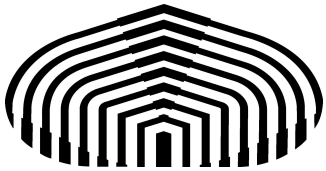


Universidad Simón Bolívar

Departamento de Computación y Tecnología de la Información

Inteligencia Artificial II

Informe del Proyecto 1

Gabriel Álvarez 09-10029
Francisco Martinez 09-10502

Sartenejas, Febrero de 2016

Título

"Clasificando patrones con redes neuronales"

Resumen

Este trabajo tiene como objetivo la implementación y prueba del algoritmo de Backpropagation para dos tipos de problemas distintos, uno de estos es la clasificación de puntos que se encuentran en una área cuadrada delimitada desde el centro del plano cartesiano, con cada lado teniendo 20 de longitud. En este problema se realizó un entrenamiento con 2000 puntos del set dado por la profesora de 2 a 10 neuronas y con la mejor configuración resultante de esta corrida se realizaron otras corridas con los conjuntos de datos restantes, posteriormente se realizó lo mismo para el conjunto de datos generados por nosotros mismos, los cuales en comparación a los hechos por la profesora tienen una distribución de mitad y mitad de puntos fuera y dentro del círculo. Los resultados de estos experimentos son comparados entre sí para hallar la mejor respuesta y luego se corre una verificación con un barrido de 10 mil puntos en la misma área del cuadrado.

El otro problema consiste en la clasificación de flores, en particular, las flores iris: Iris-Setosa, Iris-Versicolor y Iris-Virginica. Se utilizó el conjunto de datos creado por Ronald Fisher. Este problema fue subdividido en dos: en el primer caso se utilizó una clasificación binaria que separa las "Iris-Setosa" del resto y el segundo caso que se encarga de clasificar los tres tipos de flores.

Detalles de implementación y experimentación

El algoritmo de BackPropagation es un tipo de aprendizaje supervisado el cual aprende de varias entradas, este algoritmo comienza con los pesos inicializados en un valor aleatorio, pero utilizando esta fórmula para el rango de valores posibles sqrt(6/(cantidad entradas + cantidad de neuronas))), el algoritmo posteriormente se encarga de alimentar las entradas a través de las neuronas en la capa oculta la cual utilizan la función logistica de manera tal que por el teorema de aproximación universal permite a la red neural crear una convergencia o aproximación hacia los valores de la función mencionada, esto introduce la no-linealidad en el algoritmo, además de que permite tener una derivada de la función sencilla para el proceso derivacional que se realiza. Posteriormente se calcula una comparación entre la respuesta del ejemplo dado con la respuesta calculada por la red neural, con este error y a través de varias derivadas parciales (explicación que utilizamos para escribir y guiarnos en la implementación del algoritmo) se ajustan los pesos que posee la red entre la capa de entradas y las neuronas al igual como los pesos entre las neuronas y las salidas, de manera tal que se disminuya el error actual en la red, este proceso es repetido por cada caso de entrenamiento disminuyendo el error cada iteración.

Se debe mencionar que los datos de ambos problemas fueron normalizados mediante este método y que las salidas fueron normalizadas a 0 y 1. En el caso del segundo problema, para la clasificación binaria se asigna: 0 a Iris-Setosa y 1 a las otras dos; para la clasificación ternaria se asignó: 1 para Iris-Setosa, 0.5 para Iris-Versicolor y 0 para Iris-Virginica.

Discusión y presentación de los resultados

Estas corridas fueron realizadas usando 10000 iteraciones.

Ejercicio 2

Set generado aleatoriamente

En este caso la configuración que dio menor error fue la de 2 neuronas y constante de aprendizaje 0,01. A continuación se muestran los resultados en cada conjunto:

	500	1000	2000
Error Total	61.7988811402	126.022502205	253.256196997
Tiempo	0.249000072479	0.265000104904	0.265000104904
Error Promedio	0.12359776228	0.126022502205	0.126628098499

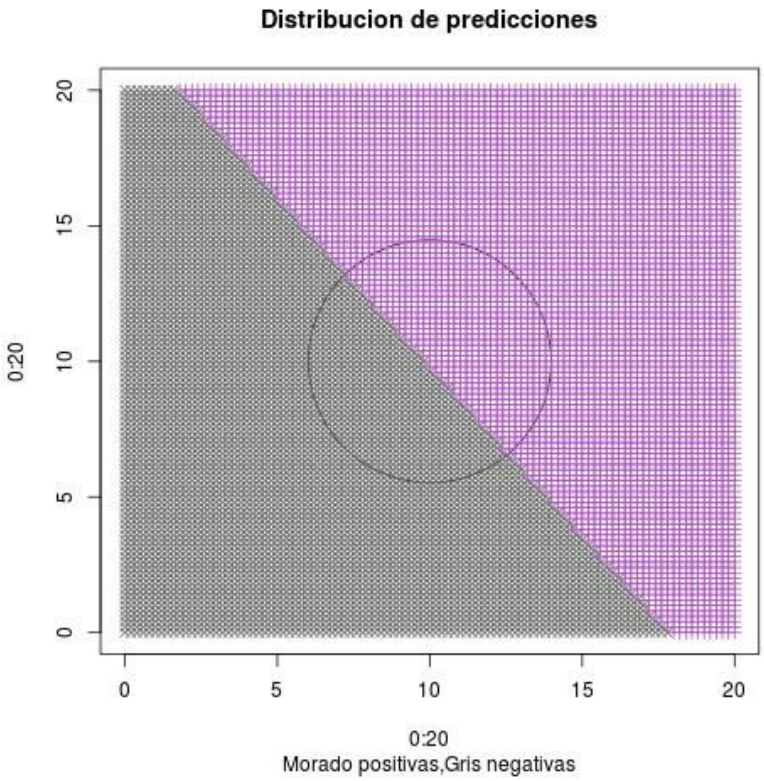
Set dado en el problema

En este caso la configuración que dio menor error fue la de 2 neuronas,y constante de aprendizaje 0,3. A continuación se muestran los resultados en cada conjunto:

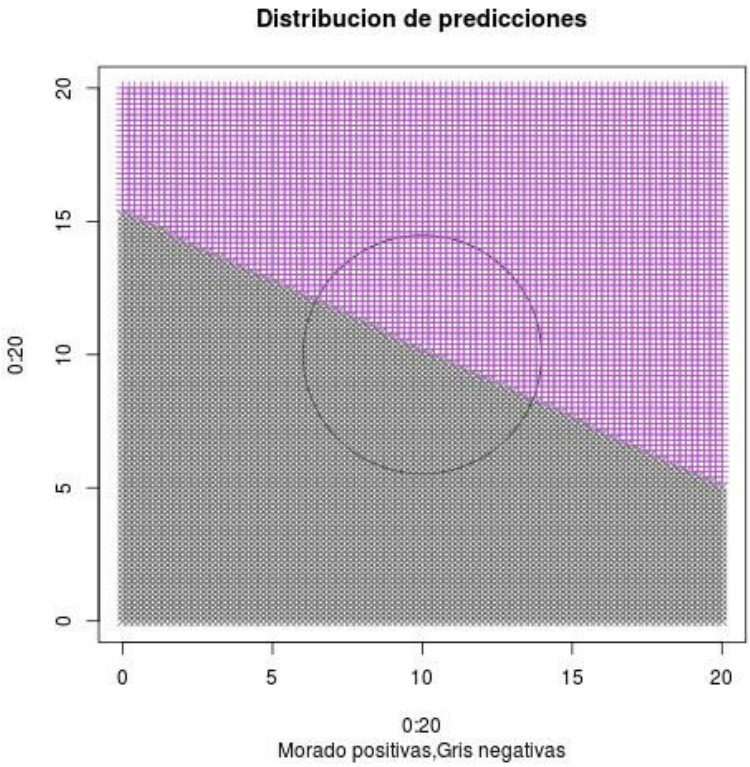
	500	1000	2000
Error Total	60.1673676927	118.732468388	234.020677279
Tiempo	0.265000104904	0.25	0.266000032425
Error Promedio	0.120334735385	0.118732468388	0.11701033864

Se puede observar que el caso en el que el error fue menor fue en el caso de los datos dados por la profesora, en los cuales el error promedio dio 0.1170103 aunque la diferencia fue poca al comparar con el mejor caso de nuestros ejemplos generados, la causa de esto debe ser la distribución de los puntos en el espacio ya que esto es lo que difiere en las diferentes conjuntos de entradas aleatorias, los datos que son distribuidos de manera uniforme en el cuadrado son los que realizaran una mejor predicción ya que en la distribución equitativa de puntos(nuestra generación de datos) la relación entre las cantidades no es análoga a la relación entre las áreas de las figuras, varios de estos puntos se aglomeraron en el área del círculo causando que el algoritmo haga más predicciones donde el resultado sea dentro del círculo. Se puede también observar otra tendencia que es el incremento del error a medida que los puntos aumentaron considerablemente, esto puede observarse como el overfitting, en particular esto sucede en el caso de los datos generados aleatoriamente ya que el algoritmo aprende en cada iteración del ruido generado por la mala distribución de los datos. Mientras que en los datos suministrados, al ser una mejor representación del problema el error disminuye pues aprende mejor con más puntos.

Se probó el porcentaje de predicción del algoritmo con un barrido del área en 10 mil puntos distintos, donde el algoritmo realizó una predicción representada por la siguiente gráfica en los datos generados por el equipo:



En contraste, ahora se muestran las predicciones del conjunto de datos suministrados :



Donde los puntos morados representan los que el algoritmo dijo que estaban dentro del círculo, mientras que los grises fueron los que quedaron fuera. Se puede ver que el algoritmo falla en gran proporción(~45%), pero fallamos en manipular las diferentes variables del algoritmo para mejorarlo. Consideramos que puede deberse a una mala normalización ya que para los resultados del XOR(entradas para verificar la correctitud del algoritmo) y de la Iris Setosa fueron satisfactorios.

Ejercicio 3
El algoritmo de backpropagation fue probado con el 50%, el 60%, el 70%, el 80%y el 90% del conjunto de datos, estos conjuntos fueron generados manualmente y con un número igual de flores. Además el algoritmo fue probado con diferentes números de neuronas para la capa oculta, en el rango [4,10].

Clasificador binario:

	50%(75)	60%(90)	70%(105)	80%(120)	90%(135)
Error Total	0.00888758	0.012725305	0.012320742	0.014895546	0.017376802
Error Promedio	0.00011850	0.000141392	0.000117340	0.000124129	0.000128717
Tiempo	0.57700014	0.530999898	0.529999971	0.576999902	0.592999935
Neuronas	5	4	4	5	5
Constante de aprendizaje	0.5	0.5	0.5	0.5	0.5

Clasificador Ternario:

	50%(75)	60%(90)	70%(105)	80%(120)	90%(135)
Error Total	0.33690846	0.458412749	0.639463510	0.688758199	0.730657634
Error Promedio	0.00449211	0.005093474	0.006090128	0.005739651	0.005412278
Tiempo	0.52999997	0.531000137	0.576999902	0.576999902	0.578000068
Neuronas	4	4	5	5	5
Constante de aprendizaje	0.5	0.5	0.2	0.3	0.2

Se puede observar que para ambos clasificadores, el número de neuronas que obtuvo mejores resultados fue entre 4 y 5. Y comparando los resultados entre ambos clasificadores, se puede ver que el clasificador binario obtuvo mejores resultados (en general) que el clasificador ternario. Además en el clasificador binario los mejores resultados fueron obtenidos con el 70% de los datos, también en el clasificador ternario fue en el 50% de los datos. Cabe destacar que varias corridas fueron realizadas, en las que, a veces, el mejor resultado estaba en 4 neuronas o 5 neuronas, por la inicialización random de los pesos. Además en el clasificador binario, los mejores resultados fueron con la constante de aprendizaje 0.5, mientras que en el ternario fue un poco más variada.