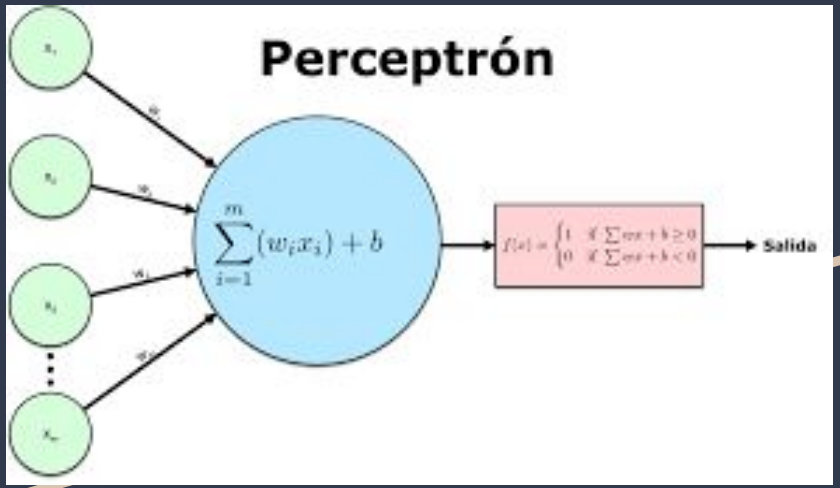
Primera Previa

Autores:

Juan Manuel Restrepo Urrego

Stiven Valencia Ramírez

1. Paper sobre perceptrón, su entrenamiento y backpropagation



Lo primeramente realizado fue el paper, en el cual explicamos y a detallada manera sobre lo que es el perceptrón, como funciona como se le ingresan los datos y un pequeño algoritmo llamado backpropagation, el cual como su nombre lo indica se basa en que se riega por toda la red neuronal y luego hace una recursión tipo backtracking.

La cual nos dice si las salidas fueron coherentes o tienen un porcentaje de error, ya que las salidas dependen totalmente de de las neuronas de primera capa por lo cual al hacer esta recursión les dirá si deben cambiar algo calculando del descenso gradiente.

Siempre se nos debe dar una red neuronal y una función de error, este método calcula el gradiente de dicha función.

El entrenamiento también es sencillo se basa en alimentarlo con múltiples muestras de entrenamiento, después cada neurona mostraría sus pesos W y ahí ya se empezaría a trabajar con diferentes algoritmos.

2. Código Colab

Para continuar con este contenido debemos indicar que a la hora de seleccionar un código ya que teníamos dos opciones, nos decidimos por lógica difusa, específicamente por un control difuso, el cual soluciona un problema de un cliente, que recibe un plato de comida con determinada calidad y también se tendrá en cuenta la calidad del servicio.

Dependiendo de estos dos factores (Calidad comida y servicio ofrecido) se le dará la cantidad de propina merecida; entonces con triángulos graficados, cada uno de ellos diciéndonos lo bueno o malo que se ofreció daremos una respuesta.

Pero claro no siempre será bueno o malo, puede ser que el plato sea bueno pero el servicio malo, entonces se aplicará la lógica difusa con reglas ya predeterminadas las cuales dicen si la calidad es baja entonces propina baja y así sucesivamente hasta medio o bueno

Pruebas

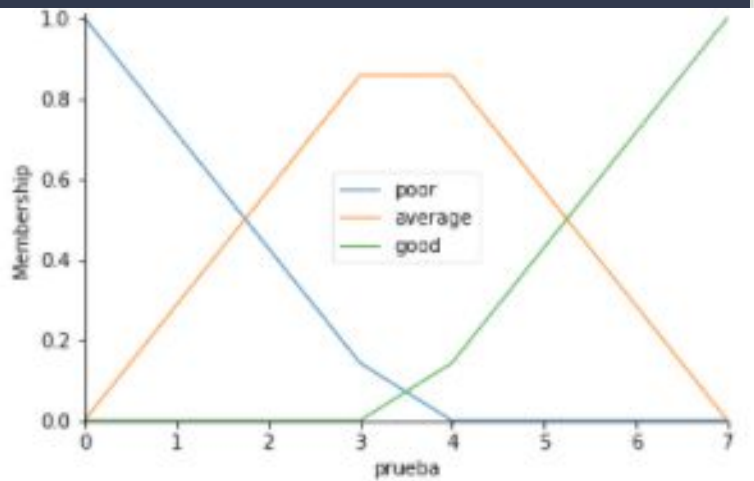
Copia de programa_40_control_difuso_API.ipynb

Archivo Editar Ver Insertar Entorno de ejecución Herramientas Ayuda Se edit...

+ Código + Texto

```
[ ] pip install scikit-fuzzy

    Collecting scikit-fuzzy
      Downloading scikit-fuzzy-0.4.2.tar.gz (993 kB)
         |████████████████████████████████| 993 kB 5.1 MB/s
    Requirement already satisfied: numpy>=1.6.0 in /usr/local/lib/python3.7/dis
    Requirement already satisfied: scipy>=0.9.0 in /usr/local/lib/python3.7/dis
    Requirement already satisfied: networkx>=1.9.0 in /usr/local/lib/python3.7/
    Building wheels for collected packages: scikit-fuzzy
      Building wheel for scikit-fuzzy (setup.py) ... done
      Created wheel for scikit-fuzzy: filename=scikit_fuzzy-0.4.2-py3-none-any.
      Stored in directory: /root/.cache/pip/wheels/d5/74/fc/38588a3d2e3f34f7458
    Successfully built scikit-fuzzy
    Installing collected packages: scikit-fuzzy
    Successfully installed scikit-fuzzy-0.4.2
```

```
# Se crean los objetos antecedentes y consecuente a partir de las
# variables del universo y las funciones de membresía
Prub = ctrl.Antecedent(np.arange(0, 8, 1), 'prueba')
calidad = ctrl.Antecedent(np.arange(0, 11, 1), 'calidad')
servicio = ctrl.Antecedent(np.arange(0, 11, 1), 'servicio')
propina = ctrl.Consequent(np.arange(0, 26, 1), 'propina')

# La población de la función de membresía automática es posible con .automf (3, 5 o 7)
Prub.automf(3)
calidad.automf(3)
servicio.automf(3)

# Las funciones de membresía personalizadas se pueden construir interactivamente con la
# API Pythonic
propina['bajo'] = fuzz.trimf(propina.universe, [0, 0, 13])
propina['medio'] = fuzz.trimf(propina.universe, [0, 13, 25])
propina['alto'] = fuzz.trimf(propina.universe, [13, 25, 25])

# Visualización con .view()
Prub.view()
calidad['average'].view()
servicio.view()
propina.view()
```



La primera imagen nos damos cuenta de la instalación de la librería para trabajar el código, la cual, como equipo de trabajo, revisamos aparte para lograr entender sus funciones, lo segundo hecho fue intentar crear un gráfico de prueba con sus triángulos que es lo que vemos en la segunda y tercera imagen.

Resultados

Para terminar decir que los resultados obtenidos para dar una propina adecuada son adecuadas debido que con un buen servicio y una calidad de comida regular no daremos la mejor propina pero si una decente que es de 19.84.

Los valores iniciales de calidad fueron 6.5 y de servicio es 9.8, con estos valores el control difuso realizo el valor propina.

Como conclusión debemos decir que este mundo de redes neuronales es muy grande y aveces algo abstracto en su entendimiento pero con un buen análisis y tiempo se entiende a la perfección al punto de decir que hasta se hace sencillo.