equivalente a la matriz previa.

El output de la función original es normalizado siguiendo un algoritmo de dos pasos. Dado un dataset D de números donde m es el mínimo y M es el máximo, se definen los siguientes valores:

- range = M - m
- a = m + range/2

- b = range/2

La normalización luego consiste en restar a todos los elementos de D el valor de a y luego dividir por b a todos los elementos. Para desnormalizar simplemente se hace el proceso inverso: se multiplica por b al dataset normalizado y luego a todos sus elementos se les suma el valor de a.

Funciones de activación

Se utilizaron dos (2) funciones de activación a lo largo de las pruebas, las mismas siendo la tangente hiperbólica y la función exponencial.

La tangente hiperbólica posee la siguiente forma:

$$f(x) = tanh(x * \beta)$$

cuya derivada es la siguiente:

$$f(x) = \beta * (1 - x^2)$$

y la función exponencial:

$$f(x) = \frac{1}{1 + e^{-2 * x * \beta}}$$

cuya derivada es la siguiente:

$$f(x) = 2 * \beta * x * (1 - x)$$

Conjuntos de entrenamiento y testeo

Se decidió que para testear la red neuronal se le indique un porcentaje de la muestra que se lee del archivo y en base a dicho porcentaje se tome de forma al azar la correspondiente porción de dicha muestra de valores para el conjunto de entrenamiento, y el resto para el conjunto de testeo.

Se tomó en consideración el hecho de entrenar a la red siempre con los valores más representativos de la función, ya sean los máximos y los mínimos para que dichos valores queden siempre contemplados en el entrenamiento, pero decidimos no implementar dicha idea para ajustarnos a la realidad de tomar puntos de entrada de forma aleatoria sin tener el tiempo de analizarla por completo, tal como sería en un caso real.

Mejoras implementadas

Se realizaron dos (2) mejoras al algoritmo de *backpropagation* siendo las mismas *momentum* y *eta adaptativo*.

Momentum

Se implementó *momentum* con la siguiente fórmula:

$$\Delta w_{pq}(t+1) = -\eta * (\partial E / \partial w_{pq}) + \alpha * \Delta w_{pq}(t)$$

Eta adaptativo

Tal como se nos dijo en clase, tener un eta constante puede ser favorable para el aprendizaje de la red en ciertos momentos, pero en otros puede ser contraproducente, es por esto que ésta mejora introduce la variación del eta a medida que cambia el error actual con el previo (ΔE), respetando la siguiente condición:

$$\Delta\eta = \begin{cases} +a & si \ \ \Delta E < 0 \ \ consistentemente \\ -b\eta & si \ \ \Delta E > 0 \\ 0 & en \ otro \ caso \end{cases}$$

Resultados de las diferentes arquitecturas

Incremental (sin mejoras)

La arquitectura elegida es el resultado de diversas pruebas que se realizaron variando los distintos parámetros de la red hasta encontrar una configuración que minimice el error en un tiempo razonable. Particularmente se empezó fijando todos los parámetros en un valor arbitrario y se comenzó variando la cantidad de capas ocultas y la cantidad de neuronas por capa. La configuración de capas de la red con la cual se obtuvieron los mejores resultados es la siguiente: 3 capas ocultas con la siguiente cantidad de neuronas por capa respectivamente: 6, 5 y 4. Ver [Tabla 1, filas 1-4]

Luego de conseguir la mejor combinación de cantidad de capas ocultas y neuronas por capa se dejó fijo dicho parámetro y se pasó a variar la funcion de activacion. La diferencia entre los resultados obtenidos utilizando la tangente hiperbólica como función de activación en vez de la función exponencial fue abismal. No se logro hacer que la red aprenda el terreno utilizando la función exponencial, por ende se optó por utilizar la tangente hiperbólica. Ver [Tabla 1, filas 4-5]

El siguiente parámetro a variar fue el valor de beta. Empíricamente se demostró que a menor valor de beta, mayor dificultad para aprender tiene la red, pero a mayor valor de beta más tardaria la red en aprender. El valor para beta que se consideró más cercano al óptimo es 0.75. Ver [Tabla 1, filas 5-8]

Luego se analizó el efecto que tiene variar el valor del eta. Los resultados indicaron que a menor eta más tardaria la red en aprender, pero lo haría con una mayor probabilidad de encontrar un error mínimo. Para valores más grandes del eta el entrenamiento de la red avanza de a pasos más grandes, pero estos pasos pueden hacer que la red no pueda encontrar el error mínimo. El valor para el eta que se consideró el más balanceado es 0.1. Ver [Tabla 1, filas 6, 10-11]

Por último se pasó a analizar la capacidad de generalización que tiene la red con esta configuración. Se realizaron pruebas entregando a la red el 100%, 75%, 50% y 25% de la información total. Incluso en el caso de 25% la red logró aprender el 84% de los puntos correctamente. Ver [Tabla 1, filas 6, 12-14]

Momentum

Analizando el 100% de los patrones de entrada se llegó a una configuración óptima utilizando el valor de alpha sugerido en libros 0.9 y un eta de 0.01. Obteniendo resultados muy buenos en pocas épocas 800 con un 99% de los patrones aprendidos (con un error del 5%). Esto se debe a que con un alpha de 0.9 se le da mucha importancia al peso anterior, lo cual es bueno ya que estamos entrenando la red con el 100% de los patrones y no hay puntos generalizados.

Se puede ver que para etas grandes por ejemplo 0.1 a la red le cuesta más aprender y los resultados no son estables, es decir que pueden variar mucho segun la epoca. Por otro lado para etas más chicos como 0.001, se avanza muy lento y no se logra aprender lo deseado.

Luego se analizó estos mismos parámetros para el 50% de los inputs y se noto que si bien los errores eran similares, el porcentaje de patrones y generalizaciones aprendidas fue menor llegando a un 70-75% y los resultados también varían. Se dedujo que esto sucede por tener un alpha grande, con lo cual se le está dando mucha importancia a los pesos de puntos generalizados lo cual no es muy conveniente. Por lo tanto para este caso se redujo el alpha a un 0.5, obteniendo resultados más favorables. (ver tabla 4)

Por último para el 25% de los inputs, sucedió lo mismo que para el caso anterior (50% de los inputs) y se tuvo que modificar alpha a un 0.35.

Eta adaptativo

Para encontrar la configuración óptima con la siguiente mejora se procedió a variar los parámetros a, b y k, fijando los valores óptimos obtenidos para el incremental

sin mejoras, alimentado la red con un cien por ciento de los inputs a aprender para poder así sacar conclusiones más precisas.

Lo primero que se notó fue que al hacer variar el valor de *a* por valores cada vez mayores se obtuvo que el error mínimo de entrenamiento cada vez es más grande, y esto se llegó a la conclusión de que se debe a que lo que hace *a* es aumentar el eta, por lo que al aumentarlo se pierde precisión en la aproximación, por lo tanto se optó por elegir un *a* pequeño, siendo el mismo de 0.1, ya que a valores más pequeños que 0.1 el error mínimo de entrenamiento aumentaba (Ver Tabla 5).

Luego para el valor de *b*, se pudo apreciar que mientras más alto el valor, mayor el error mínimo de entrenamiento, y a su vez que el porcentaje aprendido se hacía cada vez menor (Ver Tabla 3). Es por esto que el valor óptimo elegido fue de 0.05.

Algo parecido pasó con el valor de *k*, que mientras éste subía aumentaba el error mínimo, a pesar de que el porcentaje aprendido no variaba mucho entre los distintos valores. (Ver Tabla 2). Es por esto que el valor óptimo elegido fue de 2.

Conclusiones

- La configuración óptima de la red es particular del problema a resolver. Se intentó aprender el terreno de otro grupo con nuestra configuración de la red y notamos que la red tenía ciertas dificultades para aprenderlo. Al cambiar de configuración a la configuración utilizada por aquel grupo la red pudo aprender el terreno sin mayor problema.
- La cantidad de neuronas por capa funciona mejor si se encuentra de forma descendente. Por ejemplo, una configuración de 3 capas ocultas con 7, 5 y 3 neuronas respectivamente va a aprender mucho mejor que una configuración de 3 capas ocultas con 3, 5 y 7 respectivamente.
- En el caso de momentum, la configuración óptima no se mantiene para diferentes porcentajes de inputs. Es conveniente disminuir alpha óptimo si el porcentaje de inputs no es el 100%, en otras palabras alpha está directamente relacionado al porcentaje de inputs, si el porcentaje baja alpha también y si el porcentaje sube alpha también.
- Siguiendo con momentum, es importante notar que este método necesita un eta bajo para poder operar sin problemas.

Anexo

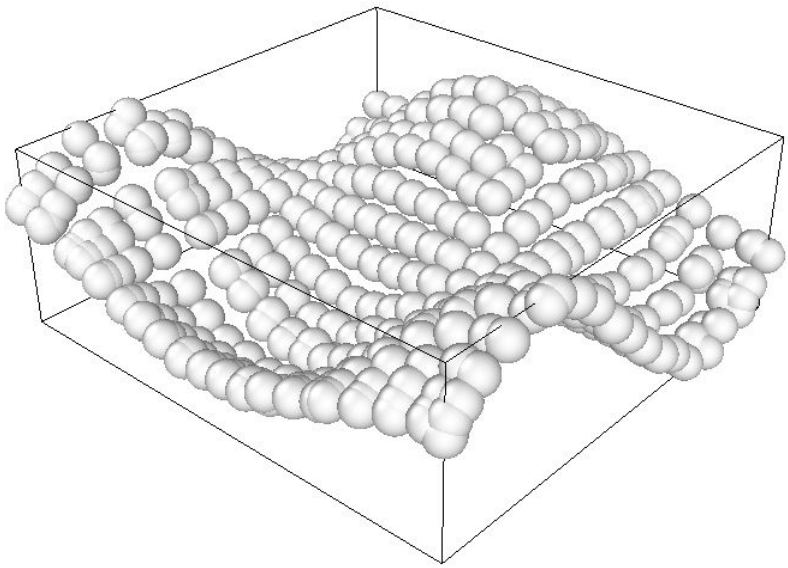

Figura 1

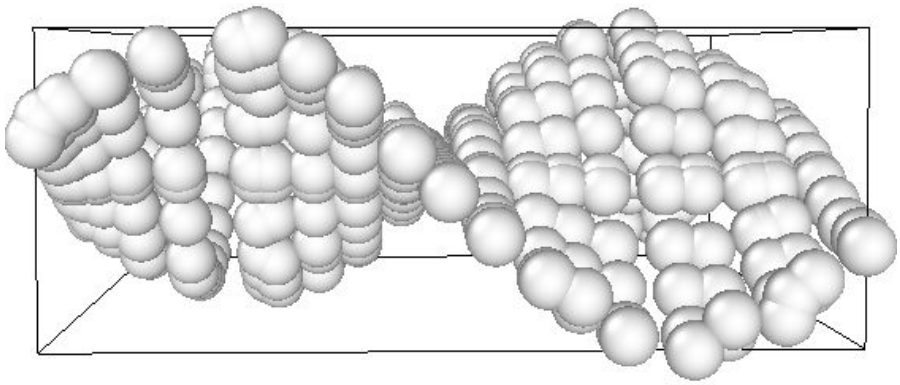

Figura 2

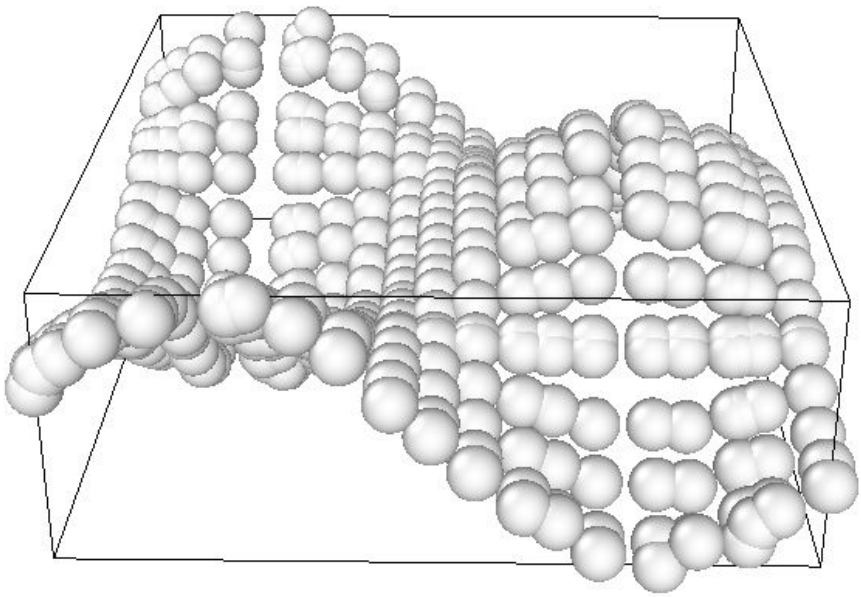

Figura 3

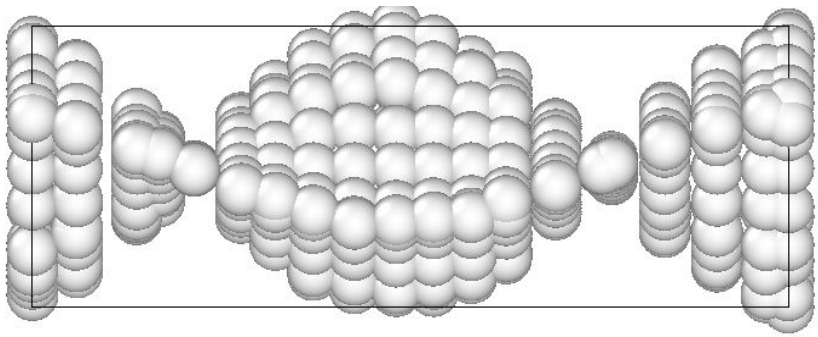

Figura 4

fila	Tipo	Estructura de la red	% de inputs	Función activación	etha	betha	alpha	épocas	Error mínimo de entrenamiento	% aprendido
1	normal	[2 2 1]	100	tanh	0.1	0.5	0	750	0.0494377	23
2	normal	[2 2 2 1]	100	tanh	0.1	0.5	0	900	0.0409222	31
3	normal	[6 5 1]	100	tanh	0.1	0.5	0	3150	0.000938275	95
4	normal	[6 5 4 1]	100	tanh	0.1	0.5	0	1649	0.000494431	98
5	normal	[6 5 4 1]	100	exp	0.1	0.5	0	600	0.107839	20
6	normal	[6 5 4 1]	100	tanh	0.1	0.75	0	690	0.000471692	99
7	normal	[6 5 4 1]	100	tanh	0.1	0.25	0	900	0.00344382	76
8	normal	[6 5 4 1]	100	tanh	0.1	0.875	0	767	0.000485247	99
9	normal	[7 1]	100	tanh	0.1	0.75	0	1800	0.00134028	95
10	normal	[6 5 4 1]	100	tanh	0.5	0.75	0	900	0.00172682	87
11	normal	[6 5 4 1]	100	tanh	0.05	0.75	0	4244	0.000499474	99
12	normal	[6 5 4 1]	75	tanh	0.1	0.75	0	1030	0.000486636	94
13	normal	[6 5 4 1]	50	tanh	0.1	0.75	0	2185	0.000491268	90
14	normal	[6 5 4 1]	25	tanh	0.1	0.75	0	1676	0.000491595	84
15	normal	[6 5 1]	75	tanh	0.1	0.75	0	1200	0.00135565	85
16	normal	[4 5 6 1]	100	tanh	0.1	0.75	0	900	0.00113088	96

Tabla 1

Estructura de la red	% de inputs	Función activación	eta	beta	a	b	k épocas	épocas	Error mínimo de entrenamiento	% aprendido
[6 5 4 1]	100	tanh	0.1	0.75	0.01	0.01	2	1900	0.000341132	99,7732
[6 5 4 1]	100	tanh	0.1	0.75	0.01	0.01	4	2400	0.000747129	97,7324
[6 5 4 1]	100	tanh	0.1	0.75	0.01	0.01	5	2400	0.00074476	97,3197
[6 5 4 1]	100	tanh	0.1	0.75	0.01	0.01	6	2400	0.000624499	98,4127

Tabla 2

Estructura de la red	% de inputs	Función activación	eta	beta	a	b	k épocas	épocas	Error mínimo de entrenamiento	% aprendido
[6 5 4 1]	100	tanh	0.1	0.75	0.2	0.05	2	1600	0.000331447	100
[6 5 4 1]	100	tanh	0.1	0.75	0.2	0.1	2	1400	0.000640667	98,1859
[6 5 4 1]	100	tanh	0.1	0.75	0.2	0.2	2	1300	0.000996232	95,4649
[6 5 4 1]	100	tanh	0.1	0.75	0.2	0.3	2	2100	0.00108314	90,9297
[6 5 4 1]	100	tanh	0.1	0.75	0.2	0.4	2	2200	0.00169906	92,0635

Tabla 3

Tipo	Estructura de la red	% de inputs	Función activación	etha	betha	alpha	épocas	Error mínimo de entrenamiento	% aprendido
momentum	[6 7 7 1]	100%	tanh	0.02	0.8	0.4	700	0.000660157	97%
momentum	[6 5 4 1]	100%	tanh	0.01	0.75	0.9	800	0.000538703	99%
momentum	[6 7 7 1]	50%	tanh	0.02	0.4	0.8	2200	0.00251108	68%
momentum	[6 5 4 1]	50%	tanh	0.01	0.75	0.9	4300	0.00032502	70%
momentum	[6 5 4 1]	50%	tanh	0.01	0.75	0.6	3300	0.000857664	86%
momentum	[6 5 4 1]	50%	tanh	0.01	0.75	0.5	3000	0.000890983	91%
momentum	[6 5 4 1]	30%	tanh	0.1	0.2	0.9	2100	0.00124841	75%
momentum	[6 5 4 1]	25%	tanh	0.01	0.75	0.35	1600	0.00193145	78%
momentum	[6 5 4 1]	25%	tanh	0.01	0.75	0.35	6300	0.000921665	90%

Tabla 4

Estructura de la red	% de inputs	Función activación	eta	beta	a	b	k épocas	épocas	Error mínimo	% aprendido
[6 5 4 1]	25	tanh	0.1	0.75	0.01	0.05	2	2300	0.00258682	78,0045
[6 5 4 1]	25	tanh	0.1	0.75	0.1	0.05	2	2300	0.000465755	86,5215
[6 5 4 1]	25	tanh	0.1	0.75	0.2	0.05	2	2300	0.000608095	87,0748
[6 5 4 1]	25	tanh	0.1	0.75	0.3	0.05	2	2300	0.000598901	81,8594
[6 5 4 1]	25	tanh	0.1	0.75	0.4	0.05	2	2300	0.00519381	75,9637

Tabla 5