

Computación Neuronal y Evolutiva

(Práctica 6)



UNIVERSIDAD DE BURGOS

Autores:

Mario Ubierna San Mamés

Jorge Navarro González



Índice de Contenido

Descripción de la carga y almacenamientos de los datos	4
Descripción de cómo se ha adaptado la computación evolutiva a la solución del problema:	5
¿Cómo se representan los individuos?	5
¿Cómo se calcula la adaptación?	6
Restricciones	6
Multi-Objetivo	6
Ventajas e inconvenientes del genotipo y el fitness	7
Fenotipo que nos devuelve nuestro algoritmo	8
Restricciones	8
Multi-Objetivo	9
Experimentos	10
Restricciones	11
Caso Base:	11
Caso 1:	11
Caso 2:	11
Caso 3:	12
Caso 4:	12
Resultado caso base	13
Cambiamos la probabilidad de cruce	14
Cambiamos la probabilidad de mutación	16





Descripción de la carga y almacenamientos de los datos

Para realizar la carga de datos hemos usado un fichero al igual que en la práctica anterior llamado `LecturaDatos.py`, que está contenido en el zip.

El cómo guardamos los datos, estarán contenidos una vez leídos en variables que van a ser las siguientes:

- **Anos:** es un vector en el que almacenamos en cada posición cada uno de los años que hemos leído del fichero.
- **Población:** es un vector en el que almacenamos para cada posición el valor correspondiente de la población total para cada año.
- **Pob_estimada:** es un vector en el que una vez realizado el algoritmo evolutivo, guardamos la población estimada para cada año para el mejor genotipo.
- **Mejor_genotipo:** es un vector en el que almacenamos el mejor genotipo una vez realizado el algoritmo evolutivo.

Estas son las variables en las que guardamos los datos necesarios que leemos de cualquiera de los ficheros proporcionados en la plataforma.

Descripción de cómo se ha adaptado la computación evolutiva a la solución del problema:

¿Cómo se representan los individuos?

Esta práctica trata sobre la programación genética, la principal diferencia respecto a las anteriores prácticas es que ahora el genotipo viene representado por un árbol y no por un vector.

Para ello debemos definir el conjunto de nodos terminales, y el conjunto de nodos funcionales.

Los nodos funcionales vienen determinados por las siguientes operaciones:

```
pset.addPrimitive(operator.add, 2)
pset.addPrimitive(operator.sub, 2)
pset.addPrimitive(operator.mul, 2)
pset.addPrimitive(protectedDiv, 2)
pset.addPrimitive(operator.neg, 1)
pset.addPrimitive(math.cos, 1)
pset.addPrimitive(math.sin, 1)
```

Por otro lado, en este problema solo consideramos que tenemos un nodo terminal, el cual es la X:

```
pset.renameArguments(ARG0='x')
```

Según nuestro programa, la representación de un genotipo podría ser la siguiente:

```
sub(mul(x, add(x, x)), neg(mul(x, x)))
```

Como podemos apreciar en la captura anterior, este árbol está formado por una resta de dos elementos, de una multiplicación, de una suma, de una negación y de otra multiplicación.

¿Cómo se calcula la adaptación?

Restricciones

Para nuestro ejercicio realizado con restricciones tendremos nuestra función fitness que es la que se encarga de evaluar a cada genotipo, en este caso no se ha considerado introducir una penalización, ya que al hacer error cuadrático medio siempre nos devuelve una población positiva, por lo tanto cualquier solución positiva es válida para nuestro problema.

A nuestra fitness, como podemos observar en la siguiente imagen, le pasamos dos parámetros, el toolbox y el genotipo.

```
def fitness(toolbox,genotipo):  
    resultado = 0  
    funcion = toolbox.compile(expr=genotipo)  
    for i,j in zip(dc.anos,dc.poblacion):  
        resultado += (funcion(i) - j)**2  
    return resultado/len(dc.anos),
```

Tal y como podemos ver, la fitness es muy simple, ya que lo primero que hacemos es calcular la función del genotipo que le pasamos por cabecera, luego recorremos los años y la población, para así poder calcular para cada año el error cuadrático, y finalmente dividimos entre el número de años para obtener el error cuadrático medio.

Multi-Objetivo



Ventajas e inconvenientes del genotipo y el fitness

En esta práctica en cuanto al genotipo no tenemos mucho que comentar, ya que todos los compañeros tenemos como genotipo un árbol ya que se trata de programación genética. Lo que sí que podemos comentar es que el conjunto de nodos terminales y nodos funcionales, hemos considerado que con un nodo terminal sirve, ya que solo con éste nos permite obtener el valor necesario para calcular el error cuadrático medio.

Por otro lado, para el conjuntos de nodos funcionales hemos considerado que lo mejor era usar la documentación de deap (https://deap.readthedocs.io/en/master/examples/gp_symbreg.html), ya que aunque nuestro programa no nos proporciona una predicción exacta, sí que podemos intuir como va a ser la población en un futuro con dicho operadores.

En cuanto la fitness, se pueden usar diferentes errores para poder realizar la predicción, sin embargo, estamos más acostumbrados a usar el error cuadrático medio, y es por ello que ésta fue nuestra elección.

Además cabe destacar, que el cálculo del error cuadrático medio es muy sencillo, no es como otros errores los cuales necesitan más datos, es por ello que elegimos este error, como la base para realizar el cálculo de nuestra fitness.



Fenotipo que nos devuelve nuestro algoritmo

Restricciones

El fenotipo que nos va a devolver nuestro algoritmo es un fichero, el cual se llama "fenotipo.out". En dicho fichero vamos a almacenar para cada año la predicción de la población de la mejor solución.

El fenotipo que nos devuelve nuestro programa es el siguiente:

Anio	Poblacion Estimada
1960	11524800
1961	11536563
1962	11548332
1963	11560107
1964	11571888
1965	11583675
1966	11595468
1967	11607267
1968	11619072
1969	11630883
1970	11642700
1971	11654523
1972	11666352
1973	11678187
1974	11690028
1975	11701875
1976	11713728
1977	11725587
1978	11737452
1979	11749323
1980	11761200
1981	11773083
1982	11784972
1983	11796867
1984	11808768
1985	11820675
1986	11832588

La predicción para cada año viene determinada por el fichero de entrada que le pasamos al programa, la fecha de inicio puede variar, sin embargo, la fecha final es siempre el 2017.

Como podemos ver en la imagen el significado de este fichero sería, para el año 1986 la población estimada sería 11832588.



Multi-Objetivo



Experimentos

Para la realización de los experimentos hemos seguido las pautas que nos indicaron en ubuvirtual, las cuales son las siguientes:

- 1- Seleccionar una configuración base. Basta con que aporte una solución medianamente buena. Simplemente es para tener un punto de partida.
- 2- Seleccionar un aspecto a comprobar (Ej: Tipo de Selección, Número de Individuos, Probabilidad de Cruce, etc).
- 3- Elegir el rango disponible para probar. El resto de los parámetros quedan fijos según la configuración base elegida en (1):
 - a. Si es un número, se debe comprobar un rango de valores. 4 o 5 son suficientes (Ej: num individuos = [50, 100, 150, 200, 250]).
 - b. Si es una selección, simplemente se eligen entre los disponibles (Ej: Selección = [Torneo, Ruleta, Restos])
- 4- Seleccionar 2 o 3 configuraciones del problema / ficheros.
- 5- Repetir la ejecución del algoritmo una vez por cada opción en el punto (3) y cada fichero en (4).
- 6- Agrupar los datos de cada experimento en tablas o gráficos para presentarlos. Por cuestiones de resumen de información, incluso se puede comprobar solamente el valor óptimo encontrado en cada ejecución, en lugar de la gráfica completa de la búsqueda.

Cabe mencionar que hicimos 4 experimentos, y en cada experimento hicimos 9 casos, hemos ejecutado esos 9 casos para los 4 ficheros de cada experimento.



Restricciones

Las pruebas realizadas son sobre los siguientes ficheros:

- BrasilCensus_2017.csv
- JapanCensus_2017.csv
- EEUUCensus_2017.csv

Para cada uno de los ficheros hemos realizados los siguientes casos, cabe mencionar que ya no podemos indicar el número de generaciones que queremos, ya que nosotros lo que hacemos es, si en x generaciones consecutivas el mejor genotipo de cada generación tiene el mismo fitness, entonces finaliza el algoritmo (en la ejecución de todos los casos consideramos que si en 7 generaciones obtenemos el mismo fitness paramos). Por otro lado, indicamos que el número de individuos por generación sea siempre el número de años que ha leído por fichero.

Caso Base:

$$alg_param['cxpb'] = 0.75$$
$$alg_param['mutpb'] = 0.2$$
$$alg_param['pop_size'] = 25$$

Caso 1:

$$alg_param['cxpb'] = 0.5$$
$$alg_param['mutpb'] = 0.2$$
$$alg_param['pop_size'] = 25$$

Caso 2:

$$alg_param['cxpb'] = 0.9$$
$$alg_param['mutpb'] = 0.2$$
$$alg_param['pop_size'] = 25$$



Caso 3:

alg_param['cxpb'] = 0.75

alg_param['mutpb'] = 0.1

alg_param['pop_size'] = 25

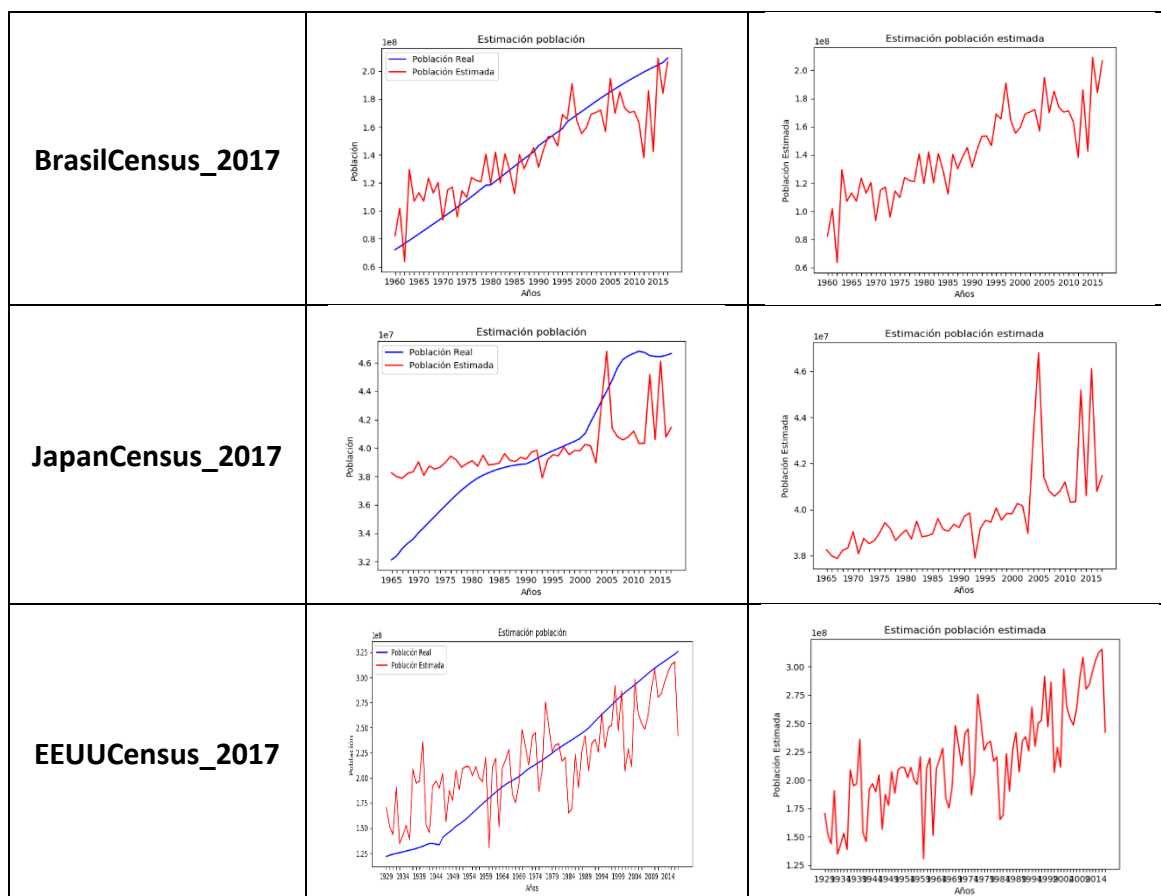
Caso 4:

alg_param['cxpb'] = 0.75

alg_param['mutpb'] = 0.5

alg_param['pop_size'] = 25

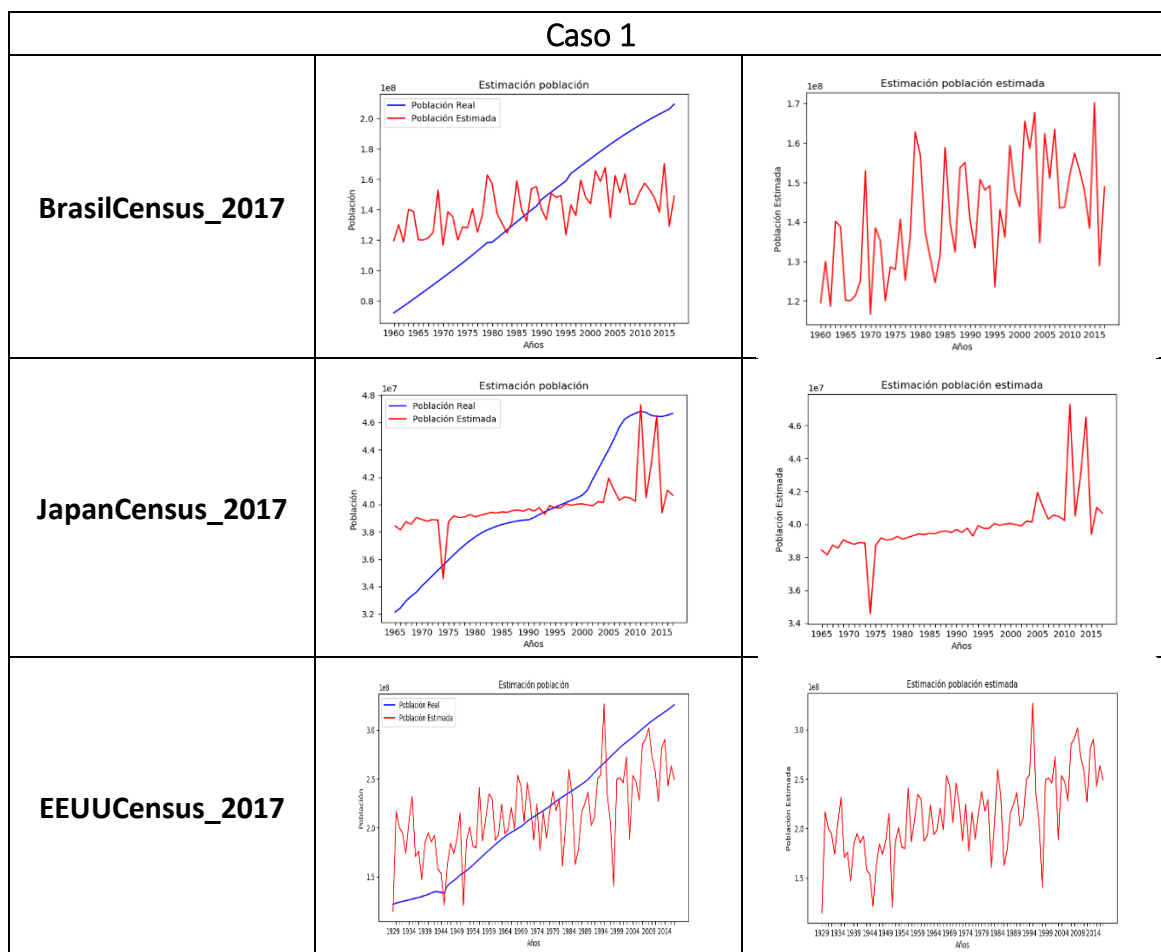
Resultado caso base



Este apartado lo hemos realizado para que se vea de una forma más sencilla los cambios que van a suceder con los siguientes casos.

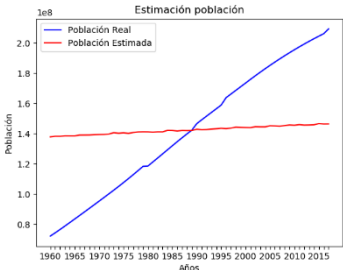
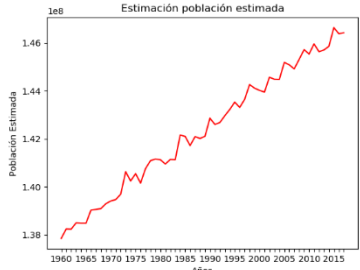
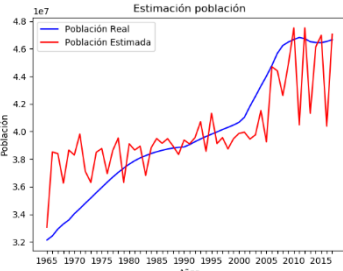
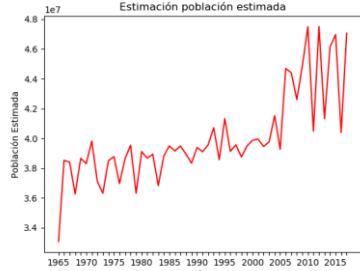
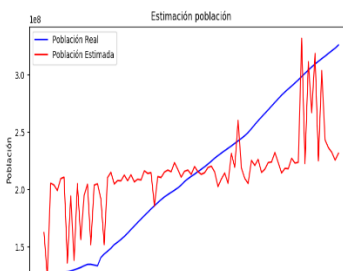
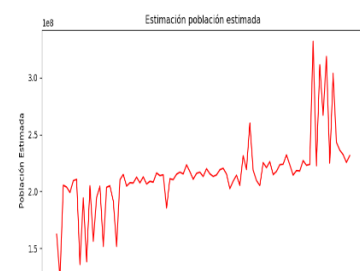
Cabe mencionar, que como podemos ver en la imagen, la estimación depende mucho del problema y de la aleatoriedad, ya que por ejemplo la estimación de Brasil la realiza bastante bien, pero la de Japón no, por lo que si hacemos que el algoritmo tarde más en pararse, vamos a obtener mejores aproximaciones, sin embargo esto puede durar horas o incluso días.

Cambiamos la probabilidad de cruce



Tal y como podemos apreciar en la tabla anterior, al disminuir la probabilidad de cruce, hacemos que en la siguiente generación haya más individuos élites, esto tiene el lado bueno de que son buenos genotipos, sin embargo, estamos disminuyendo y mucho la diversidad.

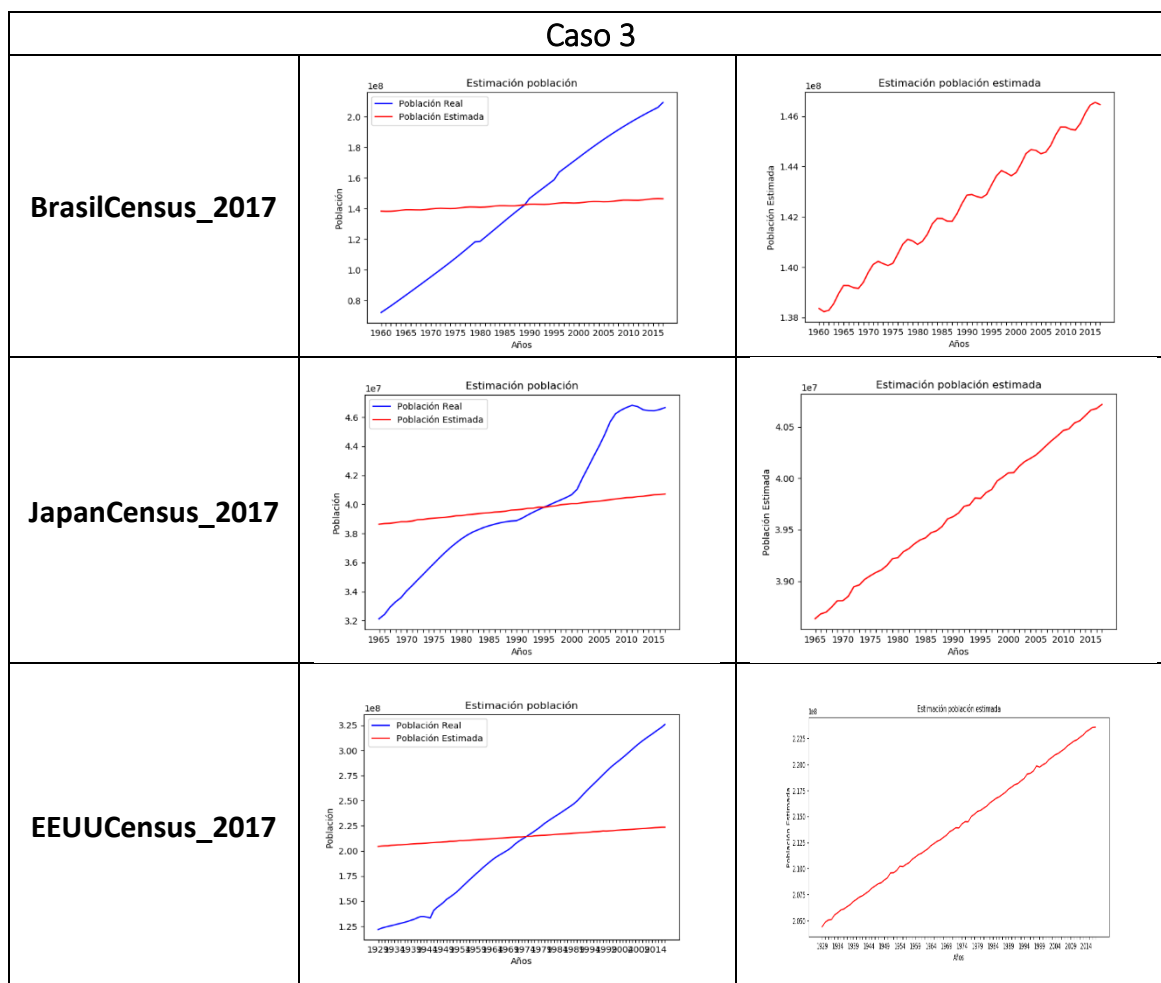
Como se puede apreciar, a excepción de los Estados Unidos, en los demás hace una peor predicción, ya que elimina diversidad y por lo tanto obtenemos una peor solución.

Caso 2		
BrasilCensus_2017		
JapanCensus_2017 7		
EEUUCensus_2017		

Cuando uno aumenta la probabilidad de cruce lo que provoca es que haya menos individuos élités en la siguiente generación, ya que se cruzan más. Esto tiene la ventaja de que va a generar mucha diversidad, pero no siempre generar una alta diversidad es bueno ya que esto provoca que el problema no converja de la forma correcta.

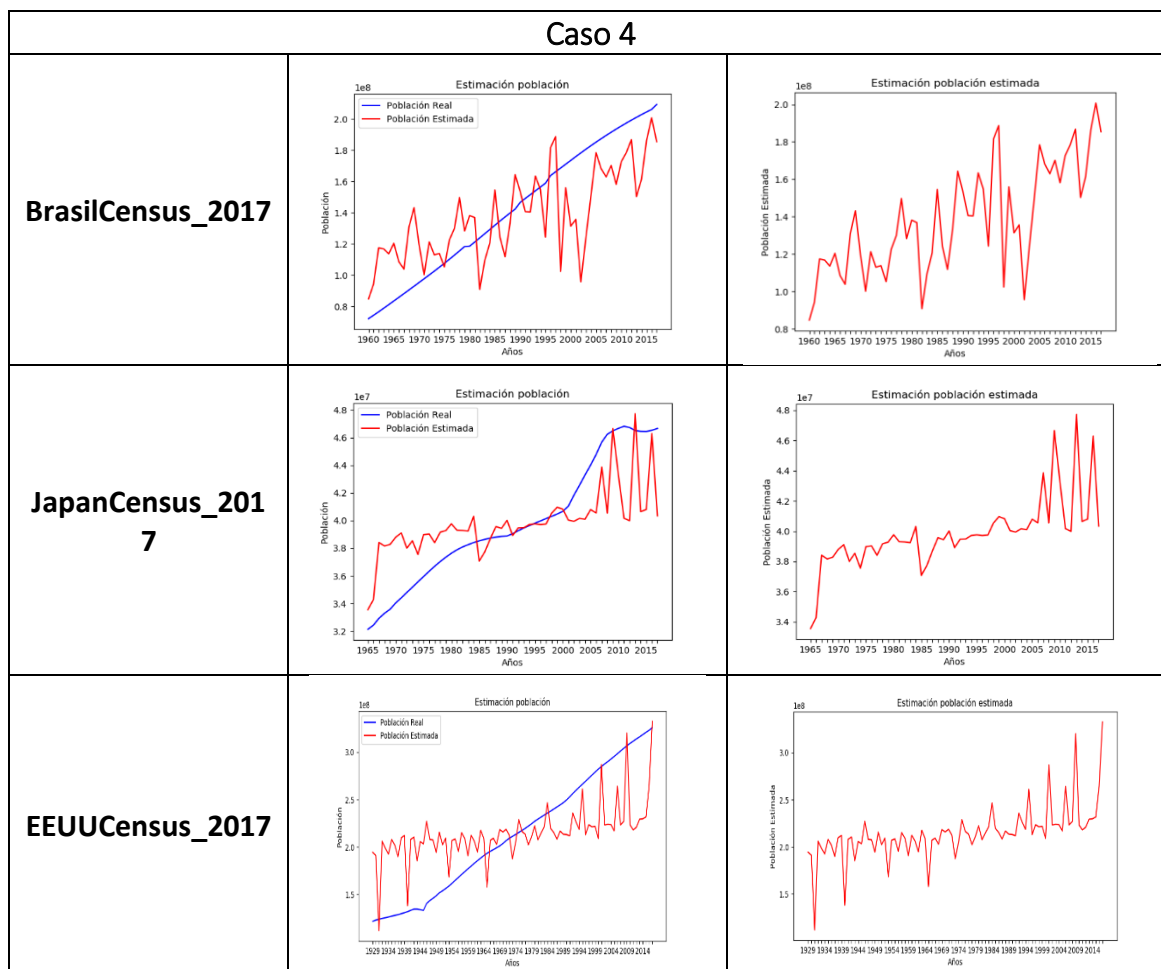
Como podemos apreciar en las capturas de la tabla, este ha sido el caso en el que peores aproximaciones tenemos (a excepción de Japón que en este caso es la mejor aproximación que hemos tenido), ya que al generar tanta diversidad no se consigue converger hacia el punto de tal forma que la función estimada se aproxime a la real.

Cambiamos la probabilidad de mutación



Cuando reducimos la probabilidad de mutación lo que estamos haciendo es reducir la diversidad, ya que no generamos nuevas características, esto no es bueno por lo que acabamos de mencionar.

Como podemos apreciar en las imágenes de la tabla, este ha sido el peor caso en el que aproximamos una función, ya que la función que generamos respecto de la real no se parecen en nada, esto se debe, a que el algoritmo no explora nuevas soluciones, es decir, la población sigue con características muy similares de generación en generación.



Cuando aumentamos la probabilidad de mutación, conseguimos aumentar la diversidad de nuestras soluciones, esto tiene una parte buena ya que conseguimos nuevas características, pero también tiene su parte mala ya que no convergemos hacia un punto.

Respecto a la aproximación que realiza nuestro programa sobre Brasil es bastante buena, ya que sigue la tendencia de la función real, es decir, generar más diversidad de lo habitual nos ha ayudado a mejorar más que en el caso anterior, sin embargo no es tan buena como desearíamos.

Con Japón nos sucede lo mismo que con Brasil, la aproximación es mejor que en el caso anterior, pero aun así no es la ideal.

Con Estados Unidos nos realiza una aproximación decente, aunque peor que la aproximación de Brasil o de Japón.