Red neuronal básica

Pablo Rodríguez Quesada. paroque28@gmail.com
Oscar Ulate Alpízar. oscarjosue94@gmail.com

***Resumen*—The goal of this project is to learn the basic implementation of a basic neural network with one intermediate layer. The intention is to design and develop the complete neural network from scratch, without using any library for the matter.**

I. Introducción

Este proyecto tiene como intención crear desde cero una red neuronal que se encargue de clasificar una serie de datos generados aleatoriamente en tres clases. Estas clases están dadas por la posición de los datos en un plano de dos dimensiones. Existen cuatro tipos de clasificaciones posibles para los datos: vertical, horizontal, anillo y pastel. La red neuronal debe ser capaz de clasificar cada uno de los datos para cualquiera de las clasificaciones posibles antes mencionadas.

La red neuronal cuenta con dos capas. La primera capa, que consta de 7 neuronas se encarga clasificar de manera no lineal a los datos. Por último, la capa de salida, con tres neuronas, brinda la salida del sistema, con los datos ya clasificados.

El método de aprendizaje utilizado es el de descenso de gradiente (*back propagation*). Este funciona calculando el error en la salida de la red neuronal y de esta manera reajusta los pesos de entrada a las neuronas de la red.

II. Descenso de gradiente

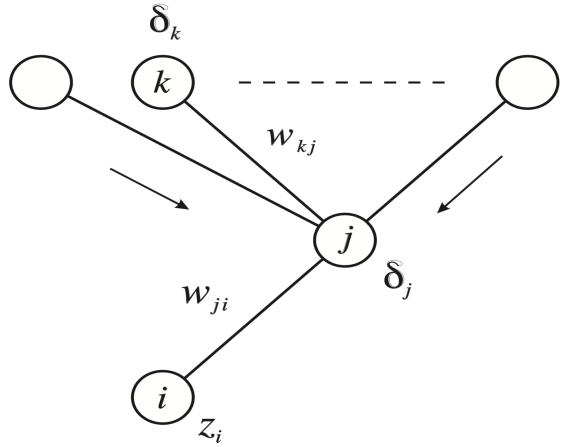

Figura 1. Notación usada por Bishop.

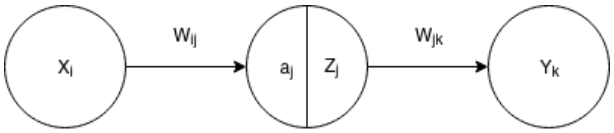

Figura 2. Esquema de red neuronal.

El descenso de gradiente se realizó basado en el artículo publicado por Bishop [1]. En la *Figura 1* y *Figura 2* se muestra la notación usada por el autor. Con esta notación ya es sencillo explicar el descenso de gradiente.

Partiendo de que:

$$z_j = g(a_j), a_j = \sum_i w_{ji} z_i \tag{1}$$

El error E^n se puede calcular como:

$$\frac{E^n}{\partial w_{ji}} = \frac{E^n}{\partial a_j} \frac{a_j}{\partial w_{ji}} \tag{2}$$

Con:

$$\delta_j \equiv \frac{E^n}{\partial a_j} \tag{3}$$

Para las neuronas de salida, δ_k:

$$\delta_k \equiv \frac{E^n}{\partial a_k} = g'(a_k) \frac{E^n}{\partial y_k} \tag{4}$$

Por lo que:

$$\delta_k = y_k^n - t_k^n \tag{5}$$

Donde t_k^n es el resultado de la función de predicción implementada en el proyecto.

Con lo anterior se llega a la ecuación final del *back propagation*:

$$\delta_j = g'(a_j) \sum_k w_{kj} \delta_k \tag{6}$$

Utilizando esta ecuación final se logra ejecutar el algoritmo de descenso de gradiente para de esta manera, recalcular los pesos w_{ji} y w_{kj} para que la red logre disminuir el error y predecir mejor las muestras en el futuro.

III. Entrenamiento de la red neuronal

Como se menciona en la introducción, la red está compuesta por tres capas de profundidad, la primera capa es la de entrada con dos neuronas correspondientes a las coordenadas en un plano bidimensional. La segunda capa posee cuatro neuronas. Esta cantidad de neuronas fue escogida arbitrariamente puesto que lleva a la red a converger de manera rápida relativo a otras cantidades de neuronas. La capa final contiene una cantidad de neuronas según la cantidad de características posibles, en este caso tres.

La función de parada está determinada por *threshold*, una variable que se encarga de verificar el cambio del error en el descenso por gradiente. Cuando el cambio del error entre iteraciones es menor que el *threshold*, el algoritmo para su ejecución. Esta variable tiene un valor por defecto de $0,001$.

El entrenamiento que se utiliza es estocástico, esto quiere decir que no se utilizan todas las mil muestras de datos para realizar el descenso de gradiente, sino que se utiliza solo 1 muestra a la vez. Fue escogido así por la velocidad en que converge el algoritmo.

Cuadro I
Entrenamiento de la red neuronal con un total de 1000 datos

α	**J**	**Iteraciones**
$0,01$	$12,377$	87205
$0,05$	$10,241$	42901
$0,1$	$7,4064$	31435
$0,5$	$2,7729$	17948
1	$2,5728$	12665
5	$1,6631$	10357
10	$1,5683$	2733

Se utiliza un *learning rate* de 5 porque el de 10 aunque en promedio parece mejor, es más lento de ejecutar en algunas ocasiones.

IV. Evaluación de resultados

La matriz de confusión [2] contiene 3 filas y 3 columnas. Las 3 columnas representan las tres clases existentes predichas. Las 3 filas representan los datos en las tres clases reales de cada dato.

La matriz de confusión muestra los valores reales, no los normaliza para representar porcentajes.

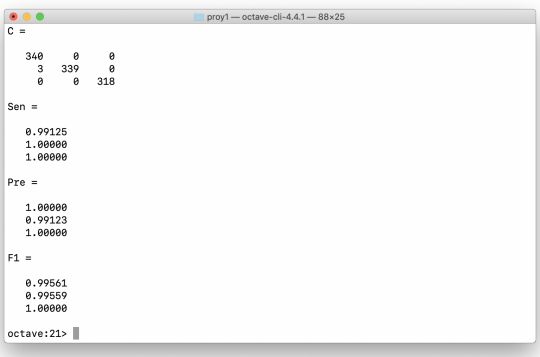

Figura 3. Datos de la matriz de confusión.

En la imagen anterior se muestra la matriz de confusión de la muestra de 1000 datos de ejemplo. El valor **C** es la matriz como tal, cada dato muestra los valores reales en las filas contra los valores clasificados por el algoritmo. El vector **Sen** muestra la sensitividad para cada clase. La precisión de cada clase se muestra en el vector **Pre** y por último en el vector **F1** para cada clase [3].

IV-A. *Resultado de predicciones*

En las siguientes imágenes se muestra el resultado de el muestreo de datos y en las imágenes de color sólido, el aprendizaje de la red neuronal. Se muestra tres posibles distribuciones: horizontal, pastel y radial. Estas muestras se realizaron con 1000 datos.

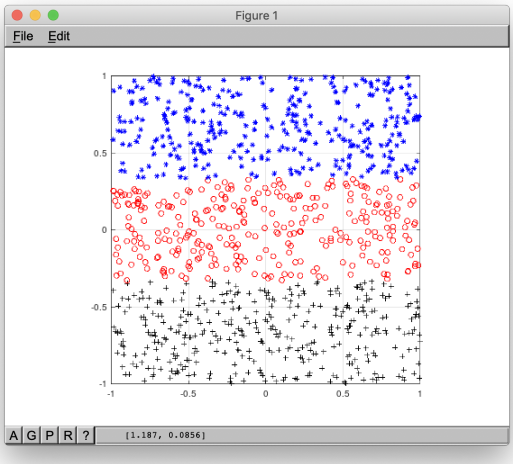

Figura 4. Distribución horizontal aleatoria de datos.

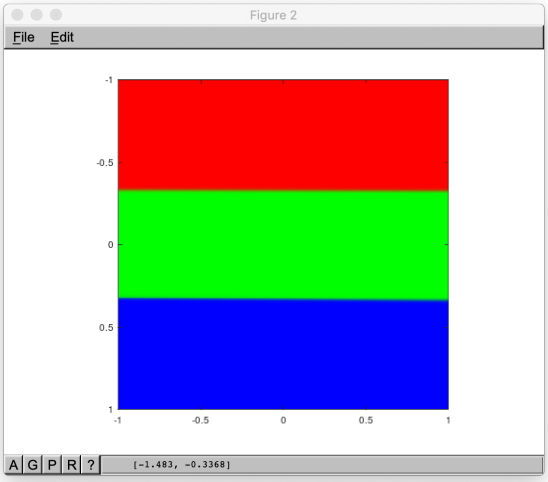

Figura 5. Clasificación horizontal.

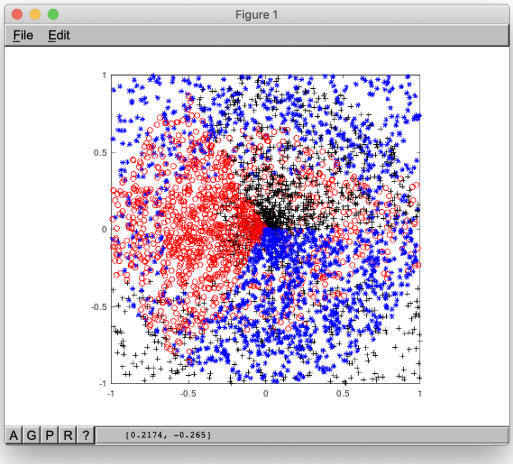

Figura 6. Distribución pastel aleatoria de datos.

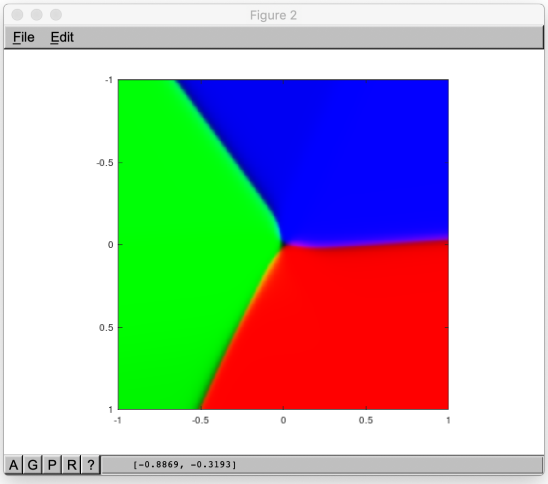

Figura 7. Clasificación pastel.

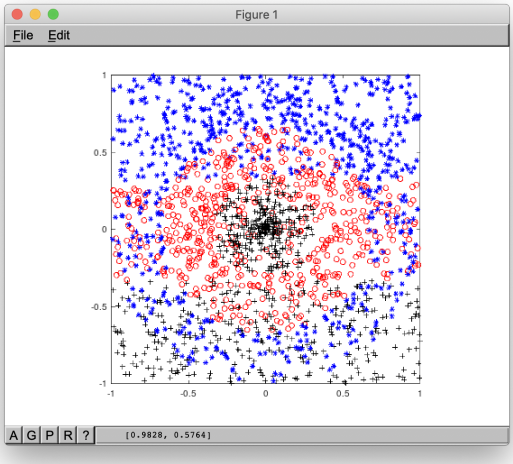

Figura 8. Distribución radial aleatoria de datos.

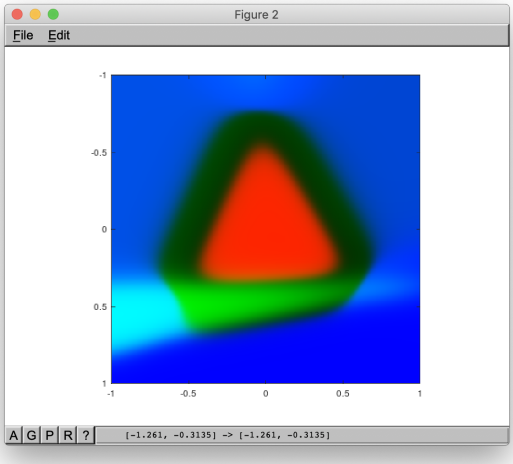

Figura 9. Clasificación radial.

V. Referencias

[1] Christopher Bishop. *Neural Networks: A Pattern Recognition Perspective*. 1996.

[2] scikit-learn. "Neural Networks: A Pattern Recognition Perspective". En: (2013).

[3] classeval. *Basic evaluation measures from the confusion matrix*. 2017. URL: https://classeval.wordpress.com/introduction/basic-evaluation-measures/.