

REPORTE TAREA #5

INTELIGENCIA

COMPUTACIONAL

línea horizontal

Jesús Rivera Flores - A00820643

Diego Arnoldo Azuela Rosas - A01208345



**CONTENIDO**

# 

[**INSTRUCCIONES**](#_cixyks6g2g16) **4**

[NOMBRE TAREA](#_49k4nmgv0c6q) 4

[OBJETIVO](#_av5f2uckkgny) 4

[INSTRUCCIONES](#_16hk2purkd4v) 4

[**DESCRIPCIÓN DEL PROBLEMA**](#_5ndwpoeto1bh) **4**

[**REPRESENTACIÓN EN LISP OBTENIDA**](#_b45yk0tvun4c) **6**

[DESARROLLO](#_wjw3hbkbzoul) 6

[add(mul(y, 9), x) (0.0,)](#_7afoui412tcx) 6

[add(10, mul(y, 8)) (0.0,)](#_md0uv51t4j49) 6

[**FUNCIÓN QUE MODELA LA REPRESENTACIÓN**](#_lynvhvofc38f) **6**

[DESARROLLO](#_61u6kspjci1i) 6

[SOLUCIÓN #1](#_7c69gvxiyit2) 6

[SOLUCIÓN #2](#_5av5v7kj7il) 6

[**SIMPLIFICACIÓN DE LA FUNCIÓN (EN CASO DE SER POSIBLE)**](#_ypkhji7bci4p) **7**

[DESARROLLO](#_gqt6pjaldb50) 7

[**REPRESENTACIÓN EN LISP (EN CASO DE QUE EXISTA)**](#_6gjwalx6gn2) **7**

[DESARROLLO](#_re4eh7r29nin) 7

[**ERROR CUADRÁTICO MEDIO (MSE)**](#_jtdem3fquwnh) **7**

[DESARROLLO](#_2unjhti1v5ue) 7

[ERROR CUADRÁTICO MEDIO](#_amravvo8owbj) 7

[**MENCIONAR SI HAY ERROR EN EL MAPEA (RELACIONADO AL PUNTO DE ARRIBA)**](#_73bq2bwzrqt0) **7**

[DESARROLLO](#_rwsobs8lloql) 7

[**CONCLUSIÓN Y RETOS ENCONTRADOS**](#_xo9m9qrh7icf) **7**

# 

# 

# INSTRUCCIONES

## NOMBRE TAREA

* Programación genética

## OBJETIVO

* Utilizar programación genética para encontrar la función que mapea los datos de entrada a un valor de salida.

## INSTRUCCIONES

* Utiliza la librería deap para correr el algoritmo de programación genética y encontrar la función que mapea los siguientes datos (dos entradas, una salida):
  1. entradas = [(0, 10), (1, 9), (2, 8), (3, 7), (4, 6), (5, 5), (6, 4), (7, 3), (8, 2), (9, 1)]
  2. salidas = [90, 82, 74, 66, 58, 50, 42, 34, 26, 18]

# DESCRIPCIÓN DEL PROBLEMA

* Al realizar este problema, el equipo utilizó la misma librería de la tarea pasada: *DEAP*. Lo que se busca es utilizar la librería de programación genética para encontrar una función que caracterice de manera adecuada y sea capaz de mapear la relación de entrada a un solo valor de salida. Tomando un par ordenado el programa debe ser capaz de evaluar dichas entradas y determinar la salida.
* El IDE utilizado es Colab de Google. Formato similar al presentado por Jupyter Notebooks, pero con la versatilidad de Google que permite acceder a la información de diferentes puntos y con la seguridad de que todo se ha guardado en la nube. Se presentará el código junto con una foto de la gráfica que produce el código.

| **DESARROLLO DEL CÓDIGO** | |
| --- | --- |
| **LIBRERÍAS** | import math  import random  import operator  import numpy as np  import pandas as pd  from sklearn.metrics import mean\_squared\_error  from deap import base, creator, algorithms, tools, gp  toolbox = base.Toolbox() |
| **DATOS DE ENTRADA** | df = pd.DataFrame({'x':list(range(0,10),),  'y':list(range(10,0,-1),),  'f(x)':[90, 82, 74, 66, 58, 50, 42, 34, 26, 18]})  x = df['x'].values.tolist()  y = df['y'].values.tolist()  z = df['f(x)'].values.tolist() |
| **FUNCIONES** | def eval\_func(ind, x, y, outputs):  func\_eval = toolbox.compile(expr=ind)  predictions = list(map(func\_eval,x,y))  return abs(mean\_squared\_error(outputs,predictions)), |
| def div (x,y):  try:  return x/y  except ZeroDivisionError:  return 1 |
| def pot(x,y):  try:  return math.pow(x,y)  except:  return 1 |
| **DESARROLLO** | pset = gp.PrimitiveSet("MAIN", 2)  pset.renameArguments(ARG0='x')  pset.renameArguments(ARG1='y')  pset.addPrimitive(operator.add, 2)  pset.addPrimitive(operator.sub, 2)  pset.addPrimitive(operator.mul, 2)  pset.addPrimitive(div, 2)  pset.addPrimitive(math.sin, 1)  pset.addPrimitive(pot, 2)  pset.addTerminal(math.e, 'e')  pset.addEphemeralConstant('R', lambda: random.randint(1,10))  creator.create('FitnessMin', base.Fitness, weights=(-1.0,))  creator.create('Individual', gp.PrimitiveTree, fitness=creator.FitnessMin, pset=pset)  toolbox.register('expr', gp.genHalfAndHalf, pset=pset, min\_=3, max\_=10)  toolbox.register('select', tools.selDoubleTournament, fitness\_size=2, parsimony\_size=1.4, fitness\_first=False)  toolbox.register('mate', gp.cxOnePoint)  toolbox.register('mutate', gp.mutNodeReplacement, pset=pset)  toolbox.register('evaluate', eval\_func, x=x, y=y, outputs=z)  toolbox.register('compile', gp.compile, pset=pset)  toolbox.register('individual', tools.initIterate, creator.Individual, toolbox.expr)  toolbox.register('population', tools.initRepeat, list, toolbox.individual)  stats = tools.Statistics(key=lambda ind: ind.fitness.values)  stats.register('min', np.min)  stats.register('max', np.max)  stats.register('mean', np.mean)  stats.register('std', np.std)  hof = tools.HallOfFame(5)  pop = toolbox.population(n=40)  last\_population, log = algorithms.eaSimple(pop,toolbox,cxpb=0.5,mutpb=0.25,ngen=20,stats=stats,halloffame=hof) |
| **IMPRIMIR RESULTADOS** | for ind in hof:  print(ind)  print(toolbox.evaluate(ind, x=x, y=y, outputs=z)) |
| **RESULTADOS** |  |

# REPRESENTACIÓN EN *LISP* OBTENIDA

## DESARROLLO

* El lenguaje LISP es una expresión que el sistema puede interpretar como una solución escrita de un objeto. De esta representación se puede pasar a una solución de lenguaje interpretado. Al correr el código se encontraron 2 soluciones, presentadas a continuación:

| add(mul(y, 9), x) | add(10, mul(y, 8)) |
| --- | --- |
|  |  |

# FUNCIÓN QUE MODELA LA REPRESENTACIÓN

## DESARROLLO

De acuerdo a lo establecido en el punto anterior, se tiene una representación LISP de la función, a continuación se tomará

|  | SOLUCIÓN #1 | SOLUCIÓN #2 |
| --- | --- | --- |
| FUNCIÓN QUE MODELA | 9y + x | 10 + 8y |
| REPRESENTACIÓN LISP | add(mul(y, 9), x) (0.0,) | add(10, mul(y, 8)) (0.0,) |
| REPRESENTACIÓN GRÁFICA |  |  |

# SIMPLIFICACIÓN DE LA FUNCIÓN (EN CASO DE SER POSIBLE)

## DESARROLLO

En este caso, no es necesario tener una representación simplificada del sistema dado que la primera respuesta ya se encuentra simplificada.

# REPRESENTACIÓN EN *LISP* (EN CASO DE QUE EXISTA)

## DESARROLLO

En este caso, no es necesario repetir la representación, dado que la primera representación LISP era de la función ya simplificada.

# ERROR CUADRÁTICO MEDIO (MSE)

## DESARROLLO

|  |  | ERROR CUADRÁTICO MEDIO |
| --- | --- | --- |
| 1 | add(mul(y, 9), x) | (0.0,) |
| 2 | add(10, mul(y, 8)) (0.0,) | (0.0,) |

# MENCIONAR SI HAY ERROR EN EL MAPEA (RELACIONADO AL PUNTO DE ARRIBA)

## DESARROLLO

En este caso, dado que no existe error cuadrático medio, por lo cual no existe error en el mapeo de las soluciones.

# CONCLUSIÓN Y RETOS ENCONTRADOS

Generalmente la parte más complicada es familiarizarse con las diferentes variables y/o ramificaciones que existen dentro de una librería. Las variables de las que se puede hacer uso y la representación con la cual se despliegan las soluciones. En este caso, dado que la tarea anterior fue utilizando la misma librería el equipo se encontraba familiarizado con los diferentes parámetros del algoritmo, algo que facilitó bastante el desarrollo de esta tarea. El usuario solamente necesita ingresar un valor fijo en el lugar adecuado para lograr un algoritmo funcional y de uso inmediato. La realización de este código fue interesante y probó ser un reto.Es un algoritmo robusto que maneja soluciones aplicables a diferentes problemáticas.