Universidad de Guadalajara

Centro Universitario de Ciencias Exactas e Ingenierías

División de Electrónica y Computación

Departamento de Ciencias Computacionales

Ingeniería en Computación

Inteligencia Artificial II

Profesora: Arana Daniel, Nancy Guadalupe

I7040 – D01

Martes y Jueves 11:00 – 12:55

Actividad 01: Reporte de practica del perceptrón

Flores Camarena, Luis Manuel
214519661

Ojeda Escobar, César Arley
216306568

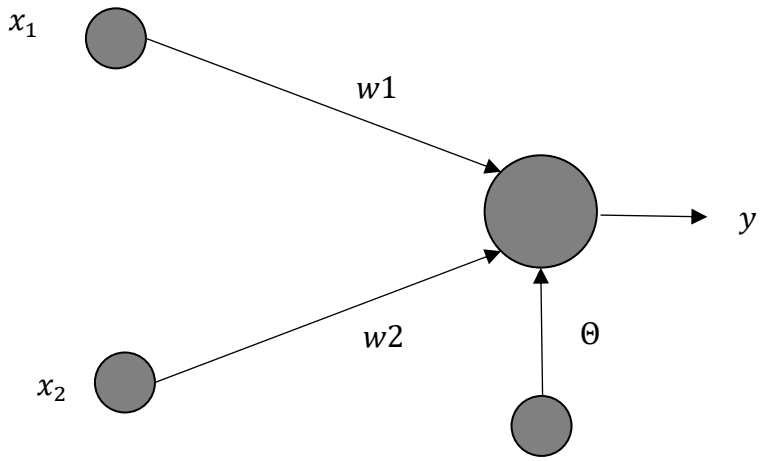
Fecha: 27/08/2019

El perceptrón

Este modelo se concibió como un sistema capaz de realizar tareas de clasificación de forma automática. La idea era disponer de un sistema que, a partir de un conjunto de ejemplos de clases diferentes, fuera capaz de determinar las ecuaciones de las superficies que hacían de frontera de dichas clases. La información sobre la que se basaba el sistema estaba constituida por los ejemplos existentes de las diferentes clases. Son dichos patrones de entrenamiento los que aportaban la información necesaria para que el sistema construyera las superficies discriminantes, y además actuara como un discriminador para ejemplos nuevos desconocidos. El sistema, al fina del proceso, era capaz de determinar, para cualquier ejemplo nuevo, a qué clase pertenecía.

Descripción del modelo

La arquitectura de la red es muy simple. Se trata de una estructura monocapa, en la que hay un conjunto de células de entrada, tantas como sea necesario, según los términos de problema; y una o varias células de salida, y son estas conexiones las que determinan las superficies de discriminación del sistema.



En la figura anterior las entradas son x_1 y x_2, y la salida, y. Los pesos son w_1 y w_2. Además, existe un parámetro adicional llamada umbral y denotado por Θ. El umbral se utiliza como factor de comparación para producir la salida, y habrá tantos como células de salida existan en la red, uno por cada una.

Código

Se implemento el código para ser ejecutado en una plataforma web, siendo el frontend desarrollado con ayuda del framework React y el backend utilizando Python y Flask.

Algoritmo del perceptrón en Python

Inicialización de la clase Perceptron

```
 def __init__(self, inputs, learningRate,
 expectedOutputs):
     self._inputs = inputs
     self._learningRate = learningRate
     self._expectedOutputs = expectedOutputs
     self._weights = np.random.rand(2)
     self._theta = np.random.rand()
     self._outputs = np.zeros(inputs.shape[0])
     self.slope = []
     self.errors = []
```

Entrenamiento

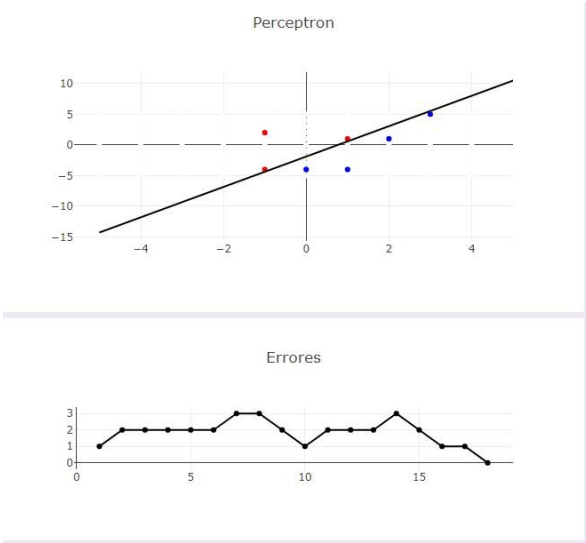
```
def train(self):
    activated = self.activation()
    results = self._expectedOutputs - activated
    error = np.sum(results)**2
    while error != 0:
        i = 0
        for inputs in self._inputs:
            self._weights = self._weights + (self._learningRate * results[i] * inputs)
            self._theta = self._theta + (self._learningRate * results[i] * 1)

            i += 1
            if(error == 0):
                break

            print('\n----------------------------\n')
            print("Expected Output:", self._expectedOutputs)
            print("Activated results:", activated)
            print("Errors:", results)
            print("Error:", error)
            print("Theta:", self._theta)

        activated = self.activation()
        results = self._expectedOutputs - activated
```

```
        error = np.sum(np.array(results**2))
        self.errors.append(error)

        self.graph(activated)

    print('\n-------------END--------------\n')
    print("Inputs: \n", self._inputs)
    print("Weights: \n", self._weights)
    print("Expected Output:", self._expectedOutputs)
    print("Activated results:", activated)
    print("Errors:", results)
    print("Error:", error)
    print("Theta:", self._theta)
```

Activación

```
def activation(self):
    results = []
    dot = np.dot(self._inputs, self._weights) + self._theta

    for result in dot:
        if result > 0:
            results.append(1)
        else:
            results.append(0)
    return np.array(results)
```

Interfaz de la aplicación



Problema de la vida real

Clasificación de basura orgánica e inorgánica

Existe una problemática bastante conocida a nivel global sobre la generación de grandes cantidades de basura que se acumula de manera desproporcionada en vertederos municipales, haciendo de que la forma más sanitaria de deshacerse de esos residuos sean los rellenos sanitarios, que terminan por filtrarse a los mantos acuíferos del subsuelo contaminando el valioso recurso líquido.

Sin embargo, también es bien sabido que una forma de combatir la generación de basura es separando los residuos en sus tipos, es decir, basura orgánica e inorgánica, siendo la segunda separable en más clases.

Si bien, existen muchas personas que ya hacen de manera personal la separación de basura, todavía existen más personas que no lo hacen, haciendo que el problema de los vertederos no sea disminuido.

Habiendo descrito la problemática, podemos observar que se podría aplicar el algoritmo del **perceptrón** para crear un sistema mecanizado que separe los residuos en dos clases, orgánicos e inorgánicos. Para esto, necesitaríamos sensores que puedan determinar las características que componen a cada uno de los residuos, para después aplicar el **perceptrón** y separarlos, conduciéndolos a su posterior almacenaje y tratamiento.

Las características que podríamos observar de estos residuos serian su forma, tamaño, color, peso, etc.

Las *ventajas* de esta aplicación serian poder crear sistemas con infraestructura suficiente para sanear todo el proceso del tratamiento de los residuos, haciendo más fácil el reciclaje de toda clase de residuos.

Las *desventajas* quedan en que el algoritmo del **perceptrón** de una capa parece demasiado corto para realizar una aplicación de este tipo, pues no solo existen dos clases de residuos.