UNIVERSIDAD TÉCNICA DEL NORTE
FICA-CIERCOM
PERCEPTRÓN SIMPLE EN ARDUINO

Velasco Angel

17 de febrero de 2020

1. Introducción

En el mundo existe cada vez una mayor necesidad de la tecnología en la vida cotidiana de las personas, esta se usa desde en un cuarto de estudio hasta en una cocina.
Una de las ramas de la ingeniería donde más se ha profundizado en investigación y experimentación es la inteligencia artificial (AI), esta se ha venido desarrollando a través de los años y se ha llegado a obtener resultados bastante acertados, hoy en día una máquina es capaz de reconocer rostros en una foto, o incluso saber qué hay en una imagen con un extenso algoritmo de machine learning detrás.

Con este propósito nacen las famosas redes neuronales, cuya función es tener la posibilidad de reconocer y clasificar objetos con el uso de ciertos aspectos determinantes de dicho objeto.

2. Marco Teórico

2.1. Sistema Neuronal Artificial

1888 Ramón y Cajal demuestra que el sistema nervioso está compuesto por una red de células individuales, las neuronas, ampliamente interconectadas entre sí [1].
La información fluye desde las dendritas hacia el axón atravesando el soma. El cerebro se modela durante el desarrollo de un ser vivo. Algunas cualidades del ser humano no son innatas, sino adquiridas por la influencia de la información que del medio externo se proporciona a sus sensores [1].

2.1.1. Modelo estándar de neurona artificial

Se va a introducir el denominado modelo estándar de neurona artificial según los principios descritos en Rumelhart y McClelland (1986) y McClelland y Rumelhart (1986). Siguiendo dichos principios, la i-ésima neurona artificial estándard consiste en:

* Un conjunto de entradas x_j y unos pesos sinápticos w_{ij}, con $j = 1, ..., n$.
* Una regla de propagación h_i definida a partir del conjunto de entradas y los pesos sinápticos.
* una función de activación la cual representa simultáneamente la salida de la neurona y su estado de activación. La Figura 4 muestra el modelo de neurona artificial estándar descrito previamente.

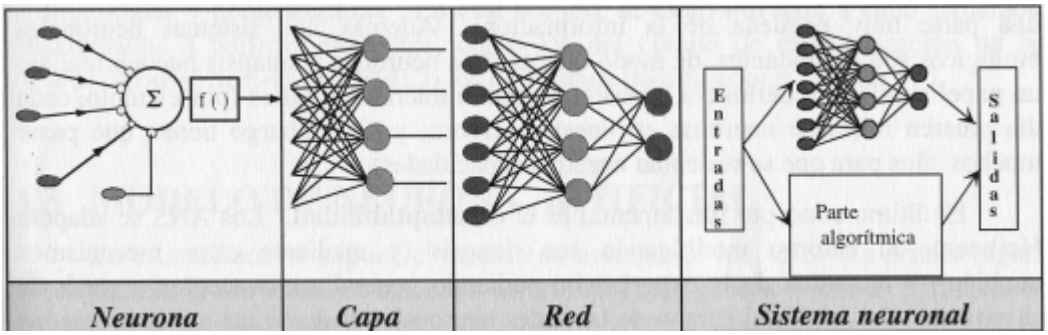


Figura 1: Sistema neuronal global de proceso de una red neuronal **Fuente:** sh.ehu.es

2.2. Perceptrón Simple

El perceptrón simple fue introducido por Rosenblatt (1962) y es un modelo unidireccional compuesto por dos capas de neuronas, una de entrada y otra de salida. La operación en un perceptrón simple que consta de n neuronas de entrada y m neuronas de salida, se puede expresar como:

$$y_i = f\left(\sum_{j=1}^{m} w_{ij}x_j - \theta_i\right) \tag{1}$$

con $i = 1, ..., m$.

Las neuronas de entrada son discretas y la función de activación de las neuronas de la capa de salida es de tipo escalón, Como se observa en la Figura 2.

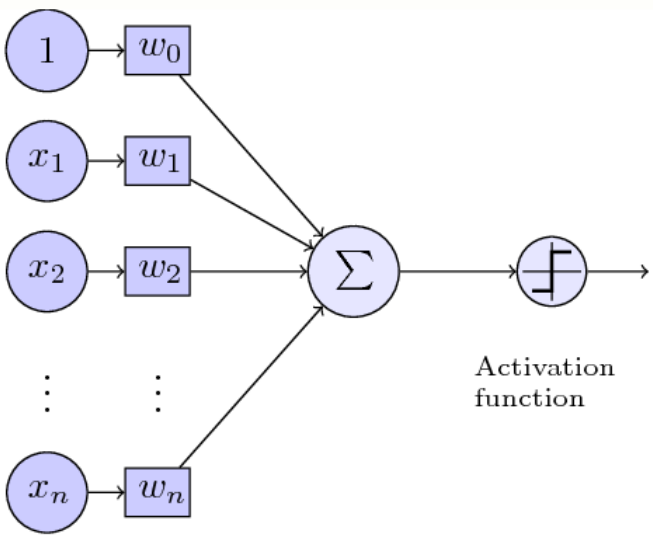


Figura 2: Diagrama de Perceptrón **Fuente:** https://tex.stackexchange.com/questions/104334/tikz-diagram-of-a-perceptron

Veamos con un ejemplo sencillo, que contiene dos neuronas de entrada, que el perceptrón simple tan solo puede discriminar entre dos clases linealmente separables, es decir, clases cuyas regiones de decisión pueden ser separadas mediante una única condición lineal o hiperplano [2].
Si denotamos por $x1$ y $x2$ a las dos neuronas de entrada, la operación efectuada por el perceptrón simple consiste en:

$$y = \begin{cases} 1, si & w_1x_1 + w_2x_2 > \theta \\ 0, si & w_1x_1 + w_2x_2 \leqslant \theta \end{cases} \tag{2}$$

Si consideramos x_1 y x_2 situadas sobre los ejes de abcisas y ordenadas respectivamente, la condición

$$w_1x_1 + w_2x_2 - \theta = 0 \tag{3}$$

es equivalente a:

$$x_2 = -\frac{w_1}{w_2}x_1 + \frac{\theta}{w_2} \tag{4}$$

y representa una recta (Figura 3) que define la región de decisión determinada por el perceptrón simple. Es por ello que dicho perceptrón simple representa un discriminador lineal, al implementar una condición lineal que separa dos regiones en el espacio que representan dos clases diferentes de patrones [2].

Por tanto, el perceptrón simple presenta grandes limitaciones, ya que tan sólo es capaz de representar funciones linealmente separables.

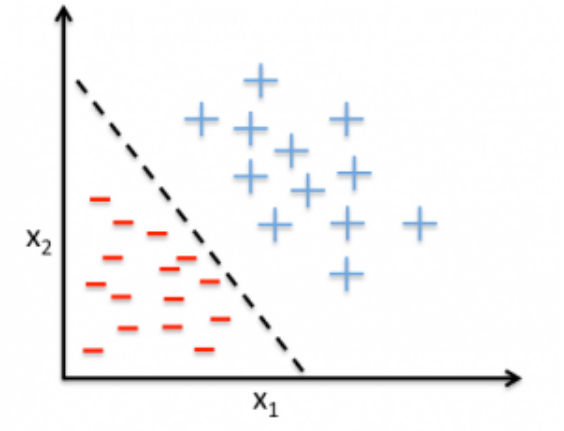


Figura 3: Región de decisión de Perceptrón Simple **Fuente:** https://geeks.ms/mrcabellom/2017/08/04/perceptron-binario-en-machine-learning/

2.3. Red Neuronal Artificial

Se puede definir una red neuronal artificial como un grafo dirigido (Figura 4), con las siguientes propiedades:

1. A cada nodo (neurona) i se le asocia una variable de estado X_i.
2. A cada conexión (i, j) entre los nodos (neuronas) i y j se le asocia un peso $w_(ij) \in \mathbb{R}$.
3. A cada nodo (neurona) i se le asocia un umbral $\theta_i \in \mathbb{R}$.
4. Para cada nodo i se define una función $f_i(x_i, ..., x_n, w_{i1}, ..., w_{in}, \theta_i)$ que depende de los pesos de sus conexiones, del umbral y de los estados de los nodos j que estén conectados con el nodo i. El valor de esta función proporciona el nuevo estado del nodo.

Por lo que respecta a la terminología habitual en redes neuronales artificiales, tal y como se ha comentado previamente los nodos del grafo representan a las neuronas y las conexiones a las sinapsis. Se denominan neuronas de entrada a aquellas neuronas sin sinapsis entrantes. A las neuronas sin sinapsis salientes se las denomina neuronas de salida, y finalmente a aquellas neuronas que no son ni de entrada ni de salida se les denomina neuronas ocultas. Una red es unidireccional cuando no presenta bucles cerrados o conexiones, mientras que una red se dice recurrente o realimentada cuando el flujo de información puede tener un bucle de atras hacia adelante, es decir, una realimentación. En la relación con la manera en la que las neuronas de una red actualizan sus estados, podemos distinguir entre dinámica síncrona (en la cual todas las neuronas pertenecientes a una misma capa se actualizan a la vez, comenzando en la capa de entrada y continuando hasta la de salida) y dinámica asíncrona (en la cual cada neurona actualiza su estado sin atender a cuando lo hacen las demás neuronas). Si bien el tipo de dinámica presente en los sistemas neuronales biológicos es asíncrono, lo habitual en las redes neuronales artificiales es que la dinámica sea síncrona [1].

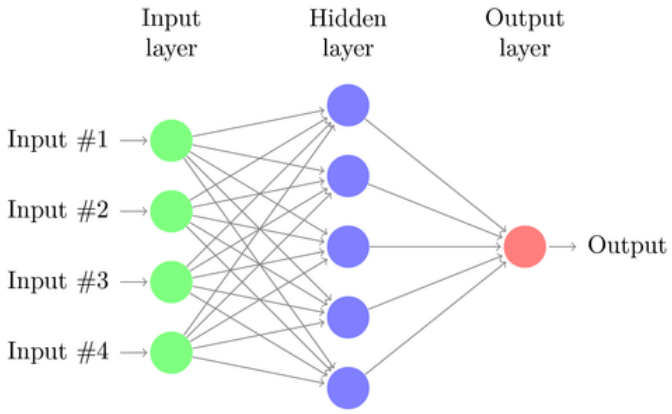


Figura 4: Partes de una red neuronal **Fuente:** http://www.texample.net/tikz/examples/neural-network/

3. Desarrollo

3.1. Perceptrón en Arduino

Para comenzar a programar nuestro perceptrón debemos tener en cuenta los siguientes aspectos:

- La base de datos a utilizar será la que tomamos en nuestro proyecto final (Ver Base de Datos).
- Esta base de datos cuenta con 300 datos (150 de etiqueta 1 y 150 de etiqueta 2) por 4 entradas y una salida de dos estados, ya que al tratarse de un preceptrón simple debemos trabajar con una ecuación lineal.
- Al tener 4 entradas tendremos también 4 pesos sinápticos y a su vez un umbral (θ o w_0 como se aprecia en la Figura 2).
- Por último hay que tomar en cuenta que las etiquetas que se tomaron en cuenta son 1 y 2, esto quiere decir que:

$$y = \begin{cases} 1, si & w_1x_1 + w_2x_2 > \theta \\ 2, si & w_1x_1 + w_2x_2 \leqslant \theta \end{cases} \tag{5}$$

Con estas premicias, pasamos a la programación de nuestra neurona como se muestra en el Programa 1.

Programa 1: Código Perceptrón Simple en Arduino

```
#include "base.h" //Llamar base de datos
float x[4], y; //Variables input y output
float peso[5] = {0.1, 0.1, 0.1, 0.1, -10}, pre_peso[5] , change[5]; //Arrays de peso actual y anterior
/* ARRAY PESO y PRE_PESO
   peso1  peso2 peso 3 peso 4 umbral

   ARRAY DE CAMBIOS (Determina si habrá cambios o no en el peso)
   cambio_w1 cambio_w2 cambio_w3 cambio W4 cambio_umbral

   ARRAY INPUTS
   x1 x2 x3 x4
*/
boolean sw = false; //Switch on off
int i = 0, j = 0; //Variables contador
int mistake;

void setup() {
  Serial.begin(250000); //Iniciar serial
}

void loop() {
  perceptron(300, 5, 0.005);
}

void perceptron(int fil, int col, float E) { //Función perceptrón, E=factor de aprendizaje
  if (!sw) {
    for (i = 0; i < fil; i++) {
      for (j = 0; j < col; j++) { //Llenar array de peso anterior con pesos actuales
        if (j < 4)
          x[j] = traning[i][j];
        pre_peso[j] = peso[j];
      }
      Serial.println("--------------------");
      Serial.print("Peso 1: ");
      Serial.println(peso[0]);
      Serial.print("Peso 2: ");
```

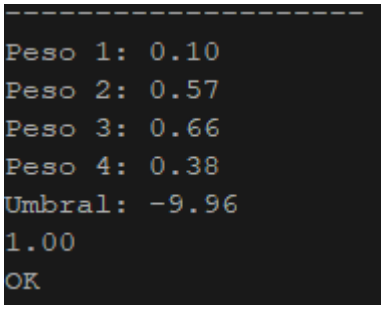
```
37        Serial.println(peso[1]);
38        Serial.print("Peso 3: ");
39        Serial.println(peso[2]);
40        Serial.print("Peso 4: ");
41        Serial.println(peso[3]);
42        Serial.print("Umbral: ");
43        Serial.println(peso[4]);
44       y = neurona(x, peso);
45       //       Serial.println(y);
46        Serial.println(funcion(y));
47        if (funcion(y) != traning[i][4]) { //Comparar estiqueta resultante y ↩
              real
48          mistake++;
49          Serial.println("F");
50          for (j = 0; j < col; j++) { //Corrección de pesos y Umbral en caso de ↩
                necesitarlo
51            if (j < 4) {
52              //error = resultado esperado - resultado obtenido
53              change[j] = E * x[j] * (traning[i][4] - y); //Cambio Pesos = E*(↩
                    error)*x
54            } else {
55              change[j] = E * (traning[i][4] - y); //Cambio Umbral = E*(error)
56            }
57            peso[j] = pre_peso[j] + change[j]; //Pesos = peso anterior + cambio
58          }
59        } else { //En caso de no necesitar corrección
60          Serial.println("OK");
61          for (j = 0; j < col; j++) {
62            change[j] = 0; //Cambio de 0
63            peso[j] = pre_peso[j] + change[j];
64          }
65        }
66      }
67      sw = true;
68      //PRESENTACIÓN DE RESULTADOS
69      Serial.println("-------------------------------------------------");
70      Serial.print("El número de predicciones fallidas es de: ");
71      Serial.println(mistake);
72      Serial.print("El número de predicciones acertadas es de: ");
73      Serial.println(300-mistake);
74      Serial.print("El perceptrón cuenta con un rendimiento del ");
75      Serial.print((300-float(mistake))*100/300);
76      Serial.print("%");
77    }
78  }
79
80  float neurona(float in[], float w[]) { //Función de neurona
81    return w[4] + in[0] * w[0] + in[1] * w[1] + in[2] * w[2] + in[3] * w[3]; //↩
        Sumatoria(Xi*Wi)+umbral
82  }
83
84  float funcion(float d) { //Función de activación
85    //  Serial.println(d);
86    if (d > 0) //Determinar etiqueta
87      return 1;
88    else
89      return 2;
90  }
```

El código se lo puede encontrar en el siguiente enlace de GitHub.

4. Resultados

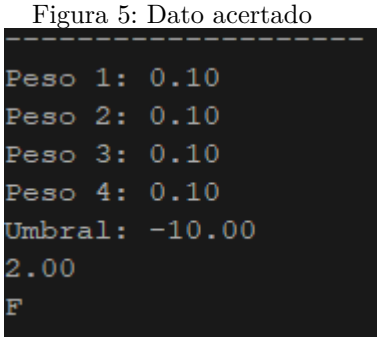
Los resultados obtenidos en la práctica son muy satisfactorios, ya que se ha obtenido un rendimiento que supera por completo los clasificadores probados anteriormente.

Primeramente, en la Figura 5 se presentan los datos donde la etiqueta predecida concuerda con la etiqueta original de los datos que se encuentran en la base. Asimismo, en la Figura 6 se presentan los datos donde la etiqueta predecida y la original no concuerdan entre sí.



7a

Figura 5: Dato acertado



7b

Figura 6: Dato erróneo

Figura 7: Presentación de datos

Por último, en la Figura 8 se presentan los resultados finales con el número de aciertos, número de errores y rendimiento expresado porcentualmente.

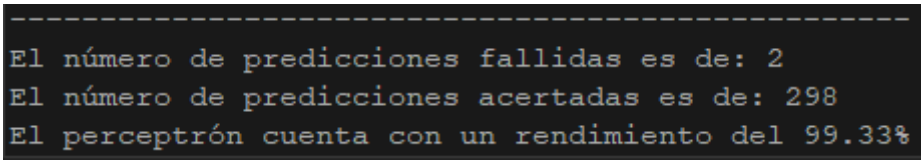


Figura 8: Resultados de rendimiento

5. Conclusiones y Recomendaciones

Conclusiones

- Un perceptrón es la unidad mínima de una red neuronal, con una sola neurona hemos sido capaces de crear un clasificador bastante eficiente, con una red neuronal compuesta por algunos perceptrones tiene muchos más cálculos detrás pero viene a ser un dispositivo muy robusto y útil.
- El factor de aprendizaje determina qué tan rápido nuestro algoritmo puede ser para aprender nuevos valores de peso para obtener una respuesta clara, si es muy bajo el prorama necesitará mayor número de iteraciones para llegar a un peso eficiente.
- El perceptrón de una capa, a pesar de su simplicidad ha logrado superar con creces al clasificador K-nn y el clasificador Bayesiano.

Recomendaciones

* Utilizar vectores o matrices en agrupaciones de datos para facilitar la programación.
* Realizar un esquema gráfico y matemático sobre cómo trabajará el algoritmo.
* Programar primeramente el algoritmo en un software más sencillo como lo es Excel, esto nos dará una idea más clara de los pasos que debemos seguir.

Referencias

[1] P. Larrañaga, "Tema 8. Redes Neuronales," Ph.D. dissertation, Universidad del Pa´ıs Vasco–Euskal Herriko Unibertsitatea. [Online]. Available: http://www.sc.ehu.es/ccwbayes/docencia/mmcc/docs/t8neuronales.pdf

[2] BIBING, "El Perceptrón," pp. 1–11, 2017.