Modelación y simulación del modelo de difusión de epidemia mediante simulación de redes neuronales y procesos estocásticos.

Servando López

Ricardo Rosas Esquivel	A00829089
Gilberto Juárez Rangel	A01570628
Luis Rodrigo Medina Murua	A01283783
Fátima Cruz López	A01552037
Francisco Javier Vázquez Tavares	A00827546

1. Fase 3

1.1. Introducción

Acerca del modelo SIR. Este modelo epidemiológico calcula teóricamente la cantidad de personas infectadas y recuperadas a lo largo del tiempo debido a una enfermedad contagiosa(*1*). El sistema de ecuaciones que modelan el comportamiento es,

$$\frac{\mathrm{d}S(t)}{\mathrm{d}t} = -\beta S(t)I(t), \tag{1}$$

$$\frac{\mathrm{d}I(t)}{\mathrm{d}t} = \beta S(t)I(t) - \gamma I, \tag{2}$$

$$\frac{\mathrm{d}R(t)}{\mathrm{d}t} = \gamma I, \tag{3}$$

donde $S(t)$ representa la cantidad de susceptibles, $I(t)$ la cantidad de infectados y $R(t)$ la cantidad de recuperados en el tiempo. Este modelo, considera que una cantidad de población hará contacto con otras personas tal que se transmita la infección con una taza de βN por unidad de tiempo, mientras que los infectados dejarán de infectar a una taza de γ y no hay crecimiento o decrecimiento de la población durante la transmisión de la enfermedad(*2*). Por lo tanto, el parámetro β representa la cantidad de personas infectas por unidad temporal, mientras que γ representa la tasa de personas recuperadas por unidad de tiempo. En estos casos, ambas tazas están normalizadas, causando que $\gamma, \beta \in [0,1]$.

En este sistema de ecuaciones se puede caracterizar a través de un parámetro que representa el número de personas infectadas por una persona ya infectada(*1*, *2*), este parámetro se expresa

de la siguiente manera,

$$R_o = \frac{\beta}{\gamma}. \tag{4}$$

Este parámetro es adimensional y se puede usar como una herramienta para caracterizar el comportamiento de la propagación de la enfermedad, como se muestra en la tabla 1.

R_o	**Enfermedad**
1 - 2	Gripe porcina
2 - 5	Covid-19
5 - 7	Paperas
7 - 12	Varicela
12 - 18	Sarampión

Tabla 1: Rangos de R_o para las enfermedades a identificar

Por otro lado, una manera de resolver el sistema de ecuaciones es realizando una aproximación con diferencias finitas, obteniendo las siguientes ecuaciones en un dominio temporal discreto,

$$\begin{bmatrix} S_{t+1} \\ I_{t+1} \\ R_{t+1} \end{bmatrix} = \begin{bmatrix} -\beta S_t \\ \beta S_t - \gamma \\ -\gamma \end{bmatrix} I_t \Delta t + \begin{bmatrix} S_t \\ I_t \\ R_t \end{bmatrix}, \tag{5}$$

en donde, el subíndice indica el tiempo.

Modelación de una pandemia usando programación por agentes. Otra manera de simular la evolución de una pandemia es usando un algoritmo en base a agentes, el cuál se basa en simular el comportamiento de un sistema a partir del comportamiento de agentes individuales, usando los siguientes elementos: agentes, ambiente y las reglas que gobiernan el comportamiento de los agentes (*3*).

Una implementación en base a agentes para simular la pandemia considerando una población cerrada, junto con una interacción homogénea entre personas-cumpliendo con las mismas características que el modelo SIR-se caracteriza por simular una cantidad N de agentes recorriendo un ambiente y cuando un agente no infectado se acerca a un agente infectado, tiene una probabilidad de contagiarse, mientrás que el agente infectado, con cada iteración temporal, tiene una probabilidad de recuperarse dictada por una distribución normal. Ejemplos de implementación en base a agentes se pueden encontrar en NetLogo(*4*) (figura1) o realizar una implementación desde cero en Python usando la paquetería Mesa(*5*).

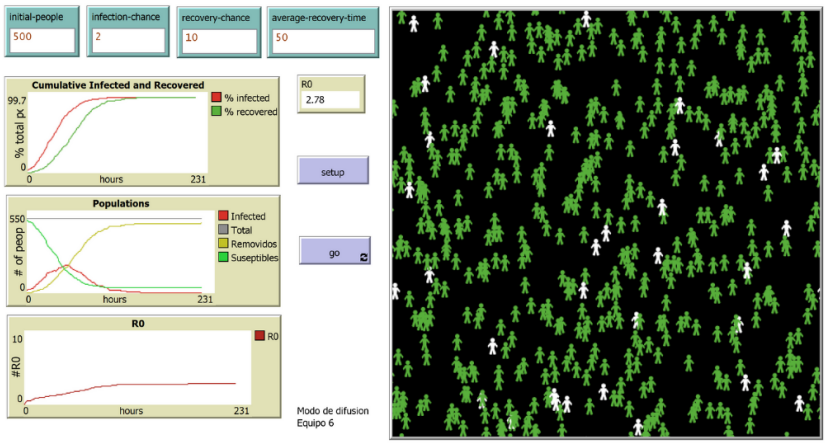


Figura 1: Ejemplo de una simulación basado en agentes de una pandemia.

Modelación de una pandemia usando algortimo de Doob-Gillispie. Este algoritmo consiste en asignar una probabilidad en la que ocurrirá los siguientes eventos con la siguiente expresión matemática,

$$p(\tau, j|x,t) = a_j(x)\exp\left(-\tau\sum_j a_j(x)\right), \tag{6}$$

en donde a_j es la función de densidad de probabilidad que ocurran los eventos, mientras que x es un vector con la cardinalidad de los eventos en momento t y τ es el diferencial temporal con distribución exponencial(*6*).

Aplicando este algoritmo para simular una pandemia, los eventos a_j son lo compartimentos de Susceptibles, Recuperados e Infectados y el diferencial temporal estará dado por,

$$\tau \sim \text{Exponencial}\left(\frac{1}{\beta S_t I_t + \gamma I_t}\right). \tag{7}$$

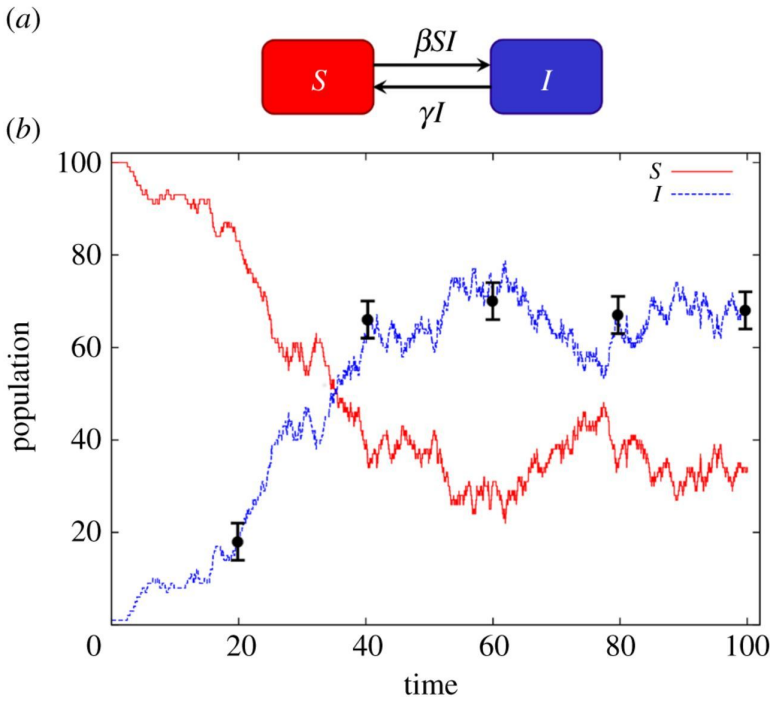


Figura 2: Ejemplo de una implementación del algoritmo Doob-Gillispie para simular una pandemia, realizado por (*3*)

Acerca de redes neuronales Para realizar una red neuronal se tienen que tomar en cuenta 3 cosas esenciales, entre ellas la entrada de datos, las neuronas ocultas y la salida. En las redes neuronales se ingresan las entradas y al pasar por funciones de activación que se encuentran en cada neurona se obtiene una salida. A partir de la salida que se obtiene se compara con la salida esperada y se busca corregir los parámetros dentro de las neuronas, de esté modo se empieza a entrenar a la red neuronal de modo que uno es capaz de ingresar entradas distintas a las utilizadas en el entrenamiento y encontrar una salida que se espera que sea correcta.

`https://github.com/Roy3838/Estocasticos.git`

Acerca de algoritmos genéticos Los algoritmos genéticos se utilizan principalmente para encontrar los mínimos globales de una función objetivo. Esto se logra a partir de generar posibles soluciones aleatorias (padres) y con ello generar los llamados hijos de un par de padres. Estos hijos se generan a través de cruzar a los padres, es decir, los datos de entrada de tal modo que de 2 padres se obtienen 2 hijos. Al seleccionar a un hijo de manera aleatoria a este se le muta para obtener valores distintos, una vez mutado se prueba el valor de este hijo y

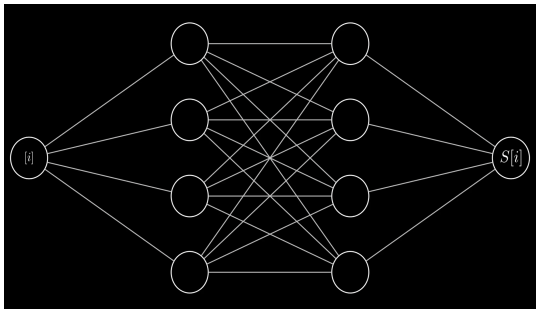


Figura 3: Representación

se reemplaza a un padre de las posibles soluciones iniciales, de este modo al repetir el proceso una cantidad suficiente de veces se pueden obtener soluciones aproximadas al mínimo global de la función objetivo.

1.2. Datos de una epidemia contaminados con ruido

En esta sección se mostrara el proceso y los resultados de crear y entrenar una red neuronal con el objetivo de realizar una predicción del parámetro R_o, dado un conjunto de información que modela el comportamiento de susceptibles, infectados y recuperados.

Diseño de la red neuronal de clasificación. ...La entrada de la red neuronal es una matriz de 3×100, en donde los renglones representan las variables Susceptibles, Infectados y Recuperados, mientras que las columnas son las iteraciones temporales. El resultado de la red neuronal es un vector con porcentajes de coincidencia a cada enfermedad.

Particularmente para esta red neuronal se utilizo una de tipo 'patternet' que se enfoca principalmente en reconocimiento de patrones. Además se utilizo el entrenamiento de gradiente descendiente conjugado, 2 capas ocultas de 4 neuronas cada una y funciones de activación logarítmicas sigmoides para las 2 capas ocultas y una función lineal para la de salida.

Diseño de la red neuronal de suavizado. Para la red neuronal de suavizado se utiliza únicamente los datos con ruido para entrenar a la red neuronal, en este entrenamiento se ingresa la parte temporal y se esperan 3 valores de salida correspondientes a cada componente del SIR. Para asegurar un suavizado se utilizó una red neuronal de 'feedforwardnet' con entrenamiento del tipo Levenberg-Marquardt, esté tipo de entrenamiento se enfoca en mínimos

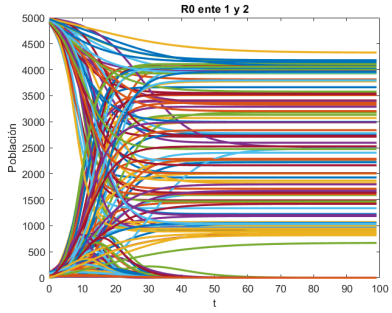
cuadrados no lineales. Al igual que la red neuronal de clasificación de enfermedades se utilizan 2 capas ocultas con 4 neuronas con funciones de activación del tipo logarítmicas sigmoide más la función lineal de salida.

Creación de la base de entrenamiento Para entrenar la red neuronal y obtener un vector de porcentajes de correlación con respecto a las enfermedades presentadas en la la tabla 1, se realizan 5000 simulaciones por rango de R_o-25 000 simulaciones en total-con una cantidad de 5000 personas, 100 iteraciones temporales y un diferencial temporal unitario. En la figura se pueden observar el conjunto de simulaciones categorizadas por rango de R_o.

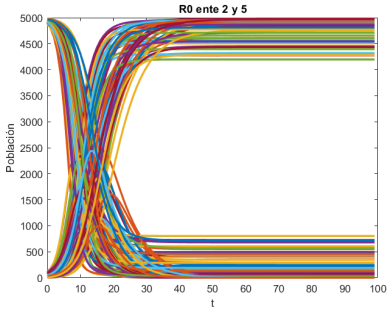
Resultados Una vez entrenadas las redes neuronales para suavizar funciones con ruido y clasificar las funciones de acuerdo a un porcentaje de correlación a una enfermedad, se introducen los datos dados a la red neuronal para suavizar la información, figura8, y posteriormente, las curvas suavizadas se introducen a la red neuronal para clasificar obteniendo una correlación del 99.15 % a Covid-19, en la tabla 2 se muestra el resto de correlaciones.

Porcentaje de correlación	**Enfermedad**
0.60 %	Gripe porcina
99.15 %	Covid-19
0.25 %	Paperas
0.00 %	Varicela
0.00 %	Sarampión

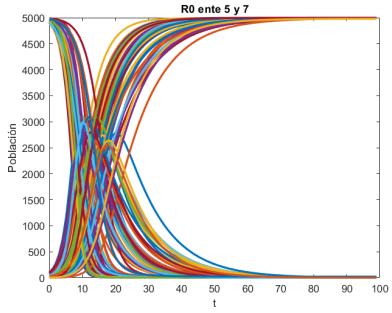
Tabla 2: Resultados de la red neuronal



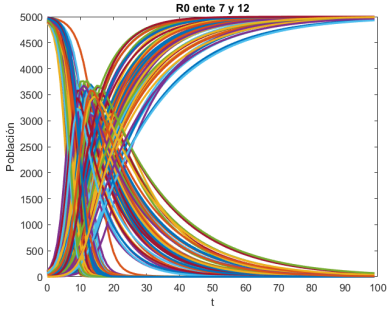
(a) 500 simulaciones con un factor $R_o \in [1,2]$



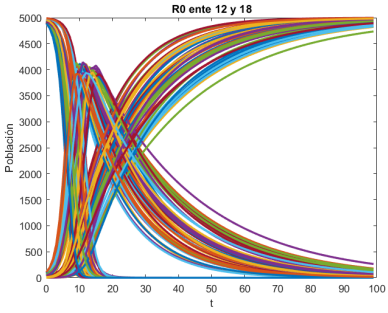
(b) 500 simulaciones con un factor $R_o \in [2,5]$



(c) 500 simulaciones con un factor $R_o \in [5,7]$



(d) 500 simulaciones con un factor $R_o \in [7,12]$



(e) 500 simulaciones con un factor $R_o \in [12,18]$

Figura 4: Muestra de 1 500 simulaciones de 25 000 para la base de datos de entrenamiento para la red neuronal, categorizada por rango de R_o, correspondientes a las enfermedades de interés.

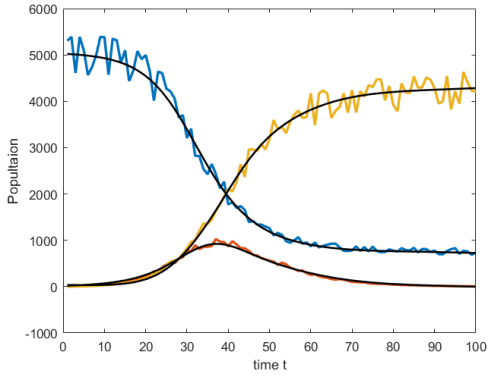


Figura 5: Caption

1.3. Modela de una epidemia con agentes

1.3.1. Caso a)

En esta sección se comparan simulaciones del modelo SIR implementado en NetLogo epiDEM Basic, con simulaciones hechas en MatLab, con el objetivo de analizar el comportamiendo de R_o. Es importante recordar que el R_o es el número de infecciones secundarias que surgen debido a la introducción de un solo infectado en una población totalmente susceptible, nosotros hemos definido los parámetros β y γ como valores fijos y Netlogo los define como tazas $\beta = \frac{I_{new}}{I_{old}}$ y $\gamma = \frac{R_{new}}{I_{old}}$ para finalmente definir nuestro esquemático $R_0 = \left(\frac{\left(ln(\frac{s_0}{s(t)}) \right)}{N - s(t)} \right) \cdot s_0$

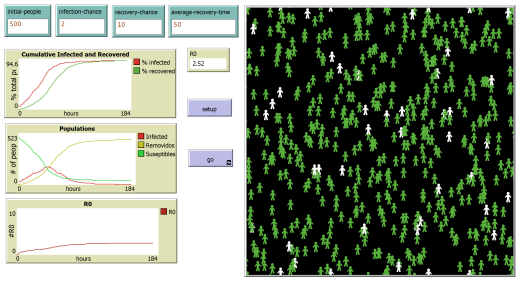
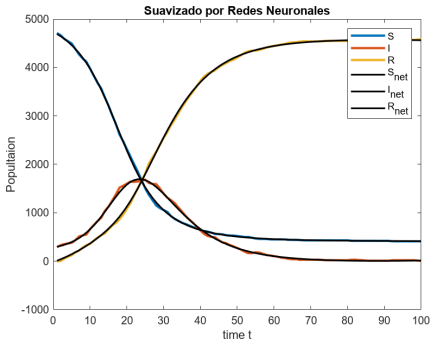


Figura 6: Initial people 500, infection chance 2, recovery chance 10 y average recovery time 50

Nuestros datos de simulacion para entranar nuestra red neuronal se basaron en la definicion de R_0 vista como $R_0 = \frac{\beta}{\gamma}$ por lo que comparamos los resultados obtenidos de nuestra simulacion de NetLogo con nuestra red neuronal suavizadora y clasifico el R_0 correctamente a la

categoría correspondiente $2-5$ como podemos apreciar en la Figura 6, en este caso podemos ver que hay congruencia con la definicion usada para entrenar y con la generada por el programa de NetLogo



(a) Comportamiento analizado y suavizado por la red neuronal

```
El R0 clasifica para:
rr0 = 5×3
    0.0006    1.0000    2.0000
    0.9915    2.0000    5.0000
    0.0078    5.0000    7.0000
    0.0000    7.0000   12.0000
    0.0000   12.0000   18.0000

El R0 Netlogo da 2.51
```

(b) R0 analizado por la red neuronal y clasificado a entre $2-5$

1.3.2. Caso b)

Ahora analizaremos los datos de una simulacion similar a la pasada pero con el cambio al parametro de recuperacion donde lo pasaremos de 10 a 90

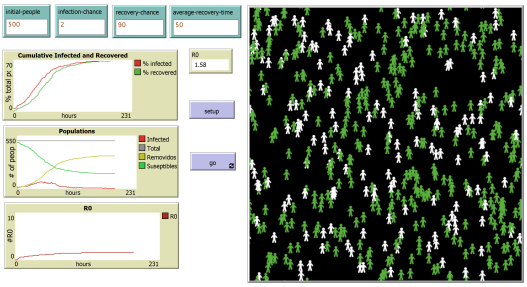
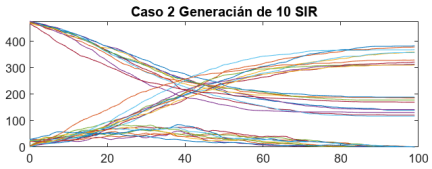
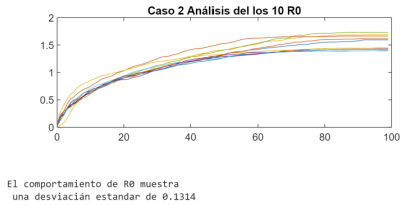


Figura 8: Initial people 500, infection chance 2, recovery chance 90 y average recovery time 50

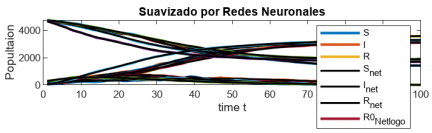
Con dichos parametros procedimos a generar 10 corridas y analizar los datos en busca de ver el comportamiento de desviación estándar existente en los datos generados por la red R_o



(a) 10 sumulaciones SIR con parametros de caso 2



(b) R0 calculado por NetLogo, desviacion calculada por Malab a $\sigma = 0.1314$



(c) 10 casos analizados uno a uno por la red

(d) Clasificación en 1 para todos los casos por la red

Para todos los casos del modelo SIR con parámetros descritos, la desviación estándar fue de 0.1314 con una predicción certera las 10 veces clasificando del R_0 entre $1-2$ por lo que los datos arrojados por nuestro método concuerdan con los datos generados por NetLogo para este caso también.

1.4. Datos de una epidemia con modelación estocástica

En esta sección se analizan datos obtenidos a través de una simulación con el algoritmo Doob-Gillipsie (figura10) para obtener el valor de R_o y realizar una correlación con las enfermedades de interés, tabla1.

Para esto se realiza un acondicionamiento de datos usando un promedio móvil, con una cardinalidad de 250 elementos por subconjunto (figura11a) y se utiliza la red neuronal para suavizar las curvas,figura11b y poder introducirlas en la red neuronal de correlación de enfermedades.

Resultado De acuerdo con la red neuronal implementada, se tuvo una correlación de 99.79 % en Sarampión y 0.21 % en Varicela.

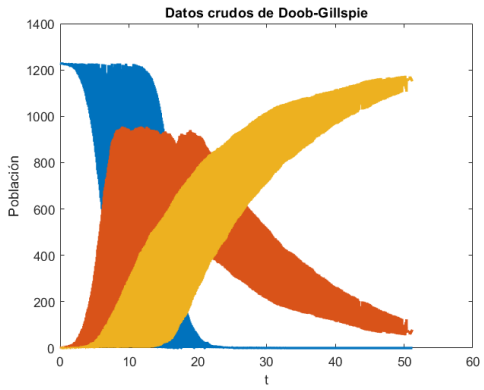
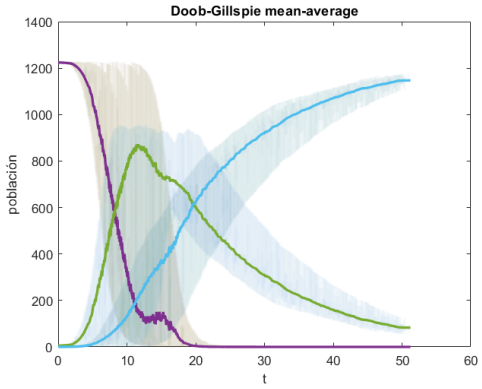
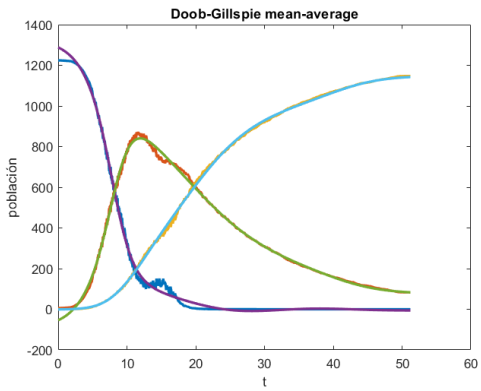


Figura 10: Datos dados a partir de una simulación con el algoritmo de Doob-Gillispie.



(a) Resultado de aplicar el promedio móvil con cardinalidad de 250 elementos por subconjunto.



(b) Suavizado de las curvas usando una red neuronal a partir de los promedios móviles.

Figura 11: Acondicionamiento de los datos.

Porcentaje de correlación	Enfermedad
0 %	Gripe porcina
0 %	Covid-19
0 %	Paperas
0.21 %	Varicela
99.79 %	Sarampión

Tabla 3: Resultado sin hacer la media móvil

1.5. Confirmación de los datos con Algoritmos Genéticos

Finalmente se hizo un algoritmo genético que optimizara para encontrar los parámetros de $\alpha y \beta$, para esto se propone una función de optimización, que en base

$$\sum_{i=0}^{N} \left[(S_i - S_{i-1} - \beta \ S_{i-1} \cdot I_{i-1})^2 + (I_{i-1} - I_{i-1} + \beta S_{i-1} I_{i-1} - \gamma I_{i-1})^2 + (R_i - R_{i-1} + \gamma I_{i-1})^2 \right]$$

Utilizando el Toolbox de Python de optimización, se quiere minimizar una función que esta definida de la siguiente manera.

Corriendo el algoritmo genético sobre la función previamente mencionada se llega al resultado de β y γ que confirma el R_0 previamente propuesto.

```
 Objective function:
 1214980885.5650804
r0 : 2.507179320389672
```

El valor de r0 obtenido fue de 2.5 lo que nos confirma el resultado previamente obtenido con la red del SIR.

2. Conclusión

La conclusión de este trabajo de investigación es que, mediante el uso de una red neuronal y un algoritmo genético, se ha logrado realizar una predicción precisa del parámetro R_o, que es

un indicador importante en el modelo SIR para evaluar la propagación de enfermedades infecciosas. Los resultados obtenidos muestran una alta correlación con Covid-19 y confirman el valor de R_o previamente calculado. Esto demuestra la utilidad de esta metodología en la predicción y análisis del comportamiento de enfermedades infecciosas.

Una de las limitaciones de este trabajo es que se ha realizado con una base de datos limitada y puede no generalizarse a otras enfermedades o situaciones. Además, la precisión de la predicción puede variar dependiendo de la calidad de la información de entrada y de la exactitud de los parámetros utilizados en el modelo SIR. También es importante mencionar que el algoritmo genético utilizado puede no ser el más óptimo para optimizar los parámetros de α y β en todos los casos.

3. Reflexiones individuales

Gilberto En este reto pude comprender mejor los conceptos del modelo SIR y la utilidad de los procesos estocásticos para tener aproximaciones o ideas de comportamientos que no son posibles de manipular de manera determinística por la cantidad tan masiva de variables que entran en juego. Además fue interesante realizar esto en su mayoría a través de redes neuronales, el proceso de pensamiento y desarrollo fue divertido al punto de generarme más entusiasmo por realizar trabajos que conlleven el uso de estas herramientas tan útiles.

Ricardo En este reto pudimos corroborar los datos experimentales de simulaciones de diversas fuentes con nuestros algoritmos y redes diseñadas para para el modelo SIR, logrando una precisión sorprendente en nuestras predicciones. Fue emocionante poder ver cómo nuestro modelo podía predecir con un alto grado de certeza el comportamiento de una enfermedad en una población determinada. A través de este proyecto, aprendimos a utilizar herramientas matemáticas y estadísticas para construir un modelo que pueda predecir el comportamiento de una enfermedad en una población. También aprendimos a trabajar en equipo y a colaborar para alcanzar nuestros objetivos. En resumen, el proyecto final de la clase de Modelación de Sistemas Estocásticos fue una experiencia enriquecedora que nos permitió adquirir habilidades valiosas y profundizar en el conocimiento sobre el modelo SIR.

Fátima Quiero decir que en esta última etapa del reto pude comprender a profundidad lo que implica el modelo SIR estocástico, donde incluso el año pasado cuando más de un tercio de la población mexicana estaba vacunada vacunada por encima de 18 años las tasas de mortalidad eran altas, pero gracias a la evolución del tiempo y las medidas sanitarias, ya no existen muertes por COVID. De igual forma, este reto me hizo ser más consciente de lo que implica la pandemia y todos sus aspectos, no sólo de los decesos sino de la importancia de

estar vacunados la mayor parte de la población lo antes posible. Incluso me pareció muy interesante la comparación realizada con las otras epidemias ocurridas a lo largo de estos últimos años ya que podemos darnos una idea más clara y cuantificable de las consecuencias de una enfermedad con capacidad de propagarse a nivel mundial. Finalmente abordando el tema físico-matemático puedo decir que este reto me ayudó en el aprendizaje de redes neuronales y modelación de funciones e incluso algoritmos genéticos, por lo que me hubiese gustado tener un poco más de tiempo para comprender estos temas a profundidad, ya que como vimos en las últimas clases, estos temas no aplicables a cuestiones realistas donde incluso personas viven de esto en la bolsa de valores, así que gracias a los profesores por asesorarnos y apoyarnos en todo momento.

Rodrigo Como estudiante de la clase de Modelación de Sistemas Estocásticos, me siento satisfecho con el proyecto final en el que trabajamos con el modelo SIR. Este modelo nos permitió entender cómo se propagan enfermedades en una población y cuáles son las variables que influyen en su propagación.

La realización del proyecto nos llevó a investigar y profundizar en el tema, lo que me permitió adquirir nuevos conocimientos y habilidades en el área de modelación y análisis estadístico. Además, trabajar en equipo nos enseñó a colaborar y a compartir nuestras ideas para llegar a un resultado final satisfactorio.

El modelo SIR nos ayudó a entender cómo las medidas de prevención y control pueden influir en la propagación de una enfermedad y cómo se pueden utilizar para reducir su impacto en una población. Esto es de gran relevancia en tiempos de pandemia como los que estamos viviendo actualmente.

En conclusión, el proyecto final sobre el modelo SIR ha sido una experiencia enriquecedora que me ha permitido adquirir nuevos conocimientos y habilidades en el área de modelación estocástica y me ha dado una visión más clara sobre cómo se propagan las enfermedades en una población y cómo se pueden controlar.

4. Anexos

5. Código de algoritmos genéticos

```
from pathlib import Path
from geneticalgorithm import geneticalgorithm as ga
```

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import latexify


ROOT_DIR = Path(__file__).parents[2]

dataSIR = pd.read_csv(ROOT_DIR/ "c_aber" / "Data.csv")
dataSIR = dataSIR.to_numpy()
S_data = dataSIR[:,0]
I_data = dataSIR[:,1]
R_data = dataSIR[:,2]

# SIR funtion
@latexify.with_latex
def f(x):
    beta = x[0]
    gamma = x[1]
    sum = 0
    for i in range(99):
        sum += (S_data[i] - (S_data[i-1] - beta*S_data[i-1]*I_data[i-1]))*
            I_data[i] - (I_data[i-1] + beta*S_data[i-1]*I_data[i-1] - gamm
                R_data[i] - (R_data[i-1] + gamma*I_data[i-1]))**2
    return 1/sum


# ALGORITMOS GENETICOS
model=ga(function=f,dimension=2,variable_type='real',variable_boundaries=r
model.run()
output = model.output_dict
variable = output['variable']
r0 = variable[0]/variable[1]
print('r0 :' , r0)
```

Referencias

1. E. Weisstein, *Kermack-McKendrick model*, (https://mathworld.wolfram.com/Kermack-McKendrickModel.html).
2. F. Brauer, *Mathematical Biosciences* **198**, 119-131 (2005).
3. E. N. Bodine, R. M. Panoff, E. O. Voit y A. E. Weisstein, *Bulletin of Mathematical Biology* **82** (2020).
4. *Epidem basic*, (https://ccl.northwestern.edu/netlogo/models/epiDEMBasic).
5. *Agent-based modeling in python 3+¶*, (https://mesa.readthedocs.io/en/stable/).
6. *Gillespie algorithm*, nov. de 2022, (https://en.wikipedia.org/wiki/Gillespie_algorithm).
7. J. E. Gentle, *Random Number Generation and Monte Carlo Methods* (Springer, 2nd ed, 2003), ISBN: 0387001786; 9780387001784; 9780387216102; 0387216103, (libgen.li/file.php?md5=9d83e3a2f8087fe77ef20804f125af9b).
8. S. D. Frost, *The Journal of Open Source Software* **1**, 42, (https://doi.org/10.21105/joss.00042) (jul. de 2016).
9. D. T. Gillespie, *Annual Review of Physical Chemistry* **58**, PMID: 17037977, 35-55, eprint: https://doi.org/10.1146/annurev.physchem.58.032806.104637, (https://doi.org/10.1146/annurev.physchem.58.032806.104637) (2007).
10. C. M. Pooley, S. C. Bishop y G. Marion, *Journal of The Royal Society Interface* **12**, 20150225 (2015).