UANL FIME

Universidad Autónoma de Nuevo León

Facultad de Ingeniería Mecánica y Eléctrica

Unidad de Aprendizaje:

Laboratorio de Percepción

Actividad 2: Agentes de percepción basados en métricas de distancias

Equipo #4

Integrante: José Leonardp Estrada Ortiz

Matrícula: 1812097

P.E.: IMTC

Hora: N1 **Día:** Jueves **Grupo:** 001

Fecha de entrega: 12/Marzo/2021

San Nicolás de los Garza, Nuevo León

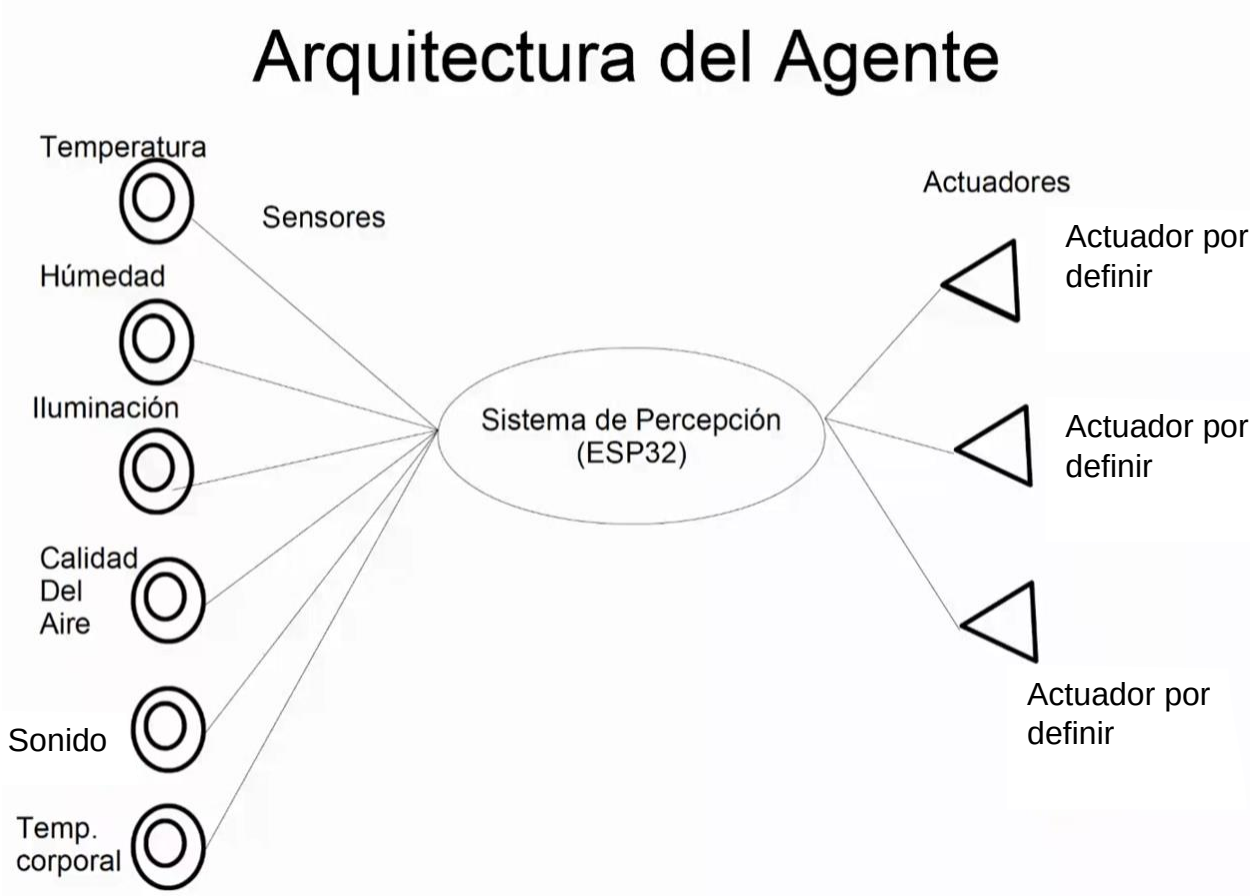

Ilustración 1. Imagen de la arquitectura del agente (basado en la clase)

Seudo-código auxiliar

```
Cálculo de distancia Hamming
Function Hamming_Distance (V1,V2)-> d
{B size of V1 or V2}
S=0;
For I=1 to B
    S=S + abs(V1(i)-V2(i))
End for
D=S
End function
```

Seudo código del agente

```
Function AgenteD(X,DB)->R
For r=1,TR {0,TR-1}
   For c=1,TC-1{0,TC-2}
       V1(c)=X(x);   V2(c)=DB(r,c);
    End for
     D(r)= Hamming_Distance (V1,V2)
End for
Minp(D)->(Val,pos)
R=DB(pos,TC)
End function
```

Tabla de Excel

5 Sonido (dB)	4 Luminosidad (%)	3 Calidad del aire (AQI)	2 Temperatura (°C)	1 Temperatura Corporal (°C)	6 Humedad (%)	Riesgo de Migraña
55	26.00%	25	36.52	37.74	4%	2
70	19.00%	339	16.92	39.68	53%	3
98	84.00%	360	36.82	39	71%	5
76	18.00%	205	25.38	38.59	72%	3
110	8.00%	393	40.80	39.92	71%	5
121	43.00%	424	33.74	39.04	66%	3
131	88.00%	119	25.40	37.06	24%	3
85	7.00%	137	3.58	39.78	69%	3
80	43.00%	322	14.93	36.64	2%	2
94	49.00%	160	14.84	39.68	38%	2
69	91.00%	92	44.11	39.65	22%	4
124	81.00%	305	29.75	39.39	88%	4
69	69.00%	202	35.43	39.97	55%	3
54	78.00%	110	8.36	36.81	12%	2
79	66.00%	340	42.66	39.02	42%	3
67	58.00%	172	1.12	36.76	46%	1
102	32.00%	27	1.30	36.08	83%	3
145	94.00%	86	36.43	36.85	58%	3
59	79.00%	176	19.23	38.22	55%	1
130	88.00%	16	6.61	38.41	17%	3
111	51.00%	310	5.49	38.62	81%	4
76	83.00%	15	13.09	36.45	33%	2
75	49.00%	237	42.18	39.01	100%	3
144	46.00%	221	26.55	37.73	77%	2
81	64.00%	25	38.95	37.93	100%	2
93	69.00%	125	34.09	39.3	2%	3
76	19.00%	12	25.86	39.82	91%	3
107	30.00%	52	15.80	36.88	93%	2
109	71.00%	85	1.53	39.19	82%	4
118	56.00%	23	22.08	38.42	42%	1
136	95.00%	90	7.90	39.08	24%	4
133	9.00%	25	38.43	38.24	44%	3
110	56.00%	52	42.38	37.94	97%	3
154	21.00%	63	16.02	39.06	75%	3
58	9.00%	125	41.35	36.05	25%	3

64	76.00%	128	14.79	38.24	50%	1
125	85.00%	31	10.96	39.1	32%	4
117	70.00%	457	5.95	38.25	88%	4
105	92.00%	121	43.43	39.33	59%	4
68	58.00%	41	27.08	37.65	44%	2
103	13.00%	111	14.41	39.97	17%	4
64	1.00%	13	39.90	38.51	85%	4
62	17.00%	15	21.40	36.25	17%	2
87	62.00%	280	29.52	39.61	31%	3
142	13.00%	113	20.06	39.11	13%	4
65	94.00%	468	44.41	36.87	57%	3
123	92.00%	57	38.02	39.43	27%	4
134	41.00%	118	44.34	37.95	23%	3
143	6.00%	304	25.09	36.62	8%	4
76	78.00%	32	13.47	39	8%	3
115	59.00%	78	36.72	37.93	55%	2
117	24.00%	65	39.44	36.81	27%	3
80	64.00%	424	37.48	38.67	37%	4
149	17.00%	45	41.72	39.35	97%	5
106	30.00%	317	1.42	38.29	64%	3
143	84.00%	35	3.63	39.71	71%	5
158	15.00%	143	26.69	37.81	20%	3
129	74.00%	45	31.73	39.39	67%	3
74	87.00%	14	36.98	37.15	63%	2
127	78.00%	6	6.81	37.94	33%	3
151	72.00%	452	32.13	39.83	98%	5
110	22.00%	327	29.52	38.33	80%	3
159	99.00%	125	31.61	36.58	2%	3
116	17.00%	23	28.77	36.85	80%	3
120	41.00%	309	5.95	38.58	59%	3
96	19.00%	98	43.53	36.35	80%	4
63	2.00%	32	19.54	40	25%	3
124	68.00%	376	3.14	38.99	90%	5
159	9.00%	121	27.73	38.86	24%	4
145	45.00%	46	33.56	36.77	6%	2
158	91.00%	124	32.59	38.87	90%	4
77	94.00%	331	21.33	38.27	97%	3
87	36.00%	10	8.74	39.5	31%	2
147	75.00%	197	42.40	39.79	53%	4
91	2.00%	71	38.25	37.17	44%	2
55	70.00%	46	28.79	36.4	17%	2

94	87.00%	402	39.63	36.81	64%	3
119	44.00%	88	16.23	38.47	22%	2
105	37.00%	44	39.22	38.34	50%	3
88	80.00%	121	6.63	38.09	75%	5
155	57.00%	13	25.10	36.55	48%	2
75	26.00%	460	32.97	38.04	94%	4
53	16.00%	173	7.16	38.25	7%	5
125	48.00%	46	26.17	37.5	45%	3
108	90.00%	27	18.74	37.85	30%	4
76	99.00%	31	11.09	37.66	47%	4
118	93.00%	446	16.40	39.28	67%	5
127	39.00%	354	27.57	39.88	32%	5
97	10.00%	176	44.04	38.82	34%	5
80	64.00%	45	0.57	39.64	89%	4
66	90.00%	500	44.77	38.4	82%	2
89	61.00%	23	13.39	36.68	28%	2
88	57.00%	249	24.27	36.37	69%	2
92	35.00%	452	16.25	39.4	49%	3
109	35.00%	450	1.24	39.08	21%	4
124	47.00%	141	20.59	37.28	14%	3
154	64.00%	46	7.94	39.51	11%	3
42	33.00%	30	20	36.1	51.00%	0
23	21.00%	35	25	36.9	56.00%	0
34	43.00%	25	23	36	55.00%	0

Nota: Debido a que usamos 100 datos para una mejor comprensión del riesgo optamos por clasificar los datos de cada sensor de la siguiente manera:

- Verde: Los datos medidos por el sensor no son peligrosos
- Amarillo: Los datos medidos por el sensor son medianamente peligrosos
- Rojo: Los datos medidos por el sensor son peligrosos

Código del agente

```
import numpy as np
from random import *
from math import *
import pandas as pd
from pandas import ExcelWriter
from pandas import ExcelFile
import matplotlib.pyplot as grafica


def normalizar(r,lb,ub):
    return (r-lb)/(ub-lb);



def desnormalizar(n,lb,ub):
    return n*(ub-lb)+lb;


def maxp(V):
    #(val,pos)=maxp(V)
    n=len(V);
    pos=0;
    val=V[pos];
    for e in range(n):
        if V[e]>val:
            val=V[e];
            pos=e;

    return val,pos

def minp(V):
    #(val,pos)=minp(V)
    n=len(V);
    pos=0;
    val=V[pos];
    for e in range(n):
        if V[e]<val:
            val=V[e];
            pos=e;

    return val,pos


def DatabaseRead():
    #Database or table
    #DataBrute=DatabaseRead();
    #Excel reading
    #df = pd.read_excel('Desktop/Projects2020/PythonCurse/Neuralnetworks/datasetNN.xls')
    #df = pd.read_excel('datasample.xls')
    df = pd.read_excel('DatosSensores.xlsx');
    Nrows=len(df); Ncols=len(df.columns);
    DataBrute = [[0 for i in range(Ncols)] for j in range(Nrows)];
    for r in range(Nrows):
        for c in range(Ncols):
            DataBrute[r][c]=df[df.columns[c]][r];
```

```
72          return DataBrute
73
74
75      def plotgraph(V):
76          N=len(V);
77          D=[ j for j in range(N)]
78          grafica.figure(1)
79          grafica.plot(D,V,'b:',linewidth=2)
80          grafica.xlabel('Número de datos')
81          grafica.ylabel('Señal')
82          grafica.show()
83
84          #programa principal
85
86
87      def NormalData(DataExp):
88      ###LMTT092018
89      ###  (DataNorm,MRange)=NormalData(DataExp)
90      ##
91          Trows=len(DataExp);
92          Tcols=len(DataExp[0]);
93          V = [0 for i in range(Trows)];
94          MRange = [[0 for i in range(2)] for j in range(Tcols)];
95          DataNorm = [[0 for i in range(Tcols)] for j in range(Trows)];
96          for c in range(Tcols):
97              for r in range(Trows):
98                  V[r]=DataExp[r][c];
99              (valmax,posmax)=maxp(V);
100             (valmin,posmin)=minp(V)
101             for r in range(Trows):
102                 DataNorm[r][c] = normalizar(DataExp[r][c],valmin,valmax);
103             MRange[c][0]=valmin;
104             MRange[c][1]=valmax;
105         return DataNorm, MRange
106
107
108
109     def Hamming_Distance(V1,V2):
110         B = len(V1);
111         s = 0
112         for b in range(B):
113             s = s + abs(V1[b]-V2[b]);
114         d = s;
115         return d
116
117
118     def Agente(X,DB,MRange):
119         Trows = len(DB);
120         Tcols = len(DB[0]);
121         Xn = [ 0 for j in range(Tcols-1)];#Tcols-1 = total de entradas
122         D = [ 0 for j in range(Trows)];
123         V1 = [ 0 for j in range(Tcols-1)];
124         V2 = [ 0 for j in range(Tcols-1)];
125         y = 0;
126         for b in range(Tcols-1):
127             Xn[b] = normalizar(X[b],MRange[b][0],MRange[b][1]);
```

```
128
129
130         for r in range(Trows):
131             for c in range(Tcols-1):
132                 V1[c] = Xn[c];
133                 V2[c] = DB[r][c];
134
135             #Cálculo de distancia Euclideana
136             D[r] = Hamming_Distance(V1,V2);
137
138         #Distancia mínima
139         (val,pos) = minp(D);
140
141         #Extraer la respuesta
142         y = DB[pos][Tcols-1]
143
144         y = desnormalizar(y,MRange[Tcols-1][0],MRange[Tcols-1][1]);
145
146         return y
147
```

Vector de respuesta satisfactoria

```
In [12]: DataBrute=DatabaseRead();

In [13]: (DataNorm,MRange)=NormalData(DataBrute)

In [14]: X=[34,43,25,23,36,55]

In [15]: r=Agente(X,DataNorm,MRange)

In [16]: print(r)
0.0
```

Ilustración 2. Vector que da como respuesta un riesgo nulo.

Vector de respuesta no satisfactoria

```
In [53]: X=[149,17,45,41.72,39.35,97]

In [54]: r=Agente(X,DataNorm,MRange)

In [55]: print(r)
5.0
```

Ilustración 3. Vector que da como respuesta un riesgo alto.

Vector de respuesta intermedia

```
In [47]: X=[69,18,338,15.92,38.68,52]

In [48]: r=Agente(X,DataNorm,MRange)

In [49]: print(r)
3.0
```

Ilustración 4. Vector que da como respuesta un riesgo intermedio.

Conclusión

Durante este trabajo se estudiaron dos tipos de cálculo de distancias. La primera se vio a lo largo de la clase y fue la Euclideana, pero para desarrollar correctamente esta actividad se estudió la distancia de Hamming. Gracias a esta actividad se puto tener un mejor entendimiento de cómo trabajan ambos cálculos, así como también se aprendió el cómo normaliza y desnormalizar los datos en Python.

Se esperaba que fuera una actividad un poco más compleja, pero una vez que el equipo entendió mejor lo que se había que hacer, fue fácil de desarrollar y tener buenos resultados.