Trabajo de Fin de Máster
Máster Universitario en Ingeniería Industrial

Algoritmos estadísticos para el modelado de comportamientos sociales

Autor: Jaime María Moreno Sosa
Tutor: Teodoro Álamo Cantarero

Dpto. de Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2020

Trabajo de Fin de Máster
Máster Universitario en Ingeniería Industrial

Algoritmos estadísticos para el modelado de comportamientos sociales

Autor:
Jaime María Moreno Sosa

Tutor:
Teodoro Álamo Cantarero
Catedrático de Universidad

Dpto. de Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2020

Trabajo de Fin de Máster: Algoritmos estadísticos para el modelado de comportamientos sociales

Autor: Jaime María Moreno Sosa

Tutor: Teodoro Álamo Cantarero

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2020

El Secretario del Tribunal

A mi familia

A mis profesores

A mis amigos

Cuando emprendas tu viaje a Itaca pide que el camino sea largo, lleno de aventuras, lleno de experiencias – Ítaca, Konstantino Kavafis

Agradecimientos

Con este trabajo de fin de máster, mis estudios como ingeniero llegan a su fin; al menos por el momento. Ha sido un camino duro, pero apasionante. Ha habido momentos de dificultad, pero también de satisfacción. Momentos de frustración, pero también de ver recompensado el esfuerzo. En los momentos buenos y en aquellos no tan buenos, siempre ha habido personas que han estado conmigo, alentándome y sirviéndome de ejemplo para continuar mi camino: mi familia, en especial mis padres, mis amigos, sobre todo los que he hecho en la escuela –ya que ellos saben mejor que nadie qué es estudiar una ingeniería–, así como muchos profesores y mentores que han seguido mi progreso y han puesto todo de su parte para ayudarme; a todas aquellas personas, gracias. No quisiera terminar esta sección de agradecimiento sin dedicar mi especial gratitud a mi tutor Teodoro Álamo, quien me ha guiado durante la mayor parte del desarrollo del trabajo, así como a Fabrizio Dabbene y Chiara Ravazzi, del Politecnico di Torino, a quienes conocí durante mi estancia en Italia, donde comencé este proyecto, y que han sido fuente de inspiración para la temática de este trabajo y han sentado las bases de este con sus investigaciones previas. A todos ellos, muchas gracias.

Jaime María Moreno Sosa

Sevilla, 2020

Resumen

Este trabajo de fin de máster, realizado en parte durante la estancia del autor en el Politecnico di Torino (Turín, Italia) gracias a la colaboración de Fabrizio Dabbene y Chiara Ravazzi, recoge primeramente la implementación de un modelo para estudiar comportamientos sociales. El modelo consiste en la generación aleatoria de una red de individuos dentro de unos límites de latitud y longitud determinados, representando así una comunidad, ciudad, pueblo, etc. Una vez generada la red, se define una distancia máxima para que dos agentes sean considerados vecinos entre sí. Posteriormente, se procede a calcular los vecinos de cada agente de la red. Una vez realizados los pasos anteriores, se simula una dinámica de comportamientos sociales en la que, un conjunto de agentes que poseen inicialmente una característica determinada (por ejemplo, posesión de un producto, una opinión determinada o una enfermedad infecciosa) llamados adoptantes, influyen en sus vecinos para que adopten ellos también la característica en cuestión. La mayor o menor facilidad de un determinado agente para ser influido por sus vecinos (susceptibilidad) se modela con un parámetro denotado por α; el mismo representa la fracción de vecinos adoptantes que debe tener el agente en cuestión para proceder a la adopción de la característica estudiada.

Durante el trabajo, se implementan diversos cambios en el modelo, consiguiendo así una mejora del mismo en términos de realismo; en concreto, se permite la elección de una distribución estadística que gobierne la susceptibilidad (α) de los agentes (la cual se suponía igual para todos los agentes en el modelo inicial), se introduce una nueva variable que permite que, aun teniendo vecinos suficientes, un agente no proceda a la adopción en un instante determinado (debido por ejemplo a la falta de recursos para la adquisición de un producto), y finalmente se añade una variable más que permite que un agente pase de ser adoptante a no adoptante; fenómeno que en el modelo inicial no era contemplado.

Tras la adición de cada cambio descrito en el párrafo anterior al modelo de red, se implementó un algoritmo de máxima verosimilitud que permite el cálculo del parámetro introducido, así como de los que se introdujeron anteriormente, llegando finalmente a obtener un algoritmo que permite calcular la probabilidad de decadencia (pasar de adoptante a no adoptante), la probabilidad de éxito en la adopción (proceder o no a la adopción una vez se ha alcanzado el número de vecinos necesario en función de la susceptibilidad α), y los parámetros de la distribución estadística utilizada para generar los distintos valores de susceptibilidad (α) de los agentes de la red.

En todos los capítulos donde se explica una determinada implementación en Matlab, se incluyen imágenes de gráficas y resultados obtenidos que permiten comprobar el correcto funcionamiento de los algoritmos.

Al final del documento se presentan las conclusiones obtenidas mediante el desarrollo del trabajo y algunas ideas sobre mejoras futuras en el modelo y algoritmos.

Abstract

This final master's degree project, developed partially during its author stay at the Politecnico di Torino (Turin, Italy) with the important collaboration of Fabrizio Dabbene and Chiara Ravazzi, comprises firstly the implementation of a model to study social behaviors. The model consists on the random generation of a network of individuals within certain limits of latitude and longitude, mimicking the population of a city or town. After the network has been generated, a maximum distance is defined so that two agents can be considered neighbors of one another. Afterwards, a social behavior dynamic is simulated in which, a set of agents which initially posess a certain characteristic (like, for example, possession of a product, a certain opinion or an infectious disease) named adopters influence their neighbors so they can also adopt the characteristic being studied. The easiness or difficulty that a certain agent has of being influenced by their neighbors is modelled using a parameter denoted with α. This parameter represents the fraction of adopting neighbors that the studied agent needs to have in order to proceed with the adoption of the aforesaid characteristic.

During this project, several changes are implemented on the model, resulting on an improvement in terms of realism; more specifically, the election of a certain statistical distribution that governs the susceptibility (α) of the agents (which was initially supposed equal for every agent), the introduction of a new variable that allows that even if an agent has enough neighbors to be a new adopter, he will not be able to proceed with the adoption on the present iteration (due to, for example, lack of resources to acquire a product), and finally, the addition of a variable that allows an agent to change from adopter to non-adopter, which is something that was not allowed in the first model.

After the addition of each change described on the previous paragraph, a maximum likelihood algorithm that allows the estimation of the newly introduced parameter, along with the ones that were previously introduced was implemented, obtaining towards the end of the project an algorithm that makes possible the estimation of the proability of decadence (changing from adopter to non-adopter), the probability of success on the adoption (whether or not an agent proceeds with the adoption once the number of necessary adopting neighbors has been reached), and the parameters of the statistical distribution used to generate the values of susceptibility (α) of every agent in the network.

In every chapter where a certain Matlab implementation is explained, graphs and results are shown to verify the correct behavior of the algorithms

Finally, the conclusions obtained during the development of the project are shown, along with some ideas for future improvements on the model and algorithms.

Índice

Índice De Tablas

Índice De Figuras

Figura 7.7. Ejecución para λ=9 p_éxito=0.6,p_decadencia=0.01 70

1 INTRODUCCIÓN Y MOTIVACIÓN

El presente trabajo tiene como objetivo la aplicación de algoritmos estadísticos para modelar diferentes comportamientos sociales. En las primeras fases, se programará en Matlab una dinámica de red que modela la propagación de una determinada opinión o evento a través de los agentes (individuos) que componen la misma. La propagación de la opinión o del evento se producirá mediante la influencia que tienen los vecinos (agentes que se encuentren a una distancia relativamente cercana de otro) en un agente determinado. Cada agente tendrá asociado un parámetro, denotado con α, que modela la susceptibilidad de dicho agente a ser influenciado por sus vecinos; en concreto, en el modelo utilizado en este trabajo, el parámetro α representa el tanto por uno de vecinos que han adoptado un determinado cambio (compra de producto, opinión, contracción de una enfermedad, etc.) a partir del cual un agente se convertirá también en adoptante. Un breve ejemplo de problema que puede estudiarse con el modelo de red implementado en este trabajo es la compra de vehículos eléctricos por parte de individuos de una determinada comunidad (ciudad, población, etc.). En el modelo existirán inicialmente determinados agentes que poseen el vehículo eléctrico, y estos influirán a sus vecinos (agentes situados en sus cercanías). Esta influencia se modela gracias al parámetro α que, como se ha explicado previamente; dicho parámetro representa el tanto por uno mínimo de vecinos que usan vehículos eléctricos que un agente debe tener para convertirse el mismo en adoptante, es decir, para comprar un vehículo eléctrico.

En el artículo [1] (*Social network analysis of electric vehicles adoption: a data-based approach,* autores: V. Breschi, M. Tanelli, C. Ravazzi, S. Strada y F. Dabbene), el cual conoció el autor de este trabajo gracias a diversos encuentros con los autores del artículo durante su estancia en el Politecnico di Torino (Turín, Italia), el parámetro α se elegía arbitrariamente y se asumía igual para todos los agentes que integraban la red; una vez se implementó el modelo descrito en dicho artículo en Matlab, se realizó una importante modificación de este permitiendo que el parámetro α de los diferentes agentes que componen la red siga una distribución estadística determinada, obteniendo así un modelo más representativo de la realidad, ya que de este modo pueden existir en la red individuos más susceptibles que otros, permitiendo además, mediante la elección del parámetro (o parámetros) característico de la distribución estadística empleada, conseguir un mayor o menor número de individuos con un valor alto o bajo de α.

Una vez se realizó la modificación antedicha en el modelo de red elegido como base del trabajo, se planteó y se resolvió el problema inverso: Dada una dinámica determinada (esto es, historial de cada agente que recoja el estado [adoptante o no adoptante] en el que se encuentra), ¿cuáles son los parámetros característicos de la distribución estadística que mejor modelan dicha dinámica? Dicho problema se resolvió mediante la aplicación de un algoritmo de máxima verosimilitud que otorga como solución los parámetros que maximizan la probabilidad de obtener la dinámica observada.

Posteriormente, para otorgar más realismo al modelo, se realizó una modificación del mismo añadiendo una variable adicional que permitiera que, a pesar de que el tanto por uno de vecinos adoptantes fuera igual o superior al valor del parámetro α de un agente determinado, dicho agente no realizara la adopción. Esta variable puede modelar diversos fenómenos en la realidad; por ejemplo, si la dinámica de red pretende modelar el proceso de compra de un vehículo eléctrico, puede darse la posibilidad de que el agente estudiado no posea el dinero necesario para realizar la compra a pesar de que la influencia por parte de sus vecinos haya sido suficiente. Una vez realizada la modificación del modelo de red, se implementó otro algoritmo de máxima verosimilitud para obtener tanto los parámetros de la distribución estadística que define el valor del parámetro α de los diferentes agentes, como el valor de la variable de probabilidad de éxito en la adopción. Este problema se resolvió de dos formas diferentes, con el objetivo de que la segunda mejorara el tiempo de cálculo de la primera.

Por último, con el objetivo de hacer que el modelo de red tuviera un uso más amplio, se modificó

dicho modelo para permitir que un determinado agente cambiara de estado desde adoptante hasta no adoptante. Esta modificación permite el uso de la dinámica de red en problemas para los que anteriormente no era válida, como por ejemplo modelar los contagios y las curas de una enfermedad (un individuo puede contraerla, pero al cabo de un tiempo puede curarse y dejar de ser un individuo contagioso). Adicionalmente, se programó un algoritmo de máxima verosimilitud que permite obtener el valor de los parámetros característicos de la distribución, la probabilidad de éxito en la adopción y la probabilidad de decaer de un determinado agente adoptante.

La motivación del trabajo es lograr un modelo relativamente sencillo para describir comportamientos sociales permitiendo a su vez su adaptación a diferentes situaciones mediante el ajuste de los diferentes parámetros de los que se compone el modelo, así como la obtención de parámetros que permitan simular con el modelo una dinámica ya observada para su posterior estudio, análisis y tratamiento de datos.

A continuación, se presentan algunos ejemplos de situaciones en los que podría usarse el modelo y los algoritmos desarrollados en este trabajo;

- Proceso de compras de vehículos eléctricos en una ciudad: este era el objetivo del artículo [1] que sirvió de inspiración para el presente trabajo. Es sabido que, con el avance de la tecnología y la mayor conciencia ambiental de la población, el mercado de vehículos eléctricos está creciendo rápidamente [2], aunque siguen existiendo factores que generan cierto escepticismo a los potenciales compradores de un vehículo eléctrico tales como su elevado precio o la dificultad para cargar sus baterías con respecto a la facilidad de repostar en un vehículo de combustión [3]. Estos obstáculos en el proceso de adopción pueden modelarse mediante la variable que define la probabilidad de éxito en la adopción descrita anteriormente. En la figura 1.1, incluida al final de este capítulo, se puede observar el rápido aumento en las ventas de vehículos eléctricos en los últimos años en regiones altamente pobladas. El hecho de que cada vez haya más usuarios de estos vehículos implica que sus experiencias positivas podrán influir a un número creciente de individuos en su entorno para inspirarles a adquirir un vehículo eléctrico cuando decidan renovar su medio de transporte.

- Distribución de una determinada opinión o noticia a través de una red social. El mundo actual, especialmente en los países desarrollados, está ampliamente influido por las redes sociales, no obstante, aún existen personas que hacen un uso muy moderado de las mismas. Del mismo modo, también existen individuos que las usan varias horas al día. El modelo de este trabajo podría utilizarse para estimar la rapidez con la que una determinada noticia se propagará a través de la red, o para predecir la influencia que tendrá el entorno de un determinado usuario en la opinión de este respecto a un tema político, la valoración de un producto, etc.

- Propagación de una enfermedad. Como se explicó anteriormente, el modelo desarrollado en el trabajo también contempla la posibilidad de que un determinado individuo pase de ser adoptante a no adoptante. Esto lo hace muy apropiado para el modelado de una enfermedad: si se conoce la susceptibilidad (en forma de distribución estadística) y el tiempo aproximado que dura la enfermedad o el estado de persona infecciosa, se puede emplear el modelo para prever como afectará la enfermedad a la población de una determinada comunidad. Asimismo, también puede utilizarse el modelo para, a partir de los resultados obtenidos con el modelo, identificar los parámetros estadísticos que modelan la susceptibilidad de los individuos a contraer la enfermedad, el tiempo de recuperación, etc.

Como puede observarse, una gran variedad de fenómenos está íntimamente relacionados con comportamientos sociales. Los anteriores son solo algunos ejemplos de situaciones en las que se puede aplicar un modelo como el desarrollado en el trabajo, pero, sin duda, existen muchas más.

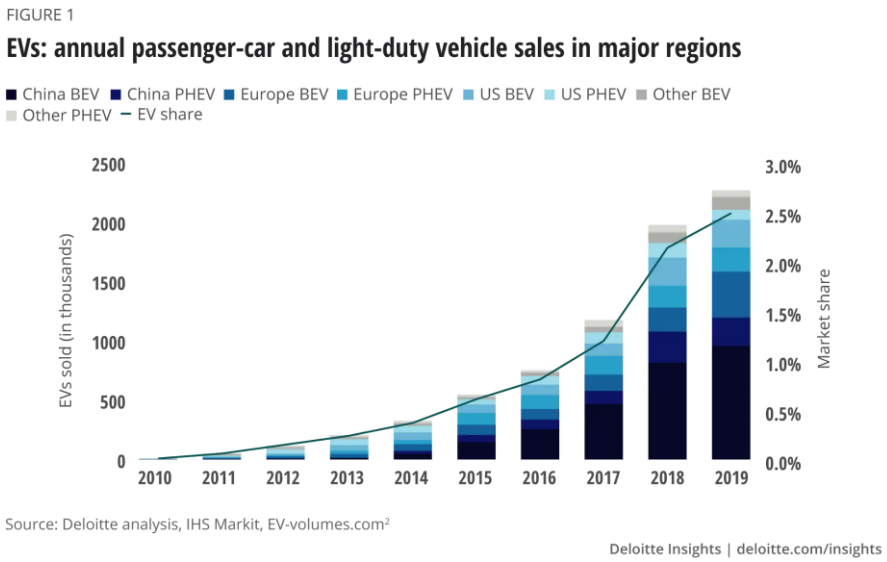


Figura 1.1. Evolución de las ventas en vehículos eléctricos [4]

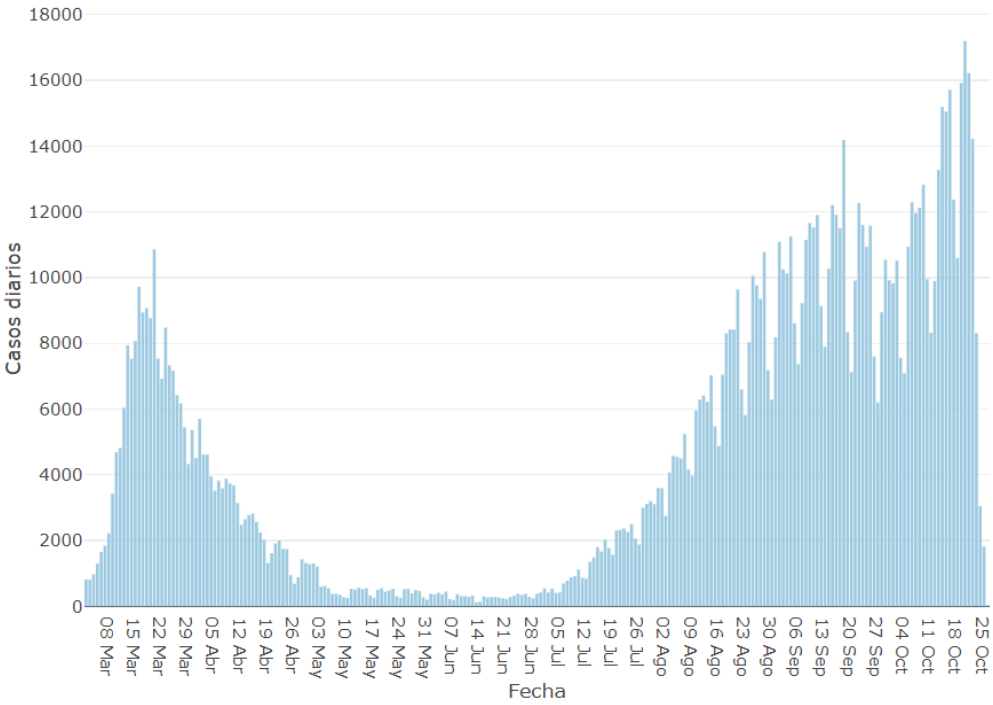


Figura 1.2. Nuevos contagios diarios de Covid19 en España. [5]

Para concluir este capítulo introductorio, se muestra una breve descripción de los capítulos que componen este trabajo:

1. **Introducción y motivación:** se explican los objetivos del trabajo, ejemplos de problemas que se resolverán y breve explicación del modelo empleado en las simulaciones
2. **Algoritmos de máxima verosimilitud:** se explican de forma breve los aspectos históricos más importantes del método de la máxima verosimilitud y su funcionamiento y objetivos. También incluye un ejemplo sencillo de su aplicación.
3. **Implementación del modelo de red:** se describe en mayor detalle el modelo empleado en el trabajo y su implementación en Matlab, así como diversas simulaciones que permiten observar diferentes dinámicas de comportamientos sociales.
4. **Algoritmo de máxima verosimilitud** – Estimación de parámetros característicos de la distribución probabilística: en este capítulo se explica la modificación del modelo de red empleado para permitir el uso de una distribución estadística para determinar los valores del parámetro α de los agentes de la red, así como la implementación de un algoritmo de máxima verosimilitud para obtener los valores de los parámetros característicos de la distribución a partir de los datos de una simulación.
5. **Algoritmo de máxima verosimilitud – Estimación de la probabilidad de éxito en la adopción:** este capítulo sigue una estructura parecida al anterior; primeramente, se modifica el modelo añadiendo una variable adicional que permite que, incluso teniendo los vecinos necesarios, un agente no se convierta en adoptante. Asimismo, se detalla la implementación de un algoritmo de máxima verosimilitud para obtener los valores de los parámetros característicos de la distribución y de la probabilidad de éxito en la adopción a partir de una simulación.
6. **Algoritmo de máxima verosimilitud – Estimación de la probabilidad de éxito en la adopción sin uso de combinatoria:** En el capítulo 6 se explica una mejora del algoritmo implementado en el capítulo anterior con el objetivo de acortar los tiempos de simulación y simplificar los cálculos de la probabilidad de observación.
7. **Algoritmo de máxima verosimilitud con cambio de opinión de los agentes:** este capítulo añade una nueva modificación al modelo empleado: la posibilidad de pasar de adoptante a no adoptante. Asimismo, se implementa un algoritmo de máxima verosimilitud para obtener, además de los parámetros que se obtenían en los capítulos anteriores (aquellos de la distribución estadística y la probabilidad de éxito), el valor de la variable que representa la probabilidad de decadencia, es decir, pasar de adoptante a no adoptante.
8. **Conclusiones y mejoras futuras:** En él se recogen las conclusiones más importantes obtenidas durante el desarrollo del trabajo y las simulaciones realizadas, y se introducen algunas posibles mejoras tanto del modelo de comportamiento social como de los algoritmos de máxima verosimilitud.
9. **Bibliografía:** título, autores y enlaces de las fuentes de información citadas en el documento
10. **Anexos:** códigos de Matlab empleados en las simulaciones

2 ALGORITMOS DE MÁXIMA VEROSIMILITUD

El siguiente capítulo tiene como objetivo recopilar algunos de los aspectos y fundamentos más importantes de la estimación por máxima verosimilitud, principal algoritmo estadístico utilizado en este trabajo. En concreto, se resumirán algunos aspectos históricos de su proposición y su uso, posteriormente se explicará de forma general su funcionamiento, y por último se expondrá un ejemplo de un problema sencillo resuelto mediante este método.

2.1. Aspectos históricos

El método de máxima verosimilitud fue formalmente propuesto por el estadista y biólogo británico Sir Ronald Aylmer Fisher entre los años 1912 y 1922 [6]. Sin embargo, se conocen usos del método anteriores a Fisher, incluso de matemáticos tan reconocidos como Gauss, pero fue Fisher el primero en otorgarle el nombre "máximum likelihood estimate" [6], que en español se traduce como "estimación por máxima verosimilitud".

Además de introducir el nombre con el que se conoce actualmente el método, Fisher también estudió en profundidad la estimación por máxima verosimilitud y contribuyó ampliamente a su popularización y uso entre la comunidad científica de la época [7].

Figura 2.1. Ronald Fisher, padre del método de la máxima verosimilitud [7]

A pesar de su gran contribución a la popularización y estudio del método, Fisher no logró la demostración formal del mismo; fue el matemático estadounidense Samuel S. Wilks quien la consiguió en el año 1938 mediante su teorema homónimo (Teorema de Wilks) [7].

Desde que se logró su demostración, el método de la máxima verosimilitud ha sido ampliamente utilizado en multitud de aplicaciones en las que se requiere la estimación de los parámetros de una función probabilística.

2.2. El método de la máxima verosimilitud

La estimación por máxima verosimilitud tiene por objetivo obtener los parámetros que gobiernan un modelo determinado. El método obtiene el valor de estos parámetros eligiendo aquellos que maximizan la probabilidad de que se obtengan los datos que han sido observados [8].

El funcionamiento del método de la máxima verosimilitud se explica a continuación [7]:

1. Se parte de un conjunto de variables $x = \{x_1, x_2, \dots, x_n\}$ aleatorias que siguen una determinada función de densidad que depende de un conjunto de parámetros $\theta = \{\theta_1, \theta_2, \dots, \theta_n\}$

2. Se tiene también una observación (esto es, historial del valor que adoptan las variables $\{x_1, x_2, \dots, x_n\}$) cuya probabilidad de obtención se quiere maximizar.

3. Se calcula una función de verosimilitud (en inglés likelihood function) que, a partir de los valores de las variables (x) y de los parámetros de la función de densidad (θ) otorgue la probabilidad de que haya ocurrido lo que se ha observado inicialmente.

4. Habitualmente se trabajará con el logaritmo de la función antedicha. Esto puede realizarse ya que la función logarítmica es estrictamente monótona creciente. Gracias al uso del logaritmo, los productorios necesarios para calcular la probabilidad se convierten en sumatorios.

5. Se calcula el máximo de la función de verosimilitud. Esto puede hacerse de forma numérica, o, en algunas ocasiones, derivando dicha función respecto a los parámetros θ e igualando el resultado a cero para obtener los valores de los parámetros θ_i.

6. El conjunto de parámetros $\theta = \{\theta_1, \theta_2, \dots, \theta_n\}$ que maximicen la función de verosimilitud corresponden a la solución del problema.

La aplicación del método de la máxima verosimilitud al modelo utilizado en este trabajo se explicará posteriormente en los diferentes capítulos dedicados a esta materia, no obstante, con el objetivo de facilitar la asimilación de los conceptos recién introducidos, se expone a continuación un ejemplo de aplicación del algoritmo para resolver un problema sencillo.

2.3 . Ejemplo de aplicación del método de máxima verosimilitud. Moneda trucada.

El presente ejemplo tiene como objetivo la aplicación del método de la máxima verosimilitud para calcular la probabilidad de obtener cara en un lanzamiento de moneda trucada.

Un lanzamiento de moneda puede dar lugar a dos resultados posibles: cara o cruz. Si se lanza la moneda n veces, se obtendrá un número x de veces en los que el resultado fue cara, y un número n-x de veces en los que el resultado fue cruz. De esta manera, para una secuencia de n lanzamientos, la probabilidad de que ocurra lo que se ha observado será la siguiente:

$$p = p_1 \cdot p_2 \cdot p_3 \cdot \dots \cdot p_n \tag{2.1}$$

Siendo $p_i = p_{cara}$ si el resultado del lanzamiento ha sido cara, y $p_i = p_{cruz}$ si el resultado ha sido cruz.

De este modo, la ecuación (2.1) es la función de verosimilitud que se deberá maximizar.

Para simular el problema de la moneda trucada, en primer lugar, se programó un sencillo script de Matlab que, dada una determinada probabilidad de obtener cara, y un número n de lanzamientos, obtuviera una secuencia que mostrara los resultados obtenidos en cada lanzamiento. Dicha secuencia

es un vector de dimensión n, en el que cada componente puede valer 1 en el caso de que el resultado haya sido cara, o 0 en el resultado de que haya sido cruz.

El script programado puede verse en la imagen siguiente:

```
p_cara=0.7;
n=100;
secuencia=zeros(n,1);

for i=1:n
    x=rand;
    if x<p_cara
        secuencia(i)=1;
    else
        secuencia(i)=0;
    end
end
```

Figura 2.2. Generación de secuencia de lanzamientos de moneda

Como puede observarse, el funcionamiento es muy sencillo. Se define una probabilidad de cara, un número de lanzamientos, y posteriormente se genera un número aleatorio uniforme entre 0 y 1. Si el número obtenido es menor que la probabilidad de cara, el lanzamiento corresponde a una cara, y si no lo es corresponde a una cruz. Este proceso se repite n veces para obtener así el vector secuencia. Para el ejemplo expuesto en este apartado, se utilizó un valor de p_cara de 0.7.

Una vez obtenida la secuencia de lanzamientos, se programó el algoritmo de máxima verosimilitud. El script desarrollado se muestra a continuación debido a la sencillez del problema:

```
n=length(secuencia);
p_vector=[];

for p_cara=0:0.01:1
    p_cruz=1-p_cara;
    p=0;
    for i=1:n
        if secuencia(i)==1
            p=p+log(p_cara);
        else
            p=p+log(p_cruz);
        end
    end
    p_vector=[p_vector;p];
end

p_cara_vector=[0:0.01:1];
figure
plot(p_cara_vector,p_vector);
```

Figura 2.3. Algoritmo de máxima verosimilitud aplicado a la moneda trucada

El funcionamiento sigue el esquema explicado en el apartado anterior sobre el método de la máxima verosimilitud. Se realiza un barrido con el parámetro p_cara entre 0 y 1 con un paso de 0.01. Para cada valor del parámetro se calcula la probabilidad de que ocurra lo que se ha observado (información que

se encuentra guardada en el vector secuencia) y por último se representa una gráfica en la que se muestre el valor del logaritmo de la probabilidad de la observación para cada valor de p_cara. La gráfica obtenida (para un número de lanzamientos n de 100) es la siguiente:

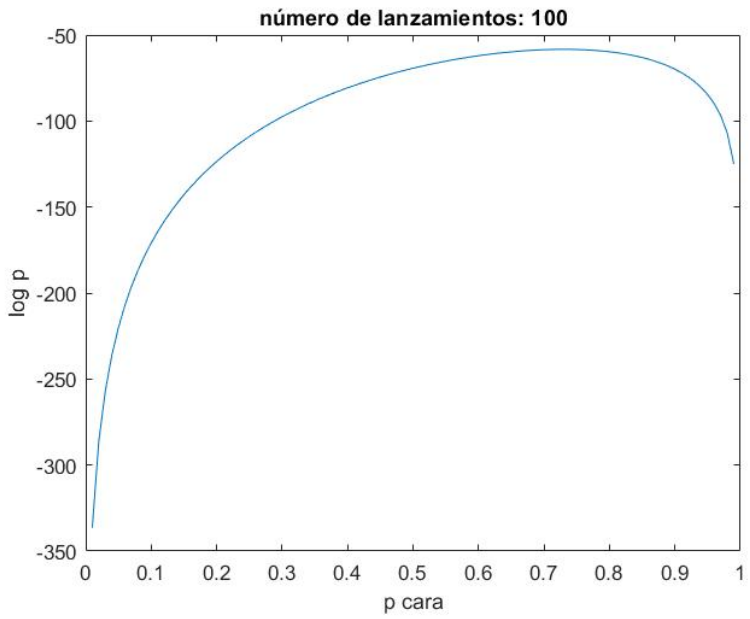


Figura 2.4. Logaritmo de la probabilidad de observación en el problema de la moneda trucada

Como se explicó en el apartado anterior, la solución del problema de máxima verosimilitud es el parámetro que maximiza la probabilidad de la observación; es decir, el máximo de la gráfica anterior, en este caso, el valor de p_cara que maximiza la probabilidad es 0.75.

Sin embargo, el valor de p_cara utilizado para la simulación de los lanzamientos de la moneda, fue 0.7. Esta diferencia entre el valor real y el valor obtenido mediante el algoritmo de máxima verosimilitud se debe al siguiente motivo: el método de la máxima verosimilitud no otorga necesariamente una solución exacta del problema, sino que su solución depende de la observación de referencia. Para aumentar la precisión del algoritmo de máxima verosimilitud, una buena solución es incrementar el número de muestras; en este caso, incrementar el número de lanzamientos. En las figuras que se muestran a continuación puede observarse como, al aumentar el número de lanzamientos, la solución del problema otorgada por el algoritmo de máxima verosimilitud se acerca cada vez más al valor real de p_cara (0.7). A la hora de emplear algoritmos de máxima verosimilitud en problemas reales, no siempre se dispondrá de un número de datos a elección, por lo que la calidad de los resultados obtenidos al aplicar el algoritmo dependerá a su vez de la calidad y cantidad de los datos disponibles.

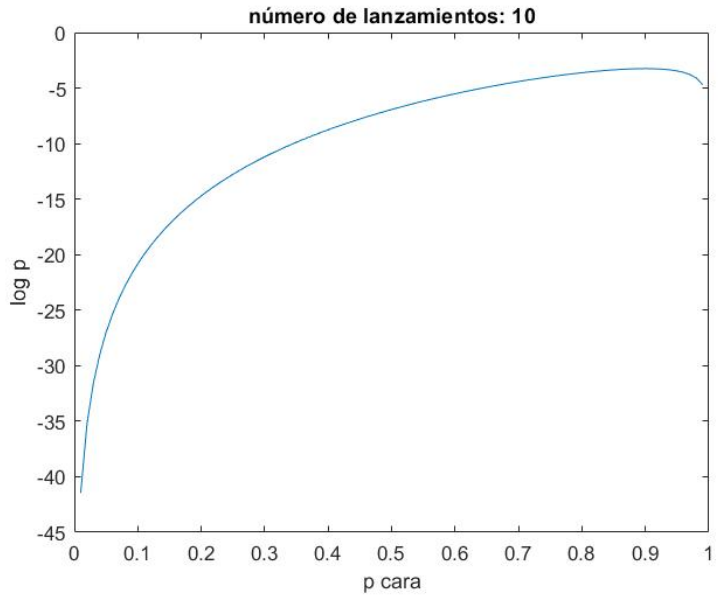


Figura 2.5. Gráfica de probabilidad para 10 lanzamientos

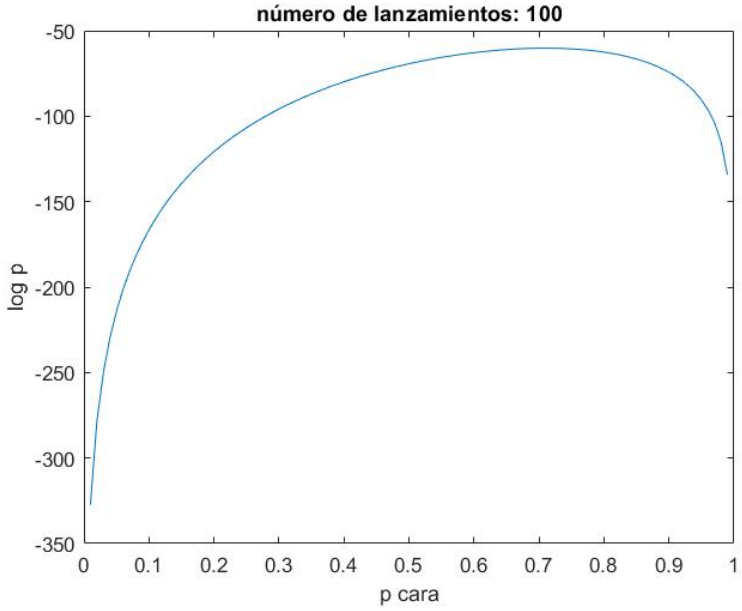


Figura 2.6. Gráfica de probabilidad para 100 lanzamientos

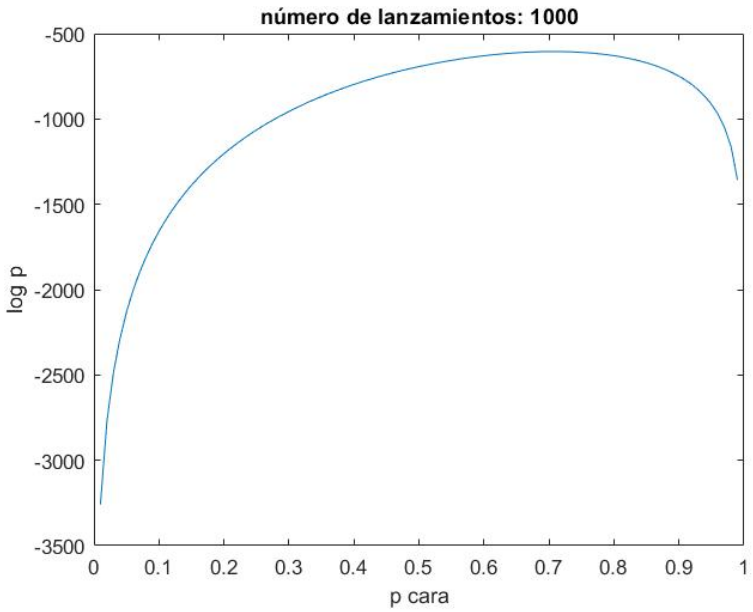


Figura 2.7. Gráfica de probabilidad para 1000 lanzamientos

Como se puede ver en las imágenes anteriores, la solución del problema es altamente dependiente del número de muestras que se poseen. Esta conclusión es importante tenerla en cuenta para obtener soluciones más precisas del problema cuando sean necesarias. Otra conclusión que puede obtenerse del incremento en el número de muestras es la utilidad de tomar el logaritmo a la función de verosimilitud. Las implementaciones mostradas en este trabajo se realizaron mediante el programa Matlab, y el cálculo de las soluciones se realizó numéricamente. Por este motivo, es importante evitar o reducir en la medida de lo posible la acumulación de errores. Dado que el número de muestras suele ser grande, y por consiguiente la probabilidad de la observación muy pequeña, el uso del logaritmo convierte un número con muchas cifras decimales en un número de menos cifras significativas, y adicionalmente convierte la operación de multiplicación necesaria para hallar la probabilidad de la observación completa en una operación de suma, por lo que la propagación de errores se reduce considerablemente.

3 IMPLEMENTACIÓN DEL MODELO DE RED

Para la implementación de los algoritmos de máxima verosimilitud que ayudan a modelar comportamientos sociales, es necesario primeramente un modelo de red. En este trabajo se utilizó el modelo propuesto en el artículo [1], que, como se explicó en el capítulo introductorio, analiza la influencia que ejercen los usuarios de vehículos eléctricos en sus vecinos y cómo, en función de la susceptibilidad de los últimos (porcentaje mínimo de vecinos usuarios de vehículo eléctrico que debe tener un individuo para adquirir uno él mismo), provocan que sus vecinos se conviertan en nuevos usuarios que influirán a su vez a otras personas. El modelo descrito en el artículo [1] presenta las siguientes ventajas:

- Los agentes están definidos por su posición geográfica. Por este motivo, es un buen modelo tanto para trabajar con redes generadas aleatoriamente, como para utilizar datos reales de ciudades o comunidades determinadas.

- El modelo incluye diversas operaciones de tratamiento de la red que permiten descartar determinados agentes con el objetivo de obtener un grafo conexo. Si el grafo no fuera conexo, existirían pequeñas "comunidades" que quedarían aisladas de la influencia del resto de agentes; por el contrario, si el grafo es conexo, cualquier agente puede llegar a influir a otro aunque sea de forma indirecta, ya que siempre existirá un camino entre un agente y otro, ya sea directamente, o a través de uno o varios vecinos.

- La influencia que tienen los vecinos sobre un agente determinado se modela de forma sencilla mediante un parámetro (α). El cálculo de la dinámica se explicará en el apartado siguiente, pero es una dinámica sencilla y a su vez realista.

A continuación, se exponen los aspectos más importantes del modelo descrito en [1]:

3.1. Modelo del artículo original

En el artículo [1], se modela una red a partir de datos obtenidos de la circulación de 1000 vehículos en una provincia de Italia durante 1 año. Con los datos de la circulación, se modela la red basando la interacción de los usuarios en su proximidad a otros usuarios. Al tratarse de un análisis de la adopción de un determinado tipo de vehículo, debe establecerse una posición determinada como el lugar asignado a un usuario. Este lugar de referencia, llamado casa, se obtiene mediante el análisis de los datos de circulación; la posición en la que cada vehículo se mantiene parado durante más tiempo en las horas nocturnas se considera la casa del usuario. Una vez obtenida la posición de la casa de cada usuario, se comienza a analizar la proximidad entre usuarios [1].

Con el objetivo de obtener un grafo conexo, algunos se eliminan [1], en concreto aquellos que cumplieran alguna de las siguientes condiciones:

- Agentes remotos *(remote agents)*: aquellos cuya casa está separada más de 60km de la de algún otro usuario, y aquellos que no cuentan con ningún agente a 5km o menos de distancia de él [1].
- Valores atípicos *(outliers)*: aquellos usuarios cuya casa se encontraba demasiado lejos de una posición (en forma de latitud y longitud) que corresponde a la media aritmética de las posiciones de todos los agentes de la red (también en forma de latitud y longitud). Está condición se calcula matemáticamente de la siguiente forma [1]:

1. Se define la posición promedio como $\mu_h = (long_\mu, lat_\mu)$. Los valores de latitud y longitud se calculan en el artículo realizando la media aritmética de las longitudes y latitudes de cada una de las casas consideradas [1].

2. Se calcula la desviación típica de cada agente a la posición promedio [1].

3. Aquellos agentes cuya distancia a la posición promedio fuera mayor a la desviación típica calculada, es considerado un usuario atípico y se elimina de la red [1].

- Por último, se eliminan de la red los agentes aislados, es decir, aquellos cuyo número de vecinos es menor a un determinado número, en concreto, el artículo [1] usa como umbral 8 vecinos.

Al realizar el anterior proceso de criba, se obtiene un grafo conexo en el que permanecían en torno al 90% de usuarios de la red original antes de la criba [1].

Debido a que el análisis realizado en el artículo de referencia consiste en observar la influencia que tienen unos usuarios sobre otros, es decir, cuántos usuarios optan por adoptar un vehículo eléctrico al verse influidos por otros adoptantes previos, los autores del artículo seleccionaron vehículos "semilla" *(seeds)*, es decir, aquellos que adoptan inicialmente el vehículo eléctrico y que servirán de influencia al resto de usuarios que no posee dicho tipo de vehículo [1].

En función de la distancia recorrida por el usuario diariamente, el artículo de referencia clasifica a los agentes en aptos para la adopción del vehículo eléctrico y en actualmente no aptos para la adopción del vehículo eléctrico [1], [5].

Al realizar la clasificación antedicha, en el artículo [1] se muestra como aproximadamente un 30% de los agentes de la red caen dentro de la categoría de aptos para la adopción del vehículo eléctrico, sin embargo, en el mundo real y actualmente, el porcentaje de vehículos que son eléctricos respecto a todas las matriculaciones de vehículos es de alrededor del 7% [1], por consiguiente, el artículo no toma como conjunto de adoptantes iniciales la totalidad de los agentes que son aptos para su adopción, sino el 25% de los mismos seleccionados de forma aleatoria, de esta manera, se consigue que el porcentaje de adoptantes iniciales sea muy similar al 7% anteriormente mencionado que corresponde a un valor más fiel a la realidad [1].

Una vez identificados los adoptantes iniciales dentro de la red, el artículo [1] implementa un algoritmo que modela la difusión de la opinión a través de la red, es decir, los agentes que inicialmente no son adoptantes del vehículo eléctrico, pero que, debido a la influencia de sus agentes vecinos (recordando esta definición, son aquellos que distan 5km o menos entre sí) lo adoptan pasado un tiempo. La dinámica que modela la adopción de vehículos eléctricos en un determinado tiempo t discreto es la siguiente [1]:

La ecuación mostrada a continuación modela la dinámica de influencia entre agentes utilizada en el artículo [1]. Dicha ecuación indica que el nuevo conjunto de adoptantes de vehículo eléctrico S_t corresponderá a aquellos agentes que no se encontraban en los conjuntos de adoptantes de vehículo eléctrico de algún instante de tiempo t anterior (es decir, un agente no puede ser adoptante dos veces), tales que la proporción de adoptantes de vehículo eléctrico en su vecindad entre el número total de vecinos sea mayor o igual a un determinado umbral α que adoptará un valor comprendido entre 0 y 1. En otras palabras, el umbral α indica como de "influenciable" es un determinado agente; si $\alpha = 0$ significará que el agente adoptará el vehículo eléctrico pase lo que pase, y si $\alpha = 1$ el agente no adoptará el vehículo eléctrico a menos que la totalidad de sus vecinos lo haya adoptado antes [1]. Recordando capítulos y apartados anteriores, α es el porcentaje mínimo de vecinos adoptantes que debe tener un determinado agente para realizar el mismo la adopción de un cambio.

$$S_t = \{v \in V \backslash (\cup_{t=0}^{t-1} S_\tau): \frac{|(\cup_{t=0}^{t-1} S_\tau) \cap N_v|}{|N_v|}\} \geq \alpha \quad (3.1)$$

v: agente de la red

V: conjunto de todos los agentes de la red

S_τ: adoptantes en un determinado instante de tiempo

N_v: vecinos del agente v

α: parámetro que modela la susceptibilidad de un agente a ser influido por sus vecinos

3.2. Implementación de la red en Matlab

El modelo inicial utilizado en este trabajo se implementó siguiendo unas pautas similares a las expuestas anteriormente, pero con algunas diferencias significativas. Adicionalmente, para poder modelar problemas más diversos, en capítulos posteriores se explican diferentes modificaciones que se hicieron al modelo.

En primer lugar, se programó un script de Matlab que dado un número n, generara aleatoriamente n agentes dentro de unos valores de latitud y longitud determinados (dichos valores pueden elegirse) y calculara sus vecinos.

El primer paso es, como se ha indicado en el párrafo anterior, generar n agentes con valores de latitud y longitud aleatorios y comprendidos entre unos límites determinados. Una vez generados, se guarda el valor de la latitud y longitud de cada agente en una matriz y se dibuja una gráfica en la que se representan los agentes y su correspondiente valor de latitud y longitud. En la imagen siguiente se muestra dicha gráfica para el caso de 500 agentes:

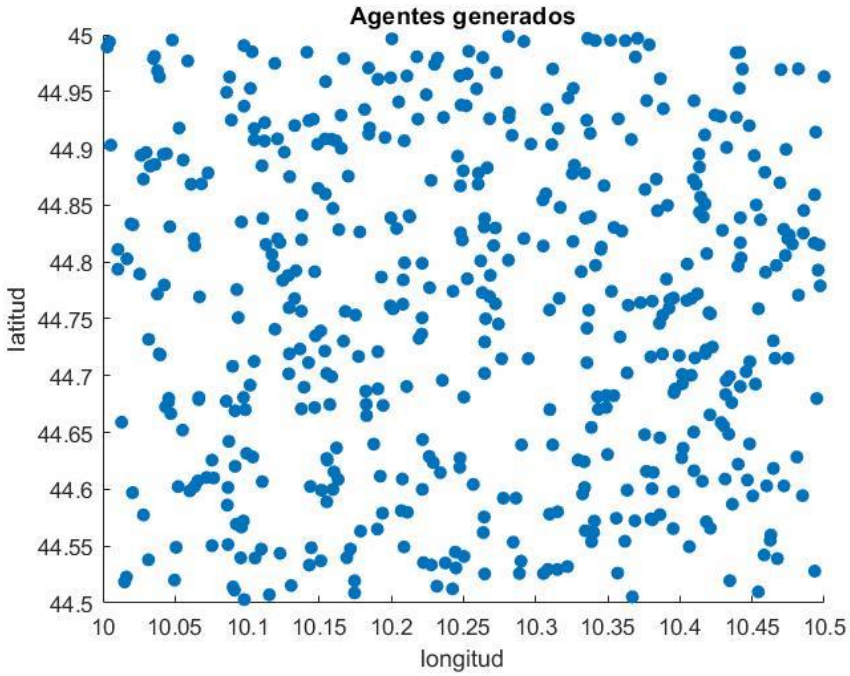


Figura 3.1. Agentes generados (n=500)

El siguiente paso fue la implementación del proceso de criba descrito en el apartado anterior que permite la obtención de un grafo conexo tras la identificación de los vecinos de los diferentes agentes. Para ello, se programó el cálculo de una matriz que guarda en cada elemento la distancia que existe entre dos agentes de la red, de este modo, la matriz tendrá dimensión n.

Gracias a esta matriz, pueden eliminarse de la matriz de agentes aquellos que disten más de 60km de algún otro agente cualquiera, y aquellos que no tengan ningún agente a 5km o menos de distancia. Para lograr este objetivo, se crearon 2 matrices lógicas llamadas prune_1 y prune_2, ambas de dimensión n. prune_1(i,j) valdrá 1 si el agente i dista menos de 60km del agente j, y prune_2(i,j) valdrá 1 si el agente i dista menos de 5km del agente j. De este modo, si existe un elemento (i,j) de prune_1 igual 1, el agente i será eliminado, y si la suma de una determinada fila i de prune_2 vale 0, el agente i será eliminado.

También se implementó la criba por número de vecinos; recorriendo la matriz de agentes, para un agente determinado se incrementa un contador para aquellos agentes que distan 5km o menos de él. Si el contador al finalizar el recorrido de la matriz es menor a un determinado valor (ajustable en el programa), el agente es eliminado de la red.

Adicionalmente, se implementó la eliminación de aquellos agentes que distaran de la posición promedio un valor superior a la desviación típica de la distancia de cada agente a dicha posición. No obstante, debido a que en la implementación realizada en este trabajo la red se genera aleatoriamente, este paso de la criba elimina demasiados agentes de la red (en torno a un 60-70%), sin embargo, realizando pruebas de simulación se observó que eliminando un número mucho menor de agentes la red era conexa, por lo que se decidió prescindir de este paso del proceso de criba. De este modo, se obtienen redes más numerosas (por consiguiente, habrá un mayor número de muestras, lo cual beneficiará a la hora de aplicar algoritmos estadísticos) sin tener que comprometer la conectividad de las mismas.

En la imagen siguiente se muestra la red obtenida tras el proceso de eliminación de agentes anteriormente explicado:

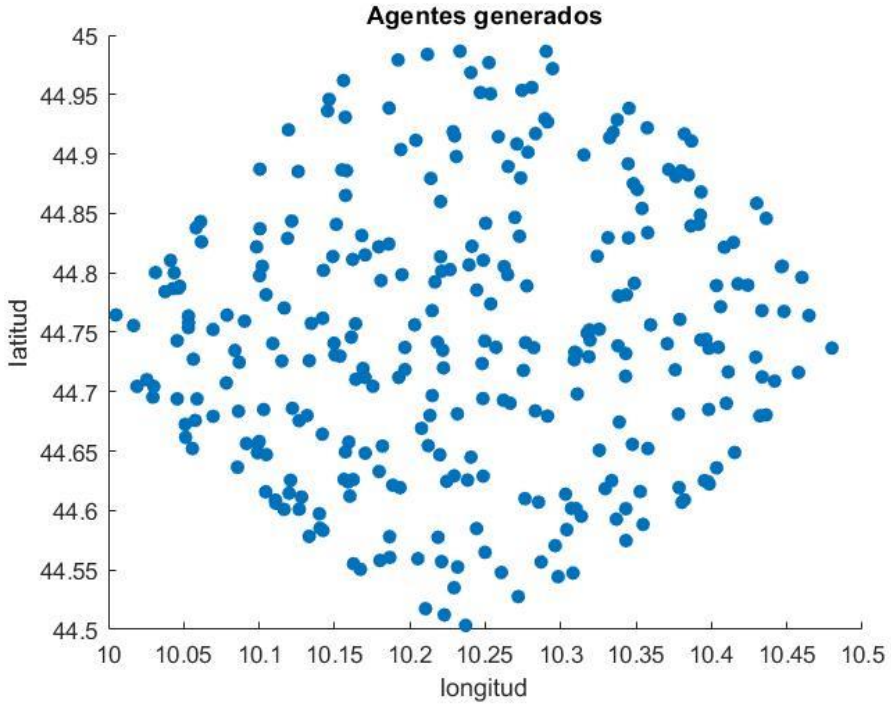


Figura 3.2. Agentes restantes tras el cribado (n=500)

Una vez realizada la criba, se procede a la identificación de los vecinos de cada agente. Para ello se emplea una matriz, llamada neighbors, que posee 2 columnas. En la primera columna se guarda el número del agente cuyos vecinos están siendo identificados, y en la otra los agentes que son sus vecinos (distan 5km o menos). A continuación, se muestra un ejemplo de la matriz neighbors:

```
neighbors =

     1    18
     1    27
     1    39
     1    42
     2    20
     2    35
     2    36
     2    56
     3    44
     3    48
     3    52
     3    68
     3    69
     4    20
     4    35
     5    38
```

Figura 3.3. Matriz de vecinos

En el ejemplo anterior, los vecinos del agente 1 son 18, 27, 39 y 42; los vecinos del agente 2 son 20, 35, 36 y 56, etc.

Esta matriz es de suma importancia para la dinámica que modela la influencia, ya que permite conocer quiénes son los vecinos de un determinado agente para saber si son adoptantes o no, y permite calcular el número total de vecinos que posee, lo cual es crucial para el cálculo de la proporción de vecinos adoptantes que se comparará con el valor de α para así determinar si tiene lugar la adopción o no.

Adicionalmente, para la comprobación de la conectividad de la red, el programa desarrollado permite representar gráficamente los agentes unidos mediante una línea a sus vecinos; de este modo es sencillo identificar posibles comunidades de agentes aislados. La figura 3.4 muestra dicha representación gráfica y permite comprobar que la red es conexa.

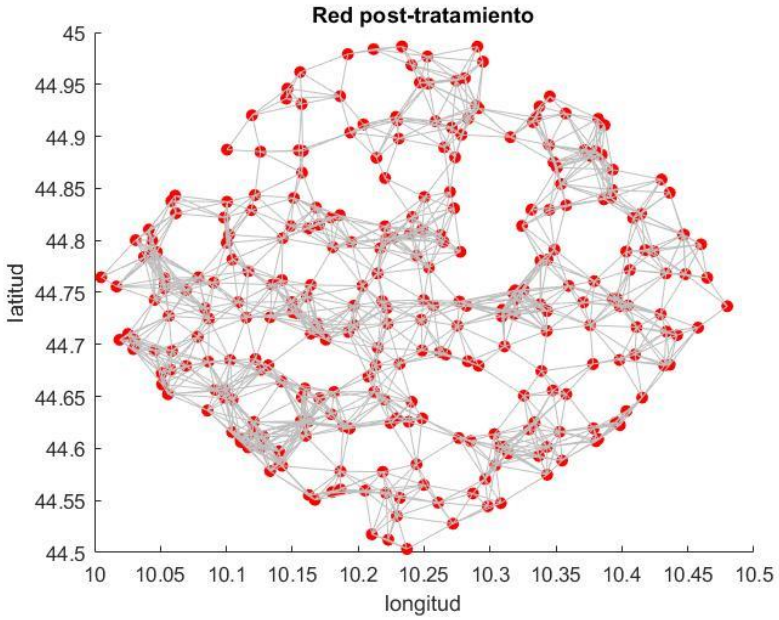


Figura 3.4. Red de agentes unidos a sus vecinos

3.3. Implementación de la dinámica de adopción en Matlab

Una vez se programó el script que genera la red de agentes, elimina aquellos que dificultan la conectividad de la red y calcula y guarda los vecinos de cada agente, el siguiente paso fue la programación de la dinámica de adopción.

Como se explicó en apartados anteriores, el proceso de adopción consiste en lo siguiente: para cada agente, se calcula el número de vecinos adoptantes que posee, se divide ese valor entre el número total de sus vecinos obteniendo así un valor que puede estar entre 0 (ninguno de sus vecinos es adoptante) y 1 (la totalidad de sus vecinos lo es), y se compara dicho valor con el parámetro α que posee el agente. Si el porcentaje (en tanto por uno) de vecinos adoptantes del agente es mayor o igual a su valor de α, el agente se convertirá en adoptante; si no lo es, seguirá sin serlo. En otras palabras, si un determinado agente de la red presenta una fracción de vecinos adoptantes suficientes (mayor o igual a α), dicho agente procederá a adoptar el mismo el cambio.

El funcionamiento del script que ejecuta la dinámica de adopción se describe a continuación:

En primer lugar, se define un número de iteraciones que permitirá simulaciones más o menos largas según se desee. Posteriormente se define una matriz llamada *historia*, de dimensiones $n \times (n_{iteraciones} + 2)$. Dicha matriz guardará el estado de cada agente en cada instante de la iteración, otorgándole un valor de 1 si es adoptante, y un valor de 0 si no lo es. Posee n filas ya que cada fila guardará la información sobre un agente concreto (en total hay n agentes), y $n_{iteraciones} + 2$ columnas ya que la primera columna guardará el número de cada agente para permitir su identificación, la segunda guardará si un agente ha sido adoptante inicial (antes de ejecutar la dinámica) o no, y el resto de las columnas, su estado para cada iteración.

Una vez definidos los elementos iniciales citados en el párrafo anterior, el siguiente paso es asignar un valor de α a cada agente. En el caso de la primera implementación, se asignó el mismo valor a todos los agentes (aunque se permite su modificación), ya que así se realizó en el artículo de referencia [1]. En capítulos posteriores, esto se modificará para permitir que el valor de α siga una distribución estadística deseada, con la posibilidad de modificar sus parámetros. El valor de α, una vez generado, se guarda en forma de vector de dimensión n, y dicho vector se coloca en forma de columna en la matriz que guarda la latitud, longitud y valor de distancia recorrida de cada agente.

Posteriormente, se procede a la computación del conjunto de adoptantes iniciales. Como se explicó

anteriormente, la matriz que guarda la información geográfica sobre los agentes también contiene el valor de la distancia recorrida de cada agente (para el caso de los vehículos eléctricos; parámetro que puede cambiarse para modelar otros problemas). Los agentes que pertenecen al conjunto de adoptantes iniciales serán aquellos cuya distancia recorrida diaria sea menor a un determinado valor. Este valor es ajustable y, permite modificar el porcentaje de adoptantes iniciales para que coincida con un valor deseado, esto es, cuanto mayor sea dicho valor, mayor será el porcentaje de adoptantes iniciales.

Una vez se tienen los adoptantes iniciales, debido a que anteriormente se calculó una matriz que permite conocer los vecinos de cada uno de los agentes, se procede a la simulación de la dinámica de adopción. Esta dinámica se implementó de la siguiente manera: para cada agente, se calcula la proporción entre vecinos adoptantes (se sabe si un vecino es adoptante o no gracias a la matriz *historia* anteriormente explicada) y vecinos totales. Si el valor obtenido es mayor al valor de α del agente en cuestión, se guarda el número que identifica a dicho agente (el cual se encuentra en la primera columna de la matriz *historia*) en un vector que guarda el número identificativo de cada agente adoptante. Posteriormente, el valor de la matriz historia en la columna correspondiente a la siguiente iteración, para cada agente del vector recién mencionado se pone a 1, de esta manera, contarán como vecinos adoptantes cuando se desee calcular los nuevos adoptantes en la siguiente iteración.

El proceso descrito en el párrafo anterior se repite $n_{iteraciones}$ veces. De este modo, se obtiene la dinámica descrita en el artículo de referencia. A continuación, se muestran varias imágenes que permiten ver el proceso de adopción. En todas ellas se utiliza un número de iteraciones igual a 4, y un número de agentes adoptantes iniciales de 500.

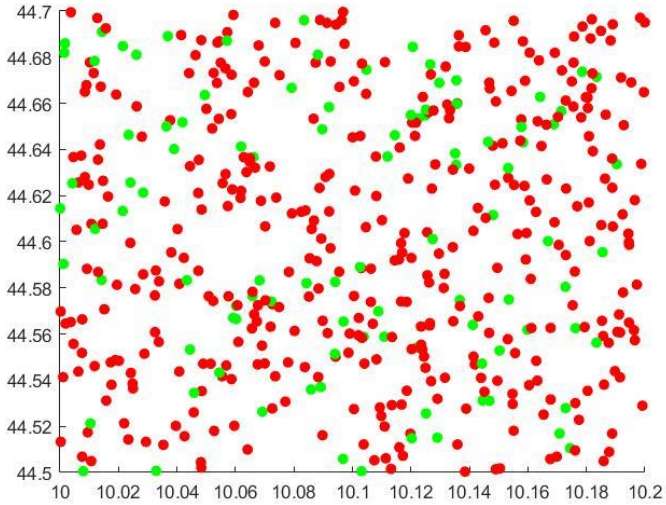


Figura 3.5. Adoptantes iniciales con $\alpha = 0.3$

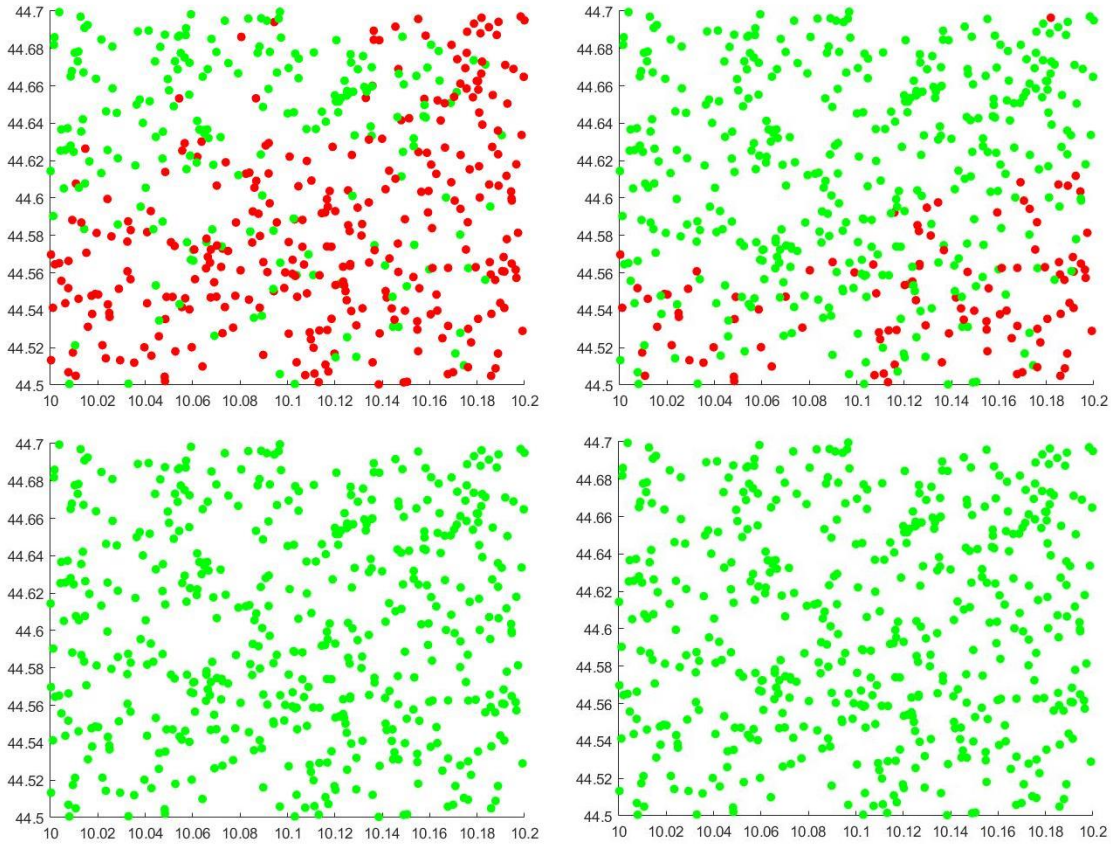


Figura 3.6. Dinámica de adopción con $\alpha = 0.3$.

En la figura 3.6 se muestra la dinámica de adopción para un número de iteraciones igual a 4. La imagen superior izquierda corresponde a la primera iteración, la imagen superior derecha a la segunda, la inferior izquierda a la tercera, y la inferior derecha a la cuarta y última. Los puntos verdes representan agentes adoptantes y los puntos rojos no adoptantes.

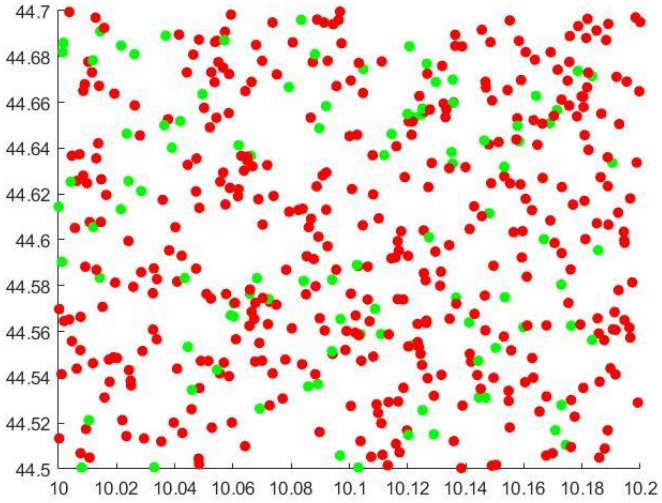


Figura 3.7. Adoptantes iniciales con $\alpha = 0.35$

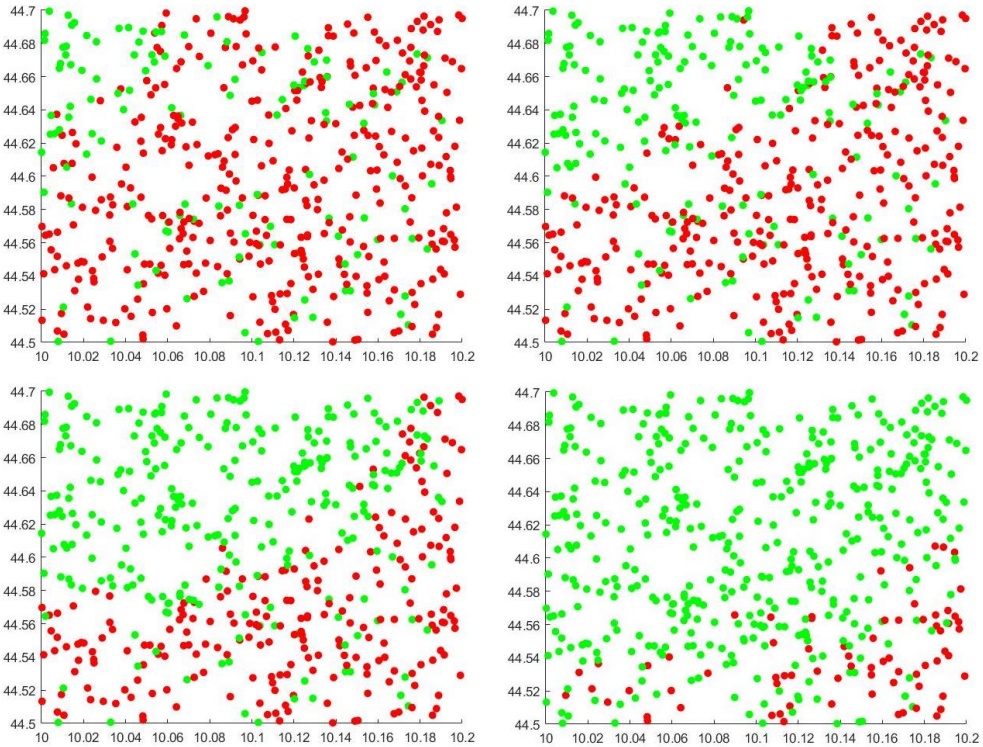


Figura 3.8. Dinámica de adopción con $\alpha = 0.35$

Nota: En la figura 3.8 se muestra la dinámica de adopción para un número de iteraciones igual a 4. La imagen superior izquierda corresponde a la primera iteración, la imagen superior derecha a la segunda, la inferior izquierda a la tercera, y la inferior derecha a la cuarta y última. Los puntos verdes representan agentes adoptantes y los puntos rojos no adoptantes.

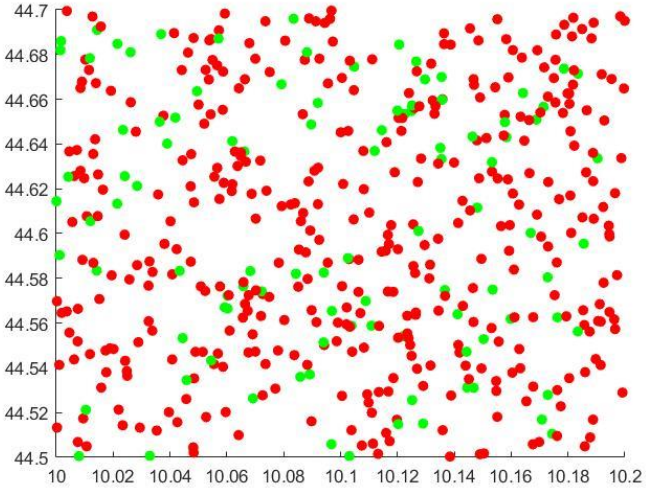


Figura 3.9. Adoptantes iniciales con $\alpha = 0.5$

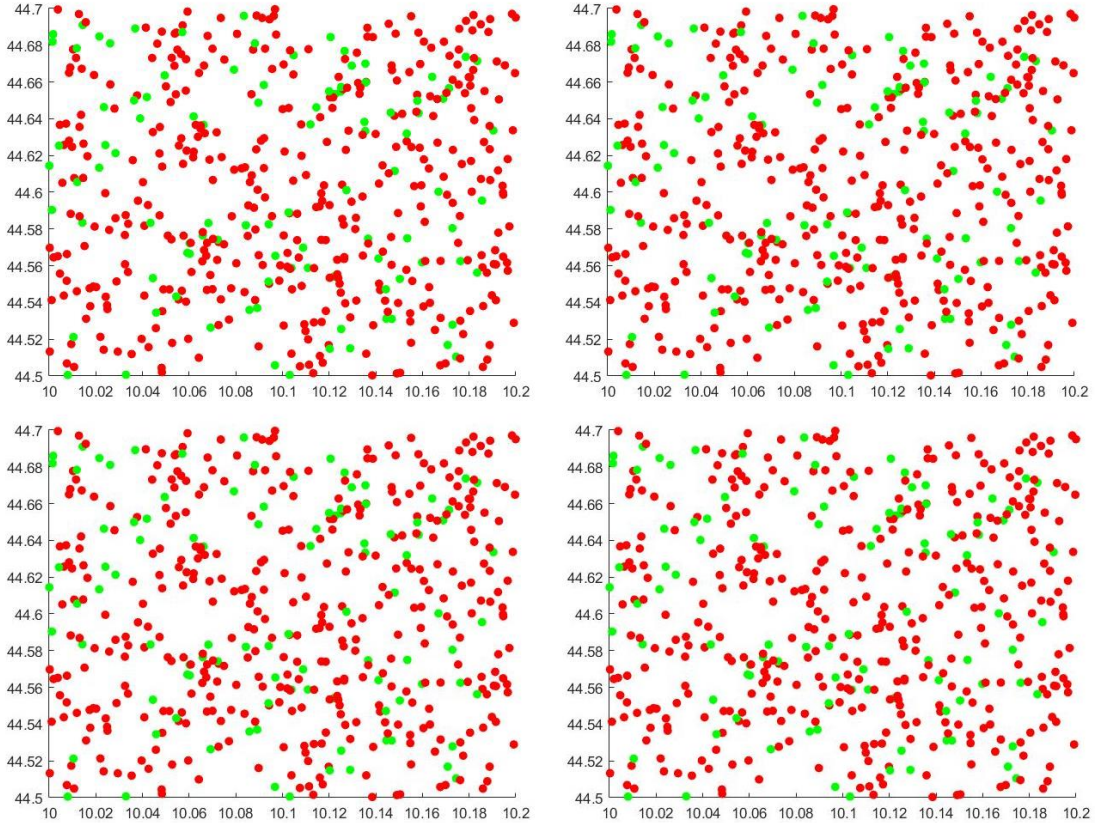


Figura 3.10. Dinámica de adopción con $\alpha = 0.5$

Nota: En la figura 3.10. se muestra la dinámica de adopción para un número de iteraciones igual a 4. La imagen superior izquierda corresponde a la primera iteración, la imagen superior derecha a la segunda, la inferior izquierda a la tercera, y la inferior derecha a la cuarta y última. Los puntos verdes representan agentes adoptantes y los puntos rojos no adoptantes.

De las imágenes anteriores se observa que, si se asigna a todos los agentes el mismo valor del parámetro α, la dinámica presenta poca variedad. Para el valor de $\alpha = 0.3$, la totalidad de los agentes se convierte en adoptantes antes incluso de terminar las 4 iteraciones. Por otro lado, para $\alpha = 0.5$, el número de agentes adoptantes apenas sufre variaciones durante el transcurso de la dinámica. Por este motivo, además de para otorgar más realismo al modelo (en una población real las personas tienen un grado de susceptibilidad diferente; existen personas muy sensibles a los actos y/o opiniones de sus vecinos y personas que lo son muy poco), se procedió a implementar una modificación al mismo que permitiera que el parámetro α siguiera una distribución estadística elegida. Este cambio, unido al empleo de un algoritmo de máxima verosimilitud para la obtención de los parámetros característicos de la distribución estadística que gobierna α para resolver un problema inverso, se describe en el siguiente capítulo.

4 ALGORITMO DE MÁXIMA VEROSIMILITUD – ESTIMACIÓN DE PARÁMETROS CARACTERÍSTICOS DE LA DISTRIBUCIÓN PROBABILÍSTICA

En este capítulo se describe la implementación de un algoritmo de máxima verosimilitud para hallar el parámetro característico de una distribución estadística que gobierna los valores del parámetro α de cada uno de los agentes de la red.

4.1. Objetivos

Para lograr el objetivo recién mencionado de hallar el parámetro característico de la distribución exponencial que sigue el parámetro α de los agentes de la red, primero debe modificarse el programa que se implementó en el capítulo anterior para generar la red de agentes y simular la dinámica de adopción.

En el capítulo anterior, el parámetro α era igual para todos los agentes; en este capítulo, se explica la modificación programa de forma que el parámetro α siga una distribución estadística elegida, así como permitir la elección del parámetro característico para, de este modo, obtener diferentes dinámicas; algunas más lentas y con menos adoptantes al final de la simulación, y otras más rápidas y con más adoptantes finales.

Una vez se modificó el programa, se explicará cómo se implementó el algoritmo de máxima verosimilitud para hallar los parámetros que caracterizan el comportamiento probabilístico de la red (por ejemplo, el parámetro λ en el caso de una distribución exponencial) a partir de una simulación ya realizada. Al final del capítulo, se mostrarán varias simulaciones que muestran el correcto funcionamiento del algoritmo.

4.2. Modificación del programa que simula la dinámica

Como se explicó en el capítulo anterior, la generación del parámetro α para cada agente se hacía creando un vector de dimensión n (igual al número de agentes) cuyas componentes son iguales entre sí y al valor del α elegido.

El siguiente paso fue la creación de una función, *generate_alpha_vector*, que recibe como parámetros de entrada la dimensión del vector deseado (es decir, el número de agentes), y el parámetro o parámetros característicos de la función. El código del que se compone la función programada se muestra a continuación:

```
function alpha = generate_alpha_vector(n)

dist=input('distribución deseada (exponencial, weibull, normal): ','s');

switch dist
    case 'exponencial'
        lambda=input('introducir parámetro lambda: ');
        mu=1/lambda;
        alpha=exprnd(mu,n,1);
    case 'weibull'
        lambda=input('introducir parámetro lambda: ');
        k=input('introducir parámetro k: ');
        alpha=wblrnd(lambda,k,n,1);
    case 'normal'
        mu=input('introducir parámetro mu: ');
        sigma=input('introducir parámetro sigma: ');
        alpha=normrnd(mu,sigma,n,1);
end

end
```

Figura 4.1. Función generate_alpha_vector

Como se aprecia en la imagen anterior, el funcionamiento es muy sencillo. La función está preparada para usar diversas distribuciones (Weibull, normal y exponencial). El argumento que recibe la función es n, el número de agentes para los que se desea generar un valor de α. Una vez se llama a la función, ésta pregunta la distribución deseada, tras lo cual el usuario deberá introducir por teclado exponencial, Weibull o normal según corresponda. Tras haber hecho esto, la función preguntará el valor del parámetro o parámetros de la distribución probabilística, que deberán ser introducidos por teclado. Una vez hecho esto, la función devuelve un vector de dimensión n cuyas componentes corresponden a valores aleatorios de α generados a partir de la distribución elegida por el usuario.

En lo que a la generación del vector se refiere, el programa utiliza los generadores aleatorios de Matlab de las diferentes funciones de distribución. Afortunadamente, Matlab posee una gran variedad de funciones de distribución listas para su uso, por lo que no hizo falta programar la generación del vector por separado.

A continuación, se mostrará un ejemplo de generación del vector de α para el caso de la distribución exponencial, mostrando que es congruente con las gráficas que se obtienen para diferentes valores del parámetro λ.

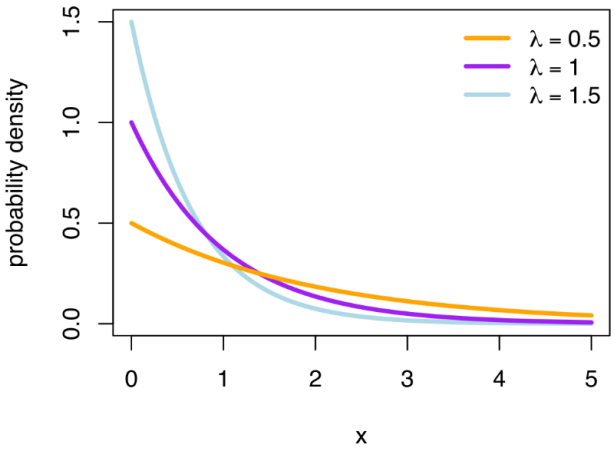


Figura 4.2. Función de densidad de probabilidad de la distribución exponencial para diferentes λ

En la imagen anterior se puede observar como, incrementando el valor del parámetro λ, los valores de la variable aleatoria tienden a situarse más cercanos al cero. El valor esperado o la media de la variable aleatoria en una distribución exponencial es $E(x) = \frac{1}{\lambda}$, de este modo, si se genera un vector con $\lambda = 5$, el valor medio de sus componentes debería ser cercano a 0.2. A continuación se muestra una imagen donde se comprueba:

```
>> mean(generate_alpha_vector(1000,5))

ans =

    0.2008
```

Figura 4.3. Comprobación de un vector generado por la función programada

De acuerdo con la imagen, que muestra la media de 1000 valores aleatorios generados mediante una distribución exponencial de parámetro $\lambda = 5$, el vector generado cumple con el comportamiento que indica la teoría sobre la distribución exponencial.

4.3. Implementación del algoritmo de máxima verosimilitud.

Es fácil intuir que si los parámetros de la distribución probabilística hacen que la mayoría de los agentes se concentren en torno a valores relativamente elevados del parámetro α, el proceso de adopción será más lento (al haber más agentes que requieran un elevado número de vecinos adoptantes) y en el régimen permanente probablemente quedarán bastantes agentes sin haber sido adoptantes. Por el contrario, si la distribución probabilística hace que la mayor parte de los agentes se concentren en torno a valores bajos de α, se esperará observar una dinámica de adopción más rápida y en el régimen permanente, un menor número de agentes que queden sin adoptar. Basándose en estas probabilidades de observación, el algoritmo de máxima verosimilitud busca los parámetros que mejor se ajusten a ellas.

En cada instante t de la simulación, un agente sólo puede ser adoptante o no adoptante. Teniendo esto en cuenta, en el transcurso de toda la simulación, pueden darse varios casos diferentes para cada agente:

1. El agente forma parte del conjunto de los adoptantes iniciales, por lo que no participa de la dinámica al no haber cambios en su estado (empezó siendo adoptante y permanecerá siéndolo durante toda la simulación)

2. El agente no formaba parte de los adoptantes iniciales y durante la simulación se convirtió en un adoptante.

3. El agente no formaba parte de los adoptantes iniciales y nunca llegó a convertirse en un adoptante durante la simulación.

Tendiendo esto en cuenta, la implementación del algoritmo de máxima verosimilitud consistirá en revisar la información sobre qué ha ocurrido con cada uno de los agentes en cada instante de la simulación, para así saber cuándo han pasado a ser adoptantes (si así lo han hecho) o si han permanecido sin serlo durante el transcurso de toda la simulación; posteriormente, se utilizará el siguiente procedimiento para calcular la probabilidad de que haya ocurrido lo que se ha observado para cada agente concreto:

1. Como se ha explicado anteriormente, el parámetro α de cada agente indica el porcentaje de

vecinos adoptantes necesarios para que dicho agente se convierta en adoptante; de este modo, observando el instante en el que un agente pasa a convertirse en adoptante, se puede obtener una cota superior del parámetro α; por ejemplo, si un determinado agente pasa a convertirse en adoptante, y en ese instante se observa que el número de vecinos adoptantes es igual a 5, mientras que el número de vecinos totales es igual a 10, se puede inferir que α será menor o igual a 0.5, ya que en caso contrario, el agente no se habría convertido en adoptante al no tener suficientes vecinos que lo sean. Por otro lado, observando el instante inmediatamente anterior a aquel en el que se produjo la adopción, se puede calcular la cota inferior de α mediante el mismo razonamiento; si con un determinado porcentaje de vecinos adoptantes, el agente estudiado no se convirtió en uno de ellos, el parámetro α debe obligatoriamente ser superior a dicho porcentaje.

2. Si un agente no se convierte en adoptante en ningún instante de la simulación, se puede utilizar el porcentaje de vecinos adoptantes en el instante final como cota inferior de α empleando el mismo razonamiento explicado en el punto anterior. En este caso, como no se ha producido la adopción, la cota superior corresponderá a la de la función de densidad de probabilidad; para el caso de la exponencial o la normal, dicha cota sería infinito.

3. Una vez se han obtenido las cotas inferior y superior del parámetro α para cada agente, integrando la función de densidad de probabilidad con límites de integración inferior y superior iguales a la cota inferior y superior de α respectivamente, se puede obtener la probabilidad de la observación para cada agente. A partir de estas probabilidades, se puede obtener la probabilidad del resultado de la simulación completa de forma muy sencilla.

Como se ha explicado recientemente, es necesario llevar un registro de qué ha ocurrido con cada agente en cada instante de la simulación. En el programa desarrollado en Matlab durante este proyecto, este registro se ha realizado mediante una matriz. Dicha matriz es la llamada *historia,* mencionada ya en capítulos anteriores, y posee un número de filas igual al número de agentes considerados en la simulación, y un número de columnas igual al número de iteraciones de la simulación más 2 columnas adicionales: una para guardar el número de agente, y otra para conocer si un determinado agente era adoptante inicial (antes de simular la dinámica) o no.

```
historia =

     1     0     1     1     1     1     1     1
     2     1     1     1     1     1     1     1
     3     0     1     1     1     1     1     1
     4     0     1     1     1     1     1     1
     5     0     0     1     1     1     1     1
     6     0     1     1     1     1     1     1
     7     0     0     1     1     1     1     1
     8     0     0     1     1     1     1     1
     9     0     0     0     0     0     0     0
    10     0     0     0     0     0     0     0
    11     0     1     1     1     1     1     1
    12     0     1     1     1     1     1     1
    13     0     1     1     1     1     1     1
    14     0     0     1     1     1     1     1
    15     1     1     1     1     1     1     1
    16     0     0     1     1     1     1     1
    17     0     1     1     1     1     1     1
    18     0     0     1     1     1     1     1
```

Figura 4.4. Matriz historia

En la imagen superior se observa un ejemplo de la matriz historia que sigue el orden explicado en el párrafo anterior. Como ejemplos para clarificar su funcionamiento, el agente 18 no es un adoptante inicial (segunda columna de la matriz en la fila 18 vale 0), y pasó a ser adoptante en el instante 2 de la simulación (columna 4 de la fila 18 vale 1). Por el contrario, los agentes 9 y 10 nunca llegaron a ser adoptantes, ya que el valor que indica su estado es 0 durante toda la simulación. Un ejemplo de adoptante inicial es el agente 2, cuyo estado vale 1 desde antes de la simulación y permanece así durante toda ella.

Por otro lado, para computar la probabilidad de obtener la secuencia observada para cada agente, es necesario calcular la cota superior e inferior del parámetro α, para poder así integrar la función de densidad de probabilidad entre dichos límites y obtener la probabilidad buscada.

Para obtener dichas cotas, se programó una función en Matlab llamada *calculate_alpha_limit*, que recibe como entradas el número que identifica al agente que se quiere analizar, el instante que se desea analizar, la matriz que guarda la información sobre los vecinos de cada agente, y la matriz historia.

El funcionamiento de dicha función es el siguiente: se recorre la matriz que guarda la información sobre los vecinos hasta llegar al agente que se quiere analizar; posteriormente observa qué estado corresponde a cada vecino de dicho agente; si se trata de un adoptante incrementa un contador de vecinos adoptantes; en caso contrario no lo hace, pero en cualquier caso incrementa el número de vecinos totales. Una vez se han terminado de analizar todos los vecinos del agente que está siendo estudiado, se obtiene el porcentaje (en tanto por uno) de vecinos adoptantes de dicho agente dividiendo el contador de vecinos adoptantes entre el contador de vecinos totales.

Dependiendo del instante que se haya introducido como entrada a la función, se puede obtener una cota superior o inferior del parámetro α, esto es, si el instante que se pasa a la función es aquel en el que se produjo la adopción, el porcentaje que devuelve corresponde a la cota superior de α. Por el contrario, si el instante que se pasa es el anterior a la adopción, o el último de la simulación en caso de que el agente no haya llegado a ser adoptante en ningún momento, el porcentaje obtenido corresponderá a la cota inferior de α.

Una vez obtenidas las cotas superior e inferior, se integrará la función de densidad de probabilidad de la distribución exponencial usando como límites de integración dichas cotas. Esto se repetirá para todos los agentes, guardando la probabilidad de la observación de cada uno en un vector auxiliar. Para obtener la probabilidad de la observación total, basta con multiplicar todas las componentes de dicho vector auxiliar entre sí, o, como se explicó en el capítulo introductorio sobre el algoritmo de máxima verosimilitud, tomar el logaritmo para así convertir el productorio en sumatorio. Debido a que esta segunda opción es menos propensa a acumular errores, se optó por ella en este trabajo.

El procedimiento explicado en el párrafo anterior se realiza para varios valores del parámetro característico de la función de densidad. Para cada valor, se guarda la probabilidad de la observación total en un vector, para luego proceder a su representación gráfica y el cálculo del valor que maximiza la verosimilitud.

En la siguiente tabla se muestra un sencillo ejemplo de identificación de la cota inferior y superior del parámetro α de un agente determinado:

Número total de vecinos del agente: 10						
Estado del agente:	0	0	0	0	1	1
Vecinos adoptantes del agente	0	1	3	4	5	7

Tabla 4.1. Ejemplo de cálculo de las cotas superior e inferior del parámetro α

En la tabla 4.1, se observa que el agente pasó a ser adoptante a partir del instante 4 de la adopción, por lo que el límite superior del parámetro α es $\alpha \leq \frac{4}{10} = 0.4$. Asimismo, se observa que del instante 3 al 4 no hubo adopción por parte del agente, lo cual permite obtener un límite inferior de α: $\alpha \geq \frac{3}{10} = 0.3$. De este modo, la probabilidad de la observación para el agente del ejemplo será la siguiente:

$$p = \int_{0.3}^{0.4} f(\alpha) d\alpha$$

Siendo f(α) la función de densidad de probabilidad.

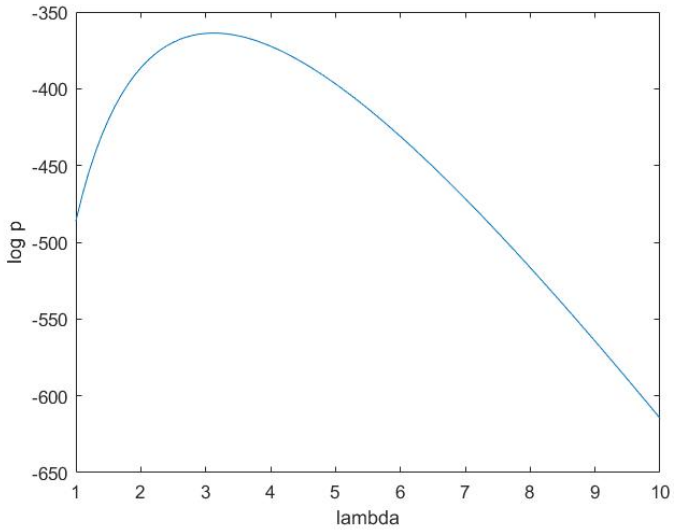
En el siguiente apartado se muestran diversos ejemplos de simulación que permiten comprobar el correcto funcionamiento del algoritmo de máxima verosimilitud.

4.4. Ejemplos de ejecución del algoritmo.

En este apartado se muestran ejemplos de ejecución del algoritmo. Para cada ejemplo, se indica primeramente el valor real del parámetro λ utilizado en la simulación real. Después, se muestra la gráfica del logaritmo de la probabilidad de observación frente al valor de λ que permite observar el máximo. Por último, se muestra una captura de Matlab que indica el valor de λ que maximiza la verosimilitud. Las simulaciones se han realizado haciendo un mallado de los parámetros de la distribución. En las siguientes figuras se muestran simulaciones para el caso de 1 y 2 parámetros, no obstante, también se podría aplicar el algoritmo a casos en los que hubiera un número más elevado de parámetros; en ese caso se buscaría el conjunto de parámetros que maximizan la verosimilitud mediante búsquedas locales.

Las siguientes gráficas se han realizado para un número de agentes igual a 500 y un número de iteraciones igual a 6.

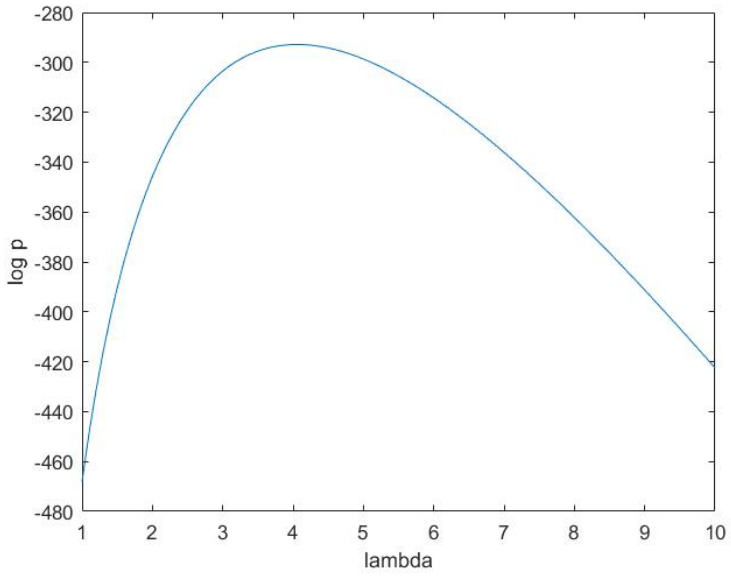
4.4.1. Simulación para $\lambda = 3$




```
lambda_solution =

    3.1000
```

Figura 4.5. Simulación para $\lambda = 3$

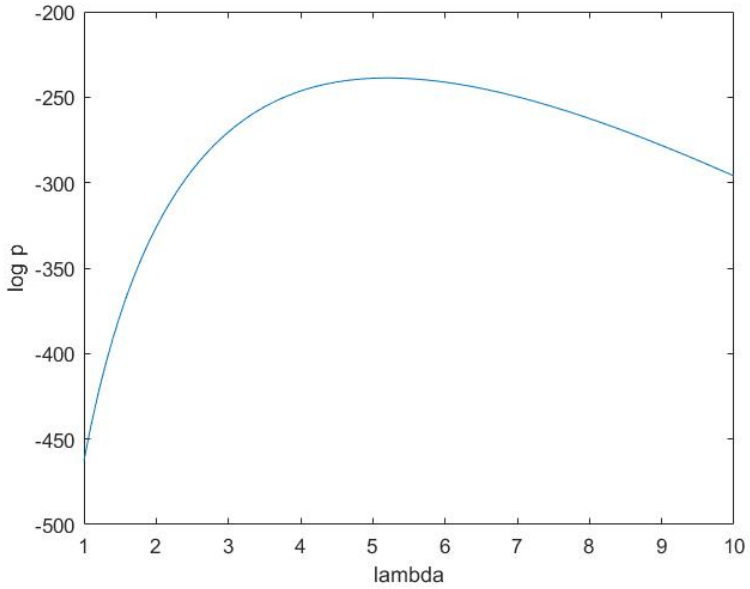
4.4.2. Simulación para $\lambda = 4$




```
lambda_solution =

    4.1000
```

Figura 4.6. Simulación para $\lambda = 4$

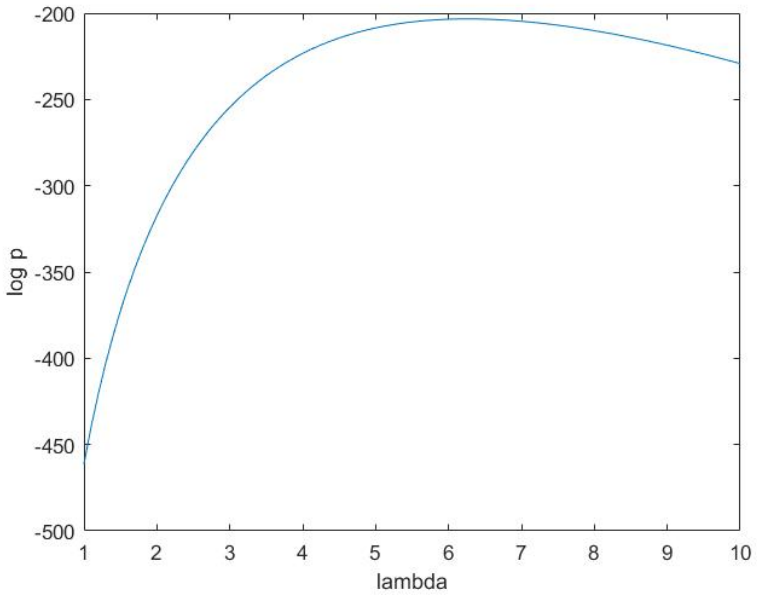
4.4.3. Simulación para $\lambda = 5$




```
lambda_solution =

    5.2000
```

Figura 4.7. Simulación para $\lambda = 5$

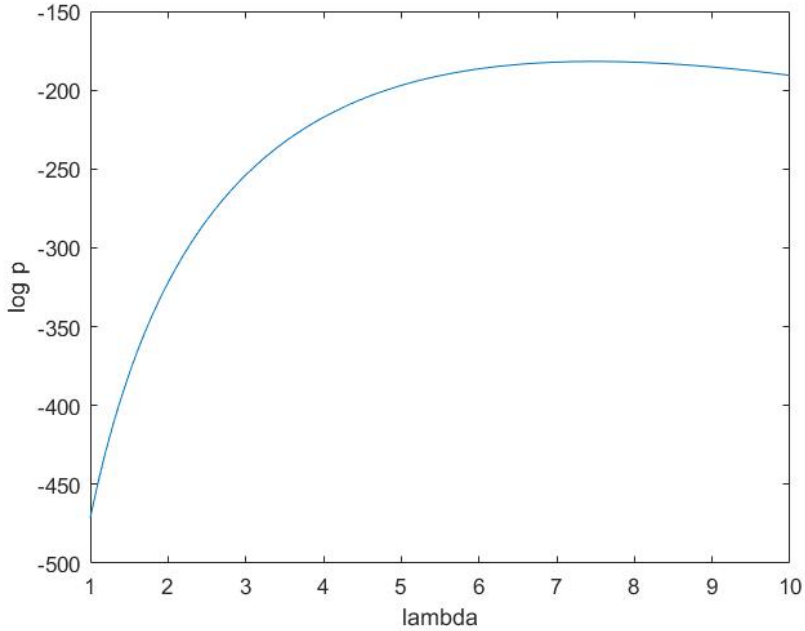
4.4.4. Simulación para $\lambda = 6$




```
lambda_solution =

    6.3000
```

Figura 4.8. Simulación para $\lambda = 6$

4.4.5. Simulación para $\lambda = 7$



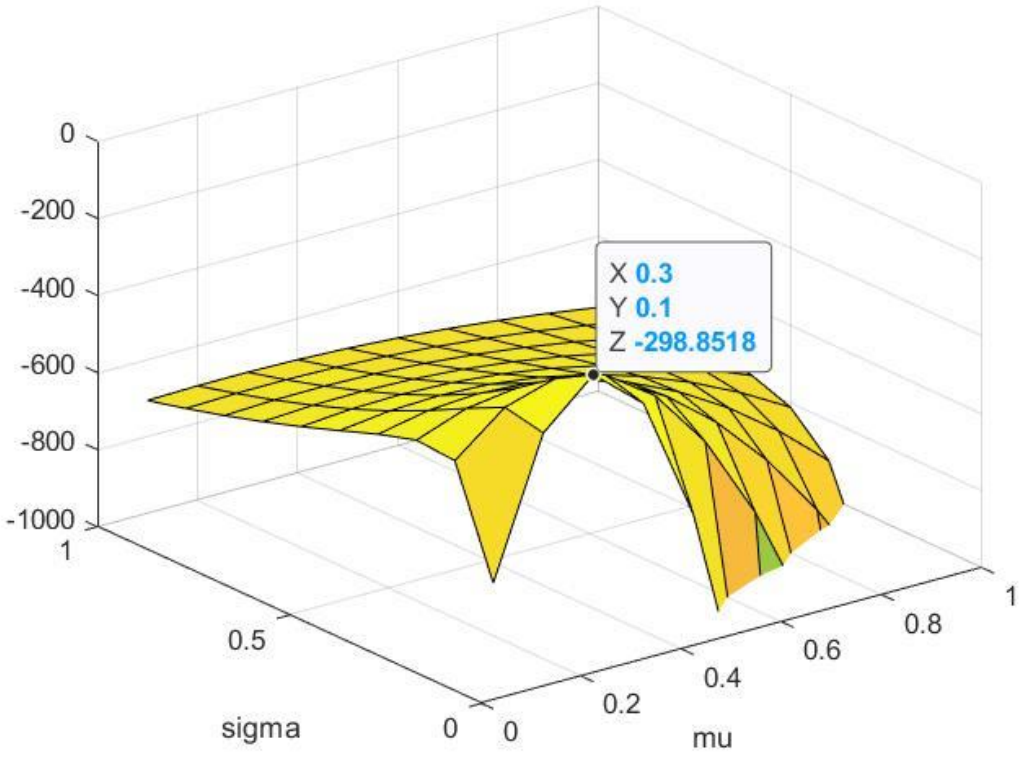

```
lambda_solution =

    7.5000
```

Figura 4.9. Simulación para $\lambda = 7$

Como se observa en las anteriores imágenes, el resultado del algoritmo es bastante cercano a la realidad, aunque no es exacto. El hecho de que el resultado obtenido no sea exacto no es motivo de preocupación, ya que, como se explicó en el apartado introductorio sobre el método de la máxima verosimilitud, se está maximizando una probabilidad, por lo que el resultado no siempre coincidirá exactamente con el valor del parámetro empleado. A continuación, se muestran más ejemplos de aplicación del algoritmo, esta vez utilizando una distribución normal para el parámetro α; en este caso, como la distribución posee 2 parámetros característicos, las gráficas de probabilidad serán en 3 dimensiones.

Al igual que en el caso anterior, el número de agentes utilizado en la simulación es de 500, y el número de iteraciones 6.

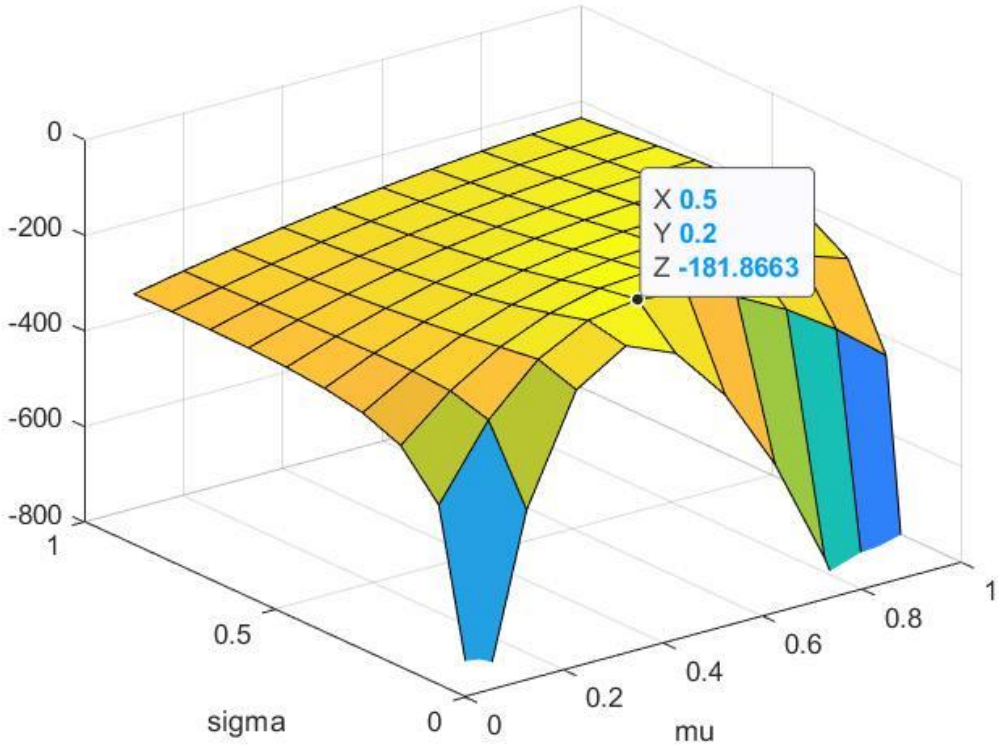
4.4.6. Simulación para $\mu = 0.3, \sigma = 0.1$




```
mu_solution =

    3


sigma_solution =

    0.1000
```

Figura 4.10. Simulación para $\mu = 3, \sigma = 0.1$

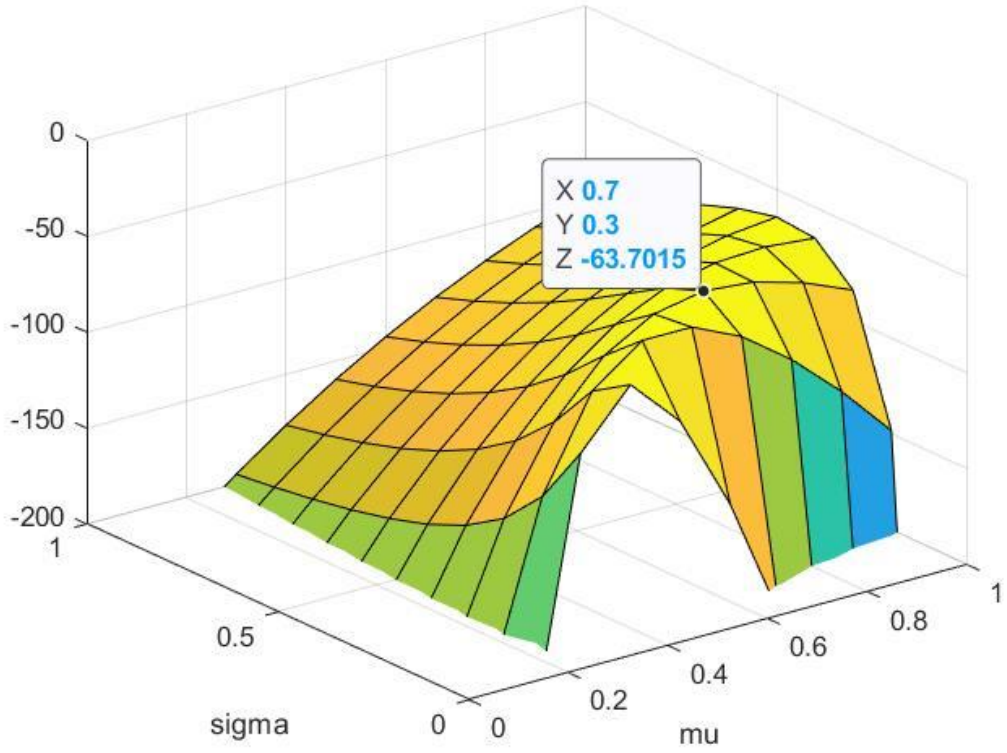
4.4.7. Simulación para $\mu = 0.5, \sigma = 0.2$




```
mu_solution =

    5


sigma_solution =

    0.2000
```

Figura 4.11. Simulación para $\mu = 5, \sigma = 0.2$

4.4.8. Simulación para $\mu = 0.7, \sigma = 0.3$




```
mu_solution =

     7


sigma_solution =

    0.3000
```

Figura 4.12. Simulación para $\mu = 7, \sigma = 0.3$

Como se ha observado en las gráficas anteriores, el algoritmo de máxima verosimilitud otorga resultados exactos en la mayoría de las ocasiones, y, en caso de no hacerlo, el valor otorgado es bastante cercano al real. Asimismo, se muestra la aplicación del algoritmo con dos distribuciones probabilísticas diferentes, obteniéndose buenos resultados en ambas.

El siguiente capítulo está dedicado a la adición de una nueva variable en el modelo de red; la probabilidad de éxito en la adopción. Mediante esta variable, podrán existir casos en los que, a pesar de que un agente posea una fracción de vecinos adoptantes igual o superior al valor de su parámetro α, dicho agente no se convertirá en adoptante. Asimismo, una vez realizada dicha modificación en el modelo de red, se implementará un algoritmo de máxima verosimilitud que permita estimar tanto los parámetros de la distribución probabilística utilizada, como el valor de la probabilidad de éxito en la adopción.

5 ALGORITMO DE MÁXIMA VEROSIMILITUD – ESTIMACIÓN DE LA PROBABILIDAD DE ÉXITO EN LA ADOPCIÓN

Este capítulo versa sobre la adición de un nuevo parámetro al modelo de red y la aplicación de un algoritmo de máxima verosimilitud para la identificación de ambos. En concreto, el nuevo parámetro corresponde a la probabilidad de éxito en la adopción. Este nuevo parámetro permitirá que, incluso teniendo un número de vecinos adoptantes suficiente, un agente de la red pueda no convertirse en adoptante en un determinado instante de la simulación.

5.1. Objetivos

El modelo utilizado hasta ahora es sencillo y eficiente para modelar la influencia que personas del entorno de un individuo pueden tener en este. La lógica empleada hasta ahora es bastante sencilla: cada agente espera a que el porcentaje de vecinos que han adoptado un determinado cambio sea mayor a su umbral de influencia, llamado α, y una vez ocurre esto, adopta el cambio y permanece así durante el resto de la simulación.

Sin embargo, a pesar de que este procedimiento es lógico y válido para diversos estudios de comportamientos sociales, puede existir el siguiente dilema: ¿Qué ocurriría si un determinado agente, a pesar de querer adoptar el cambio debido a la influencia de sus vecinos, no puede hacerlo?

Un ejemplo del dilema anterior es el de los vehículos eléctricos: un agente puede estar convencido de los beneficios de su compra, sin embargo, en ese momento no posee el dinero suficiente para su adquisición.

En este capítulo se introducirá un nuevo parámetro en la dinámica de red: la probabilidad de éxito en la adopción.

5.2. Modificaciones en el modelo de red

Para lograr la implementación anteriormente explicada, primeramente, se definió un parámetro llamado *umbral_exito*. Este parámetro corresponde a la probabilidad de adopción que tendrá cada agente tras haber alcanzado un porcentaje de vecinos igual o superior a α.

Después, una vez se han generado los adoptantes iniciales y se ha entrado en la dinámica de adopción, en cada iteración se genera un vector de dimensión n (igual al número de agentes) cuyas componentes son números aleatorios del 0 al 1. Si la componente i del vector es menor o igual al parámetro *umbral_exito*, el agente i tendrá la posibilidad de convertirse en adoptante en la siguiente iteración, siempre y cuando el porcentaje de vecinos adoptantes que posee sea igual o superior a α. De este modo, el parámetro *umbral_exito* se usa como probabilidad de éxito en la adopción.

El vector de números aleatorios que se compara con *umbral_exito* se renueva en cada iteración, de este modo, se permite la posibilidad de que un agente no adopte a pesar de poder hacerlo en un determinado instante de la simulación, pero si pueda hacerlo después.

En cuanto al funcionamiento de la dinámica, es idéntico al ya presentado en el capítulo 3 de este trabajo, con la diferencia de que a la comprobación de que el porcentaje de adoptantes sea mayor o igual al valor de α del agente que está siendo estudiado, se añade la comprobación de que su número aleatorio generado en la iteración presente sea menos o igual que *umbral_exito*

5.3. Implementación del algoritmo de máxima verosimilitud

El algoritmo de máxima verosimilitud que permite hallar tanto el parámetro característico de la distribución estadística utilizada para generar los valores de α como el valor del parámetro *umbral_exito* se ha implementado de la siguiente manera:

Se recorre la matriz historia, previamente explicada en capítulos anteriores. Para cada fila pueden darse 3 casos diferenciados que requieren cálculos diferentes:

1. **El agente no era adoptante inicialmente y se convirtió en adoptante:** si este es el caso, se generará una matriz que recoja las diferentes posibilidades que exista en términos de éxito y fracaso; es decir, si el agente ha sido adoptante en el instante 3 de la simulación, deben contemplarse las siguientes secuencias: [fracaso, fracaso, éxito]; [fracaso, éxito, éxito], [éxito, éxito, éxito]. Es importante recordar que, en este caso, éxito y fracaso se refieren al valor de la variable que modela la probabilidad de proceder a la adopción en caso de tener vecinos adoptantes suficientes, y no a que se haya producido o no la adopción propiamente dicha. Cabe destacar que el último componente de la secuencia debe ser éxito, ya que en caso contrario no se habría producido la adopción. El programa de Matlab codifica los éxitos como 1 y los fracasos como 0. Una vez hecho esto, se calcula el valor del límite superior de α como el porcentaje (en tanto por uno) de vecinos adoptantes en el instante en el que se produjo la adopción. Posteriormente, para cada una de las secuencias posibles, se calcula un límite inferior de α igual al porcentaje (en tanto por uno) de vecinos adoptantes en el último instante donde se obtuvo un éxito; por ejemplo, para la secuencia [0 0 1 0 0 1 0 0], el último éxito se produjo en la interacción número 6, por lo que el límite inferior de α se tomará como el tanto por uno de vecinos adoptantes que tenía en agente en la iteración número 6. Este proceso se repite para cada una de las secuencias posibles. La probabilidad de la observación para un agente determinado será la suma de las integrales de la función de densidad entre los límites de α calculados, multiplicada cada integral por la probabilidad de cada una de las secuencias.

2. **El agente no ha llegado a adoptar en ningún instante de la simulación:** en este caso, se procede de un modo análogo al anterior, pero el límite superior no se conoce, por lo que se integra la función de densidad hasta infinito. También es necesario considerar el caso, aunque en muchas ocasiones sobre todo para números de iteraciones altos, de que el agente nunca haya obtenido un número aleatorio superior al umbral de éxito en la adopción.

3. **El agente era uno de los adoptantes iniciales:** en este caso, el agente no ha necesitado influencia de vecinos ni estar dentro del umbral de éxito en la adopción, por lo que estos agentes no serán considerados a la hora de calcular la probabilidad de la adopción.

Cada valor de la probabilidad de la observación correspondiente a cada agente, exceptuando aquellos que eran adoptantes en el instante inicial, se guarda en un vector auxiliar. El valor de la probabilidad de la observación completa se obtiene multiplicando cada una de las componentes, o bien, tomando el logaritmo y sumándolas. Esta última opción es la elegida en el trabajo.

El proceso explicado hasta ahora se repite para cada valor del parámetro característico de la distribución estadística y para cada valor de la probabilidad de éxito en la adopción, guardando la probabilidad de la observación total en una matriz. El valor que maximiza la verosimilitud será aquel en el que la matriz de probabilidad sea máxima.

```
historia =

     1     0     0     0     1     1     1     1     1     1     1     1

cases =

     0     0     0
     0     0     1
     0     1     0
     0     1     1
     1     0     0
     1     0     1
     1     1     0
     1     1     1
```

Figura 5.1. Ejemplo de generación de secuencias posibles para un determinado agente

En la figura anterior se muestra el ejemplo de la generación de las secuencias para un determinado agente, el número 1. Como se puede observar, el agente no adoptó durante los 3 primeros instantes, por lo que cada secuencia tendrá 3 eventos. Como se explicó anteriormente, los ceros corresponden a fracasos y los unos a éxitos. Cabe destacar que, en la matriz de secuencias mostrada en la imagen, existen secuencias que poseen un cero en su última componente. Aunque esto no es posible ya que en el instante 3 de la simulación debió haberse obtenido un éxito, ya que de otra manera no podría haber sido adoptante en el instante siguiente, estas secuencias no son consideradas a la hora de calcular la probabilidad.

A continuación, en la tabla 5.1, se muestra un sencillo ejemplo de cómo el algoritmo calcularía la probabilidad de observación para un determinado agente dada la evolución temporal de sus vecinos, sus vecinos totales y las secuencias generadas explicadas en el párrafo anterior:

Número total de vecinos del agente: 10				
Estado del agente	0	0	0	1
Vecinos adoptantes del agente	1	2	4	6
Secuencia 1	0	0	1	-
Secuencia 2	0	1	1	-
Secuencia 3	1	0	1	-
Secuencia 4	1	1	1	-

Tabla 5.1. Ejemplo de cálculo de probabilidad de observación para un agente

Como se explicó anteriormente, la cota superior de α se obtiene del instante a partir del cual se produjo la adopción, y la cota inferior con el último éxito en las secuencias generadas previo al instante antedicho; en el ejemplo de la tabla 5.1 los límites de α serían los siguientes:

- Límite superior de α: $\alpha \leq \frac{4}{10} = 0.4$
- Límite inferior de α de acuerdo con la secuencia 1: $\alpha \geq 0$
- Límite inferior de α de acuerdo con la secuencia 2: $\alpha \geq \frac{2}{10} = 0.2$
- Límite inferior de α de acuerdo con la secuencia 3: $\alpha \geq \frac{1}{10} = 0.1$
- Límite inferior de α de acuerdo con la secuencia 4: $\alpha \geq \frac{2}{10} = 0.2$

Asimismo, la probabilidad asociada a cada secuencia sería:

- Secuencia 1: $p_{fracaso}^2 \cdot p_{éxito}$
- Secuencia 2: $p_{fracaso} \cdot p_{éxito}^2$
- Secuencia 3: $p_{fracaso} \cdot p_{éxito}^2$
- Secuencia 4: $p_{éxito}^3$

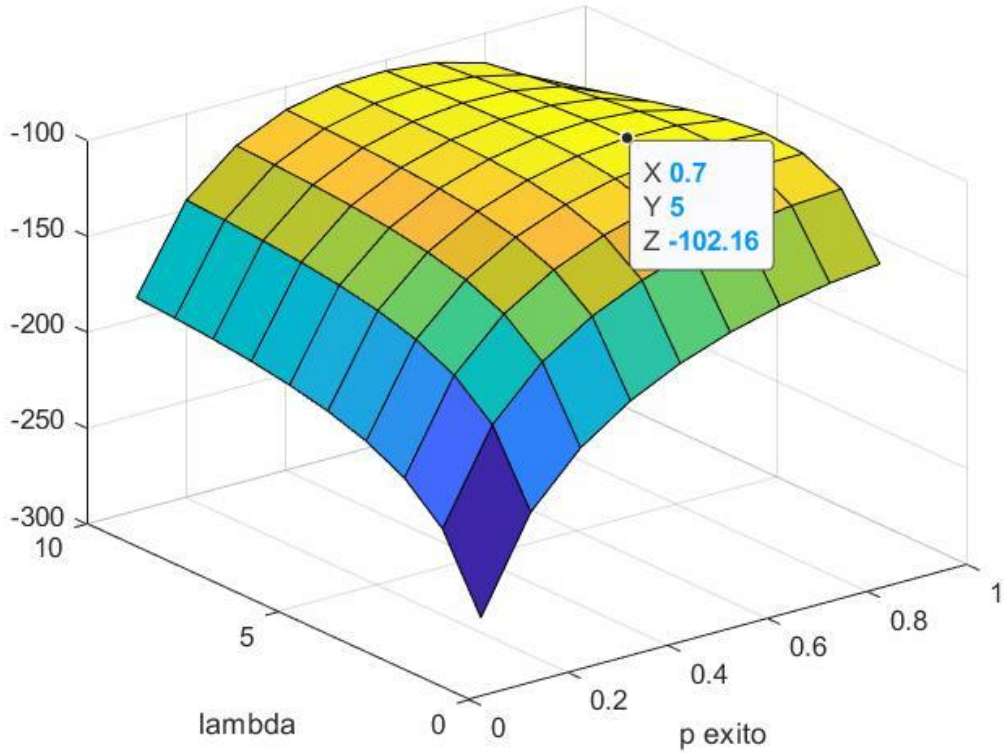
Resultando en una probabilidad de la observación para el agente estudiado:

$$p = p_{fracaso}^2 \cdot p_{éxito} \int_0^{0.4} f(\alpha)d\alpha + p_{fracaso} \cdot p_{éxito}^2 \int_{0.2}^{0.4} f(\alpha)d\alpha + p_{fracaso} \cdot p_{éxito}^2 \int_{0.1}^{0.4} f(\alpha)d\alpha + p_{éxito}^3 \int_{0.2}^{0.4} f(\alpha)d\alpha$$

5.4. Ejemplos de ejecución del algoritmo

En el presente apartado se muestran varios ejemplos de la ejecución del algoritmo de máxima verosimilitud programado, donde se puede apreciar tanto la gráfica con su máximo marcado, que corresponde a los valores de los parámetros que maximizan la verosimilitud, así como la solución que proporciona el programa de Matlab. Debido a que se ha añadido en este capítulo un parámetro extra, la distribución estadística que se ha utilizado en estos ejemplos de simulación es la exponencial, ya que consta de tan solo un parámetro estadístico, siendo así un total de 2 parámetros a hallar y, de esta manera, haciendo posible su representación gráfica en tres dimensiones. En todas las gráficas que se presentan a continuación, el número de agentes utilizado fue de 500, y el número de iteraciones de 10.

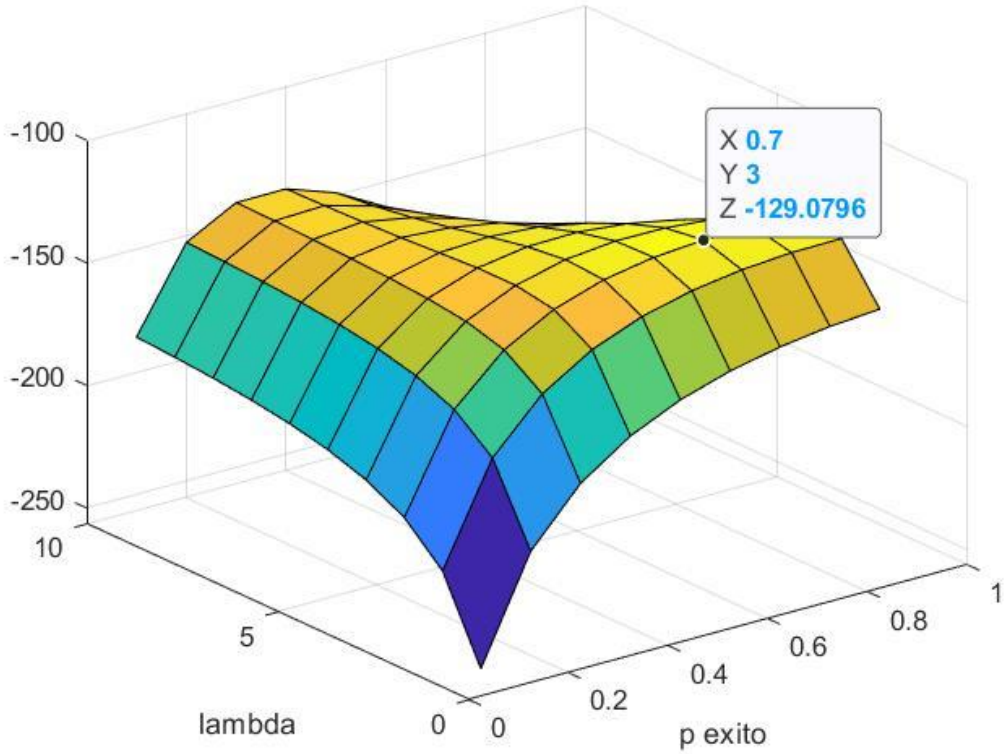
5.4.1. Simulación para $\lambda = 5$ y $p_{éxito} = 0.7$




```
lambda_solution =

     5


p_exito_solution =

    0.7000
```

Figura 5.2. Simulación para $\lambda = 5, p_{éxito} = 0.7$

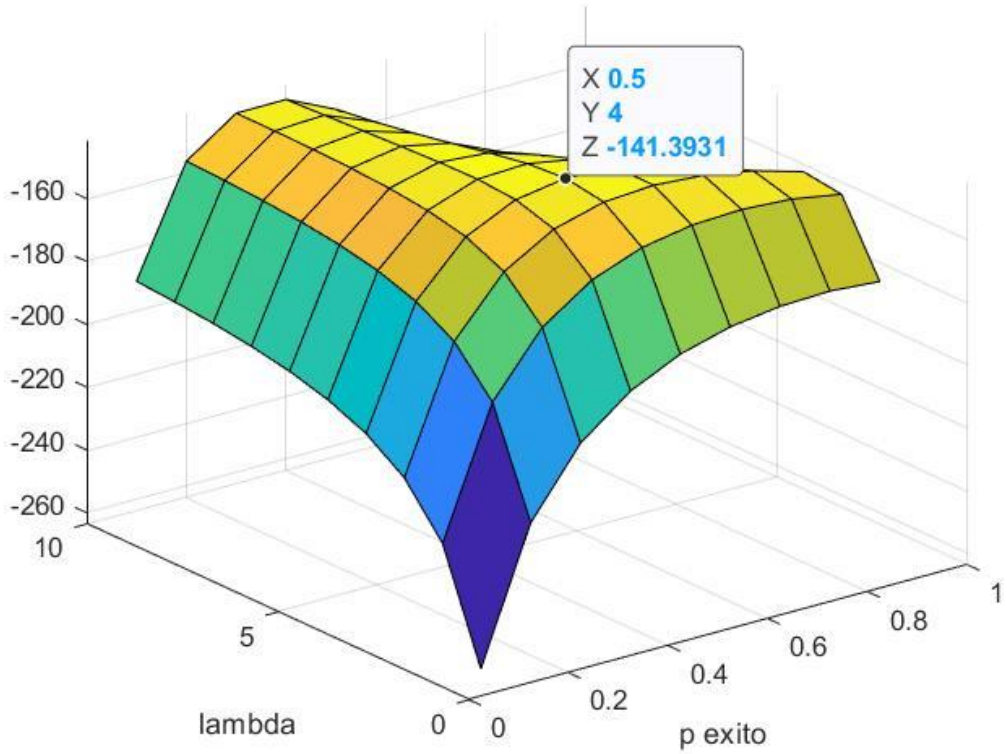
5.4.2. Simulación para $\lambda = 3$ y $p_{éxito} = 0.8$




```
lambda_solution =

     3


p_exito_solution =

    0.7000
```

Figura 5.3. Simulación para $\lambda = 3, p_{éxito} = 0.8$

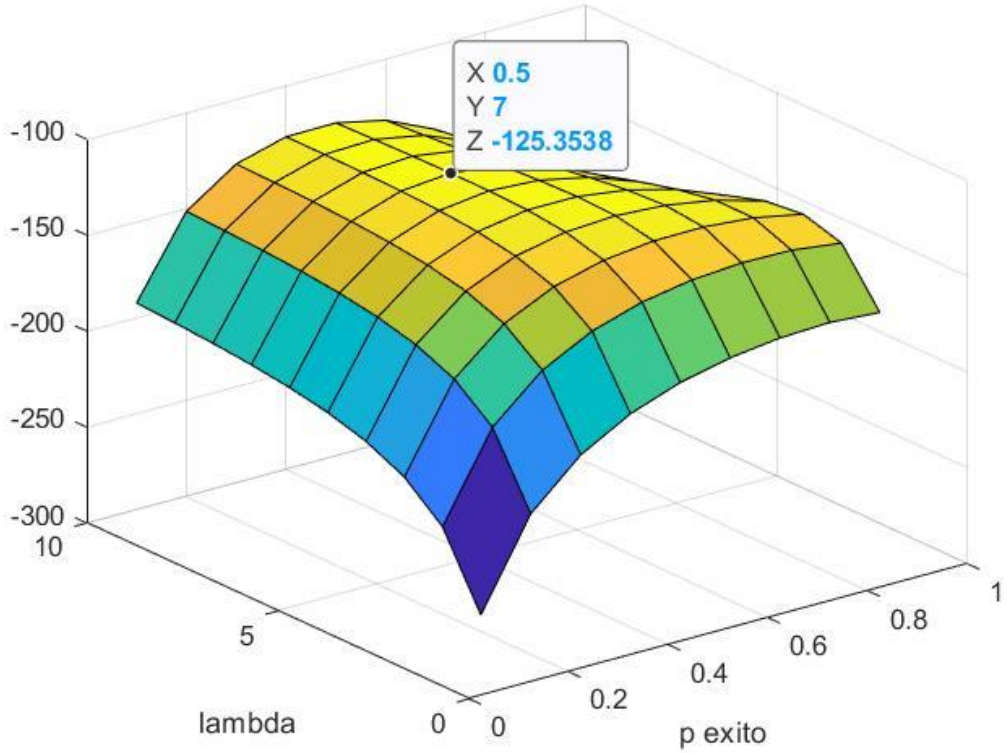
5.4.3. Simulación para $\lambda = 4$ y $p_{éxito} = 0.6$




```
lambda_solution =

     4


p_exito_solution =

    0.5000
```

Figura 5.4. Simulación para $\lambda = 4, p_{éxito} = 0.6$

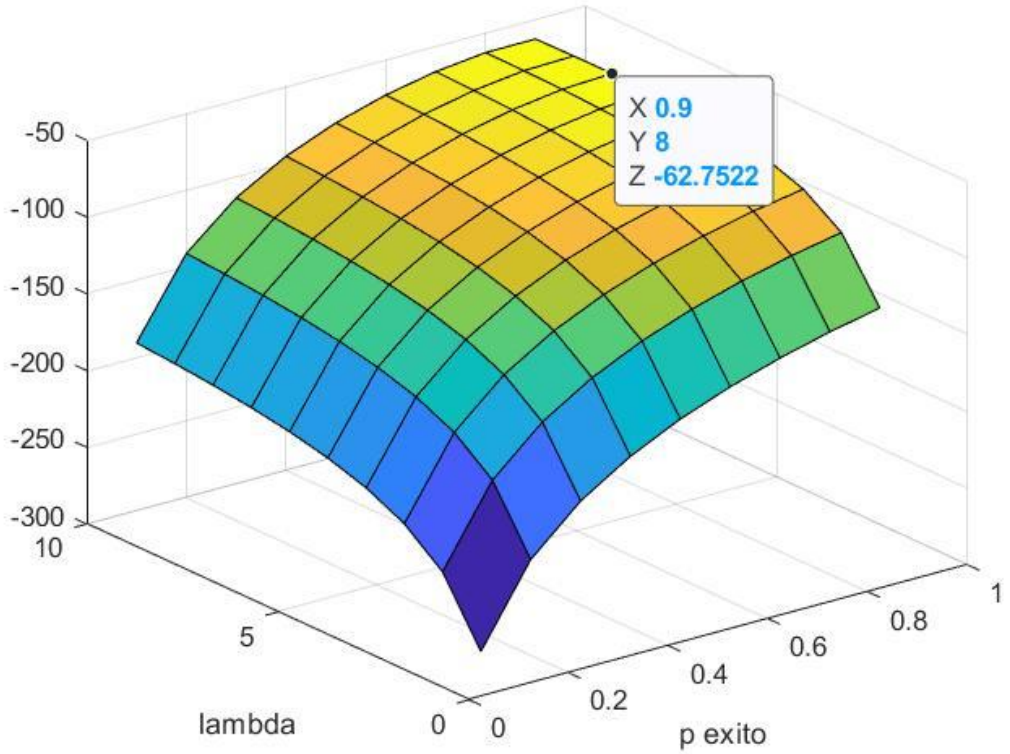
5.4.4. Simulación para $\lambda = 7$ y $p_{éxito} = 0.5$




```
lambda_solution =

     7


p_exito_solution =

    0.5000
```

Figura 5.5. Simulación para $\lambda = 7, p_{éxito} = 0.5$

5.4.5. Simulación para $\lambda = 9$ y $p_{éxito} = 0.9$




```
lambda_solution =

     8


p_exito_solution =

    0.9000
```

Figura 5.6. Simulación para $\lambda = 7, p_{éxito} = 0.5$

En el transcurso de este capítulo, se ha detallado la implementación de la nueva variable (probabilidad de éxito en la adopción cuando un agente tiene vecinos adoptantes suficientes) en el modelo de red, y el algoritmo de máxima verosimilitud para la estimación de los diferentes parámetros a partir de una simulación. Asimismo, se han realizado varias simulaciones que ponen de manifiesto el correcto funcionamiento del algoritmo; en varios de los casos se obtienen valores exactos en los parámetros y, en caso de no obtener dichos valores exactos, se obtienen valores bastante cercanos.

Debido a la forma en la que funciona el algoritmo de máxima verosimilitud detallado en este capítulo, en concreto debido a la generación de las secuencias de éxitos y fracasos posibles (se recuerda que en este caso éxito y fracaso se refiere a realizar o no la adopción cuando un agente tiene ya vecinos adoptantes suficientes para realizarla), el algoritmo puede ser bastante lento en algunos casos. En el capítulo siguiente se explicará el motivo y la implementación de un algoritmo mejorado que evite este problema.

6 ALGORITMO DE MÁXIMA VEROSIMILITUD – ESTIMACIÓN DE LA PROBABILIDAD DE ÉXITO EN LA ADOPCIÓN SIN USO DE COMBINATORIA

En este capítulo se describirá una mejora al anterior algoritmo de máxima verosimilitud para la determinación de 2 parámetros; el parámetro característico de la distribución exponencial que gobierna el valor de α de cada agente, y la probabilidad de éxito en la adopción.

6.1. Motivación y objetivos

El algoritmo de máxima verosimilitud descrito en el capítulo anterior necesitaba para su funcionamiento generar todas las secuencias posibles de éxitos y fracasos según el instante de adopción de un determinado agente. Es decir, si un agente se convierte en adoptante tras el primer instante de la simulación, significa que ha obtenido un éxito a la primera. Por otra parte, si se convierte en adoptante en la segunda iteración, puede significar que, o bien ha obtenido un fracaso y luego un éxito, o bien que ha obtenido dos éxitos, pero en la primera iteración no cumplía el requisito de tener un porcentaje de vecinos adoptantes igual o superior a su α.

Si bien esta forma de proceder puede ser efectiva para valores de λ altos y valores de $p_{éxito}$ elevados, si la simulación se realiza con un valor de λ reducido, y/o con un valor de $p_{éxito}$ bajo, habrá muchos agentes que tengan que esperar varias iteraciones de simulación antes de adoptar. Esto provocará que el número de secuencias posibles de éxitos y fracasos pueda ser muy elevado. En la tabla inferior se muestra el número de secuencias generadas según el instante de la iteración en el que un determinado agente se convierte en adoptante:

Instante de adopción	Número de secuencias generadas
1	0
2	1
3	4
4	8
5	16
n	2^{n-1}

Tabla 6.1. Secuencias generadas en función del instante de adopción

Como se puede observar en la tabla anterior, el número de secuencias que deben generarse crece de forma exponencial a medida que crece el instante de adopción, por este motivo, si la simulación inicial se ha hecho con valores de λ y/o $p_{éxito}$ bajos, significará en agentes que adoptan tarde, por lo que el

tiempo de cálculo del algoritmo será muy elevado.

6.2. Eliminación de la combinatoria

Para evitar el uso de la combinatoria que provoca una gran ralentización del algoritmo en ciertos casos, se ha partido del siguiente principio:

El algoritmo anterior generaba los casos posibles, calculaba una probabilidad de cada uno de ellos, y posteriormente calculaba la probabilidad de la observación sumando cada uno de los casos. En el algoritmo cuya implementación se discute en este capítulo, se sigue el procedimiento contrario: a partir de la secuencia de cada agente, se generan varios intervalos y se asocia a ellos una determinada probabilidad calculada a partir de la probabilidad de obtener un éxito y la probabilidad de obtener un fracaso. Para facilitar su explicación conviene ayudarse del siguiente ejemplo:

$\frac{vecinos\ adoptantes}{vecinos\ totales}$	0.1	0.2	0.3	0.4	0.5	0.6
Agente i	0	0	0	0	1	1

Tabla 6.2. Ejemplo de secuencia de un agente para la ilustración del algoritmo mejorado

El algoritmo mejorado consiste en generar intervalos en los que puede encontrarse el parámetro α del agente que está siendo estudiado, comenzando por el 0. Cada vez que la proporción entre vecinos adoptantes y vecinos totales cambie de un instante a otro, se generará un nuevo intervalo. El límite inferior del nuevo intervalo será el límite superior del intervalo anterior. El último intervalo que se generará será el que contenga la proporción de vecinos adoptantes entre vecinos totales correspondientes al instante previo a la adopción como límite inferior, y dicha proporción en el instante de adopción. En el caso del ejemplo, los intervalos generados serían los siguientes:

$$[0\ 0.1]$$
$$(0.1\ 0.2]$$
$$(0.2\ 0.3]$$
$$(0.3\ 0.4]$$

Emparejado a cada intervalo, está asociada una determinada probabilidad de secuencia. El último elemento de dicha secuencia siempre será una probabilidad de éxito, ya que de otra manera no podría haberse producido la adopción. Multiplicando a dicha probabilidad de éxito, se encuentra la probabilidad de fracaso elevada a la diferencia entre el instante de adopción y el instante actual. Por ejemplo, asociada al intervalo [0 0.1] está la secuencia $p_{fracaso}^{3} \cdot p_{éxito}$. Esto es debido a que, si se considera que el parámetro α del agente siendo estudiado debe encontrarse en el intervalo [0 0.1), este agente ha tenido 4 oportunidades de realizar la adopción: los instantes 1, 2, 3 y 4 de la simulación. Por otra parte, asociada al intervalo (0.3 0.4] se asocia la secuencia $p_{éxito}$, ya que es la única compatible con lo que se ha observado (el agente ha adoptado con una proporción de vecinos adoptantes respecto a totales de 0.4, por lo que ha debido forzosamente obtener el éxito).

A continuación, se muestran las secuencias asociadas a cada intervalo del ejemplo:

$$[0\ 0.1] \rightarrow\ p_{fracaso}^{3} \cdot p_{éxito}$$
$$(0.1\ 0.2] \rightarrow p_{fracaso}^{2} \cdot p_{éxito}$$
$$(0.2\ 0.3] \rightarrow p_{fracaso}^{1} \cdot p_{éxito}$$

$$(0.3\ 0.4] \rightarrow p_{éxito}$$

La probabilidad de la observación para cada agente, por tanto, será la suma de las integrales de la función de densidad de probabilidad de cada uno de los intervalos generados, multiplicados por sus correspondientes secuencias, es decir, para el caso del ejemplo:

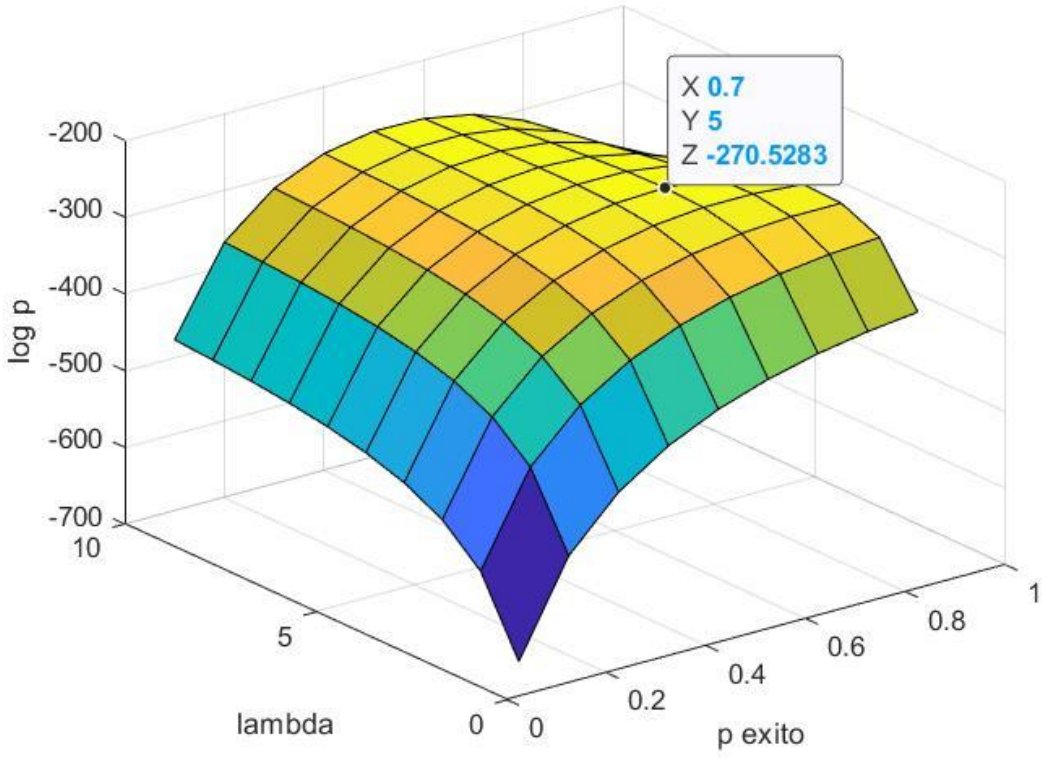
$$\left(\int_0^{0.1} f(\alpha)d\alpha\right) p_{fracaso}^3 \cdot p_{éxito} + \left(\int_{0.1}^{0.2} f(\alpha)d\alpha\right) p_{fracaso}^2 \cdot p_{éxito} + \\ + \left(\int_{0.2}^{0.3} f(\alpha)d\alpha\right) p_{fracaso} \cdot p_{éxito} + \left(\int_{0.3}^{0.4} f(\alpha)d\alpha\right) p_{éxito}$$

De este modo, se logra calcular la probabilidad de la observación sin necesidad del uso de grandes números de secuencias que ralentizan enormemente el algoritmo en caso de adopciones tardías.

6.3. Ejemplos de ejecución del algoritmo

En el presente apartado se muestran varios ejemplos de la ejecución del algoritmo mejorado, donde se puede apreciar tanto la gráfica con su máximo marcado, que corresponde a los valores de los parámetros que maximizan la verosimilitud, así como la solución que proporciona el programa de Matlab. En todas las gráficas que se muestran a continuación, se ha utilizado distribución exponencial, un número de agentes en la red igual a 500, y un número de iteraciones igual a 10.

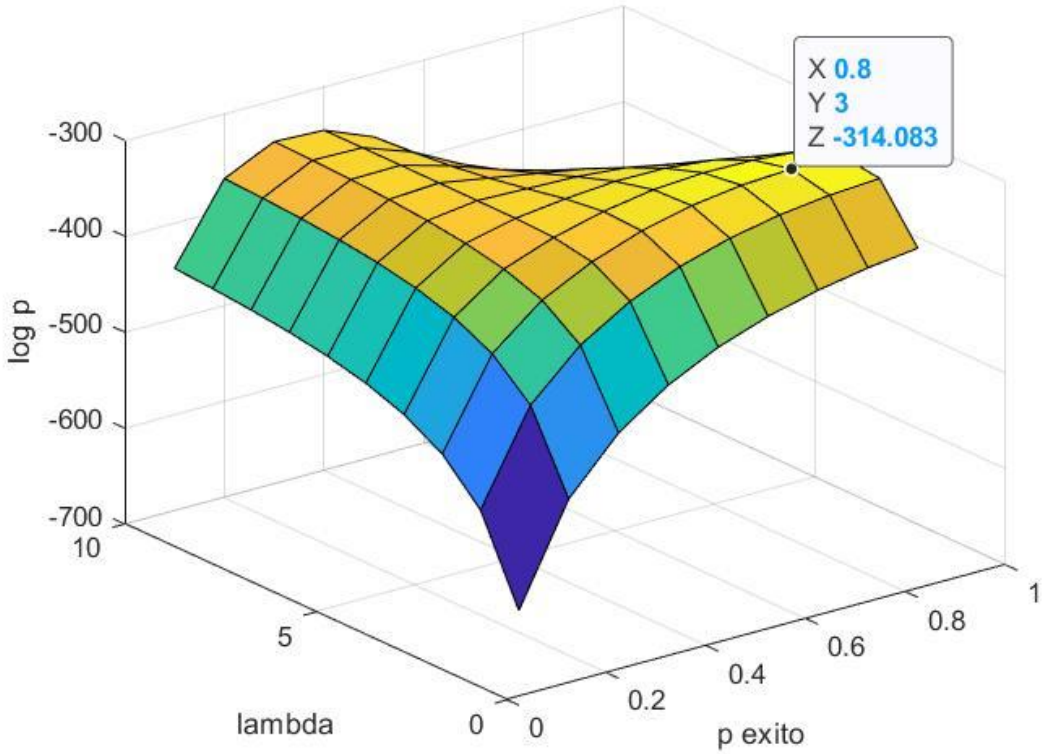
6.3.1. Ejecución para $\lambda = 5, p_{éxito} = 0.7$




```
lambda_solution =

     5


p_exito_solution =

    0.7000
```

Figura 6.1. Ejecución para $\lambda = 5, p_{éxito} = 0.7$

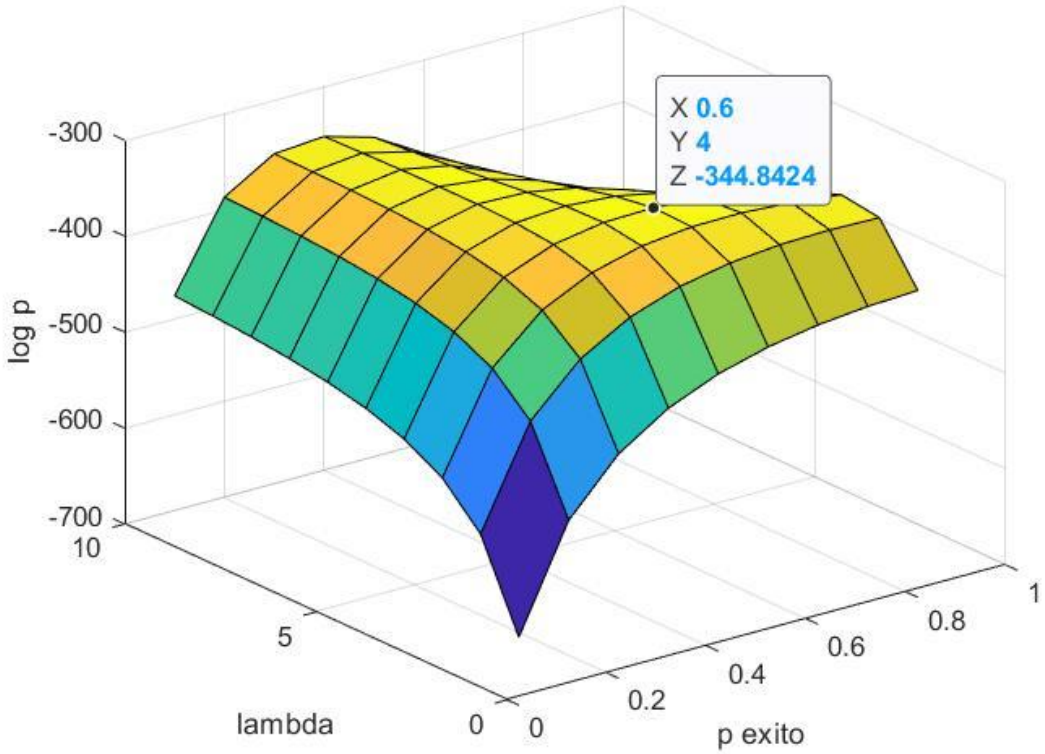
6.3.2. Ejecución para $\lambda = 3, p_{éxito} = 0.8$




```
lambda_solution =

     3


p_exito_solution =

    0.8000
```

Figura 6.2. Ejecución para $\lambda = 3, p_{éxito} = 0.8$

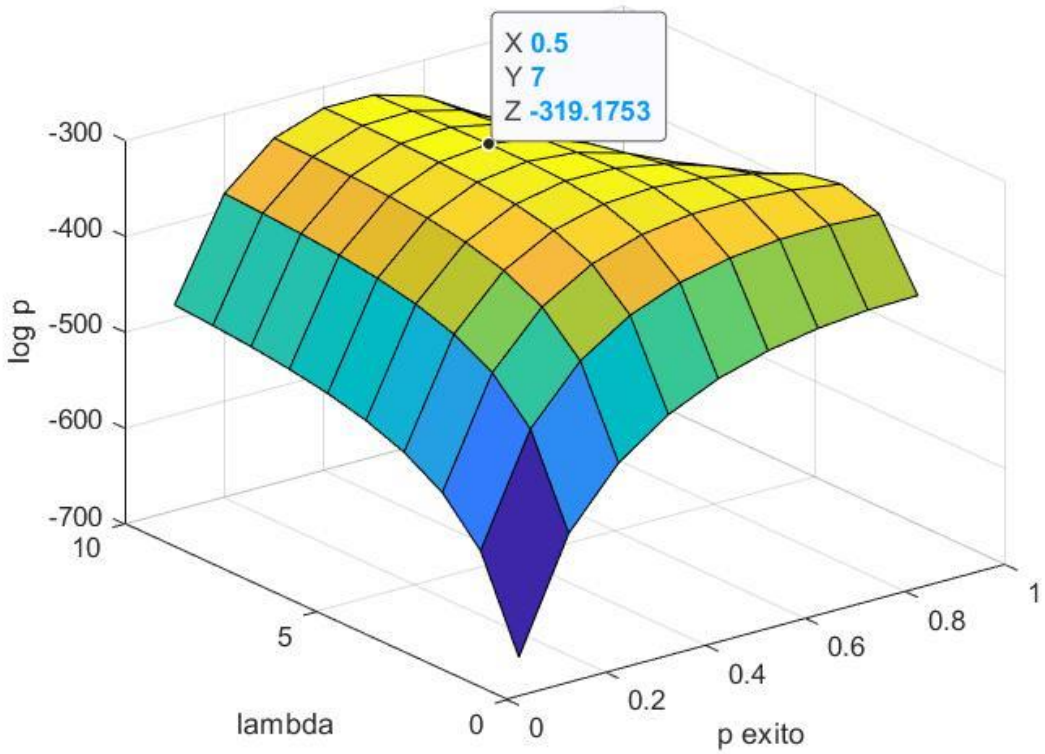
6.3.3. Ejecución para $\lambda = 4, p_{éxito} = 0.6$




```
lambda_solution =

     4


p_exito_solution =

    0.6000
```

Figura 6.3. Ejecución para $\lambda = 4, p_{éxito} = 0.6$

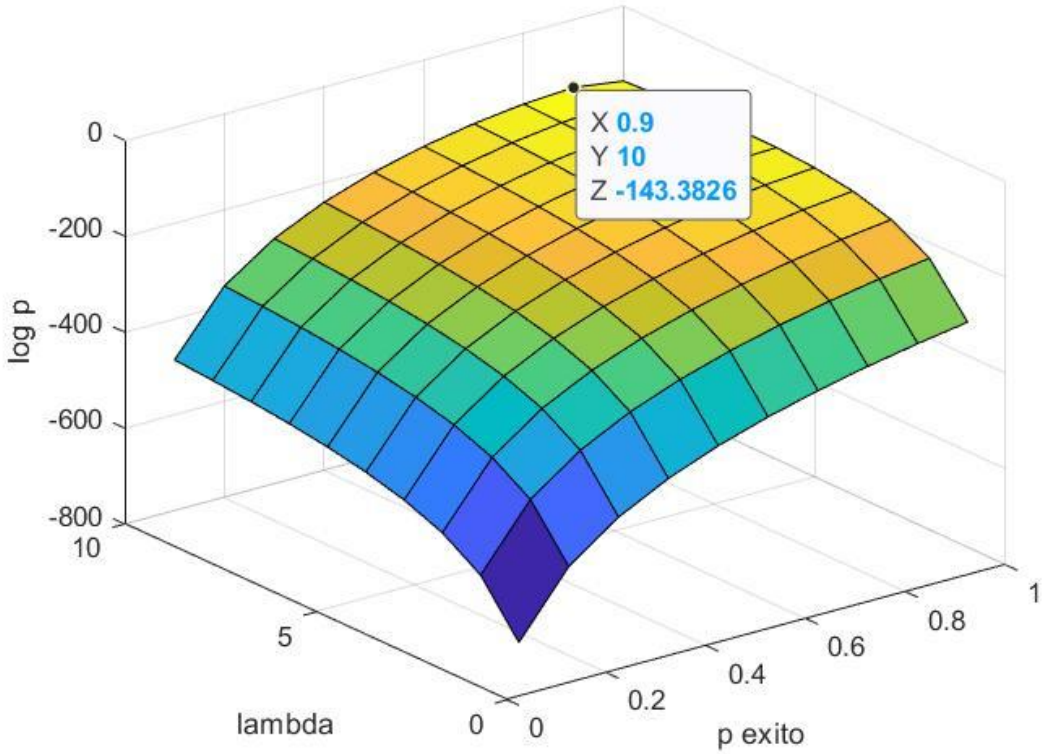
6.3.4. Ejecución para $\lambda = 7, p_{éxito} = 0.5$




```
lambda_solution =

     7


p_exito_solution =

    0.5000
```

Figura 6.4. Ejecución para $\lambda = 7, p_{éxito} = 0.5$

6.3.5. Ejecución para $\lambda = 9, p_{éxito} = 0.9$




```
lambda_solution =

    10


p_exito_solution =

    0.9000
```

Figura 6.5. Ejecución para $\lambda = 9, p_{éxito} = 0.9$

6.4. Comparativa en términos de tiempo de ejecución de los dos algoritmos.

Como se explicó en la introducción de este capítulo, el propósito de la mejora del algoritmo era evitar que grandes combinatorias generadas cuando se producen adopciones tardías ralentizaran en exceso la aplicación del algoritmo. En este apartado se muestran varias tablas comparativas de los tiempos de ejecución de ambos algoritmos (con y sin combinatoria), que permiten apreciar la mejora obtenida con el nuevo algoritmo.

Red con n=200 agentes		
Tiempo de ejecución – Algoritmo con combinatoria (s)	**Tiempo de ejecución – Algoritmo sin combinatoria (s)**	
30	14	$\boldsymbol{\lambda = 8, p_{éxito} = 0.8}$
33	15	$\boldsymbol{\lambda = 7, p_{éxito} = 0.8}$
36	15	$\boldsymbol{\lambda = 6, p_{éxito} = 0.8}$
43	16	$\boldsymbol{\lambda = 5, p_{éxito} = 0.8}$
558	18	$\boldsymbol{\lambda = 4, p_{éxito} = 0.8}$
1887	26	$\boldsymbol{\lambda = 3, p_{éxito} = 0.8}$

Tabla 6.3. Comparativa entre algoritmos para diferentes valores de λ

Como se puede observar en la tabla anterior, mientras que el tiempo de ejecución apenas varía en el caso del algoritmo sin combinatoria, en el caso del algoritmo con combinatoria crece de forma muy rápida a medida que disminuye el valor del parámetro λ. Esto es debido a que cuanto menor sea dicho parámetro, mayor será el número de agentes que se concentra en torno a valores bajos de α, y por consiguiente, mayor será el número de agentes que adopte en instantes tardíos de la simulación, provocando que se deban generar combinatorias más grandes y en consecuencia ralentizando la simulación.

Nota: para realizar una comparación fidedigna, la red utilizada, así como la dinámica de adopción (en concreto, la matriz historia que informa del estado de cada agente en cada instante de la simulación) ha sido siempre la misma para todos los casos de la tabla anterior.

A continuación, se muestra una gráfica comparativa de los tiempos de ejecución para ambos algoritmos, según el valor de lambda. En ella se puede apreciar el crecimiento mucho más rápido del tiempo de ejecución en el caso del algoritmo con combinatoria

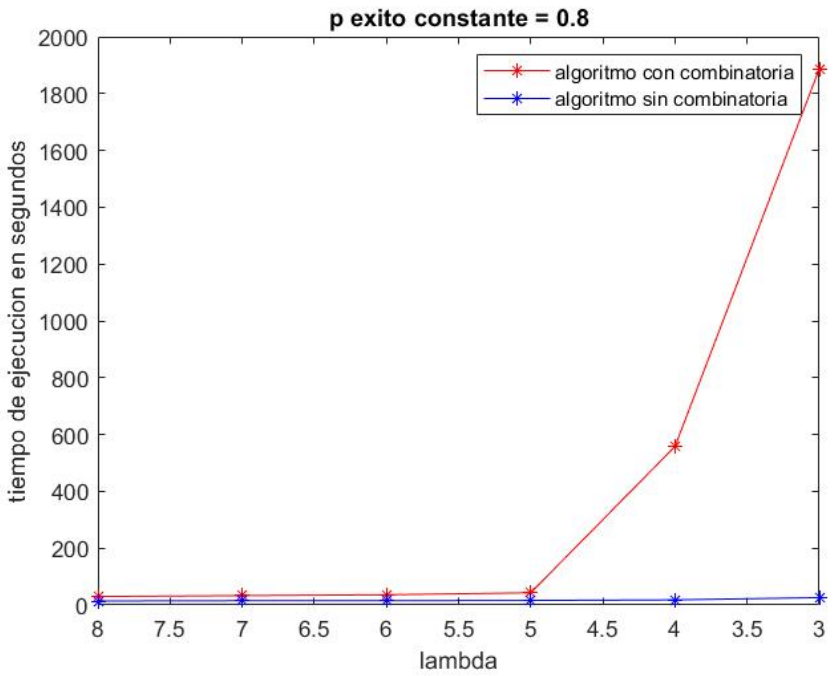


Figura 6.6. Comparativa entre algoritmos con variación de λ para una red de 200 agentes

---

Red con n=200 agentes		
Tiempo de ejecución – Algoritmo con combinatoria (s)	**Tiempo de ejecución – Algoritmo sin combinatoria (s)**	
30	14	$\boldsymbol{\lambda = 8, p_{éxito} = 0.8}$
41	16	$\boldsymbol{\lambda = 8, p_{éxito} = 0.7}$
52	18	$\boldsymbol{\lambda = 8, p_{éxito} = 0.6}$
308	21	$\boldsymbol{\lambda = 8, p_{éxito} = 0.5}$
560	23	$\boldsymbol{\lambda = 8, p_{éxito} = 0.4}$
1365	30	$\boldsymbol{\lambda = 8, p_{éxito} = 0.3}$

Tabla 6.4. Comparativa entre algoritmos para diferentes valores de $p_{éxito}$

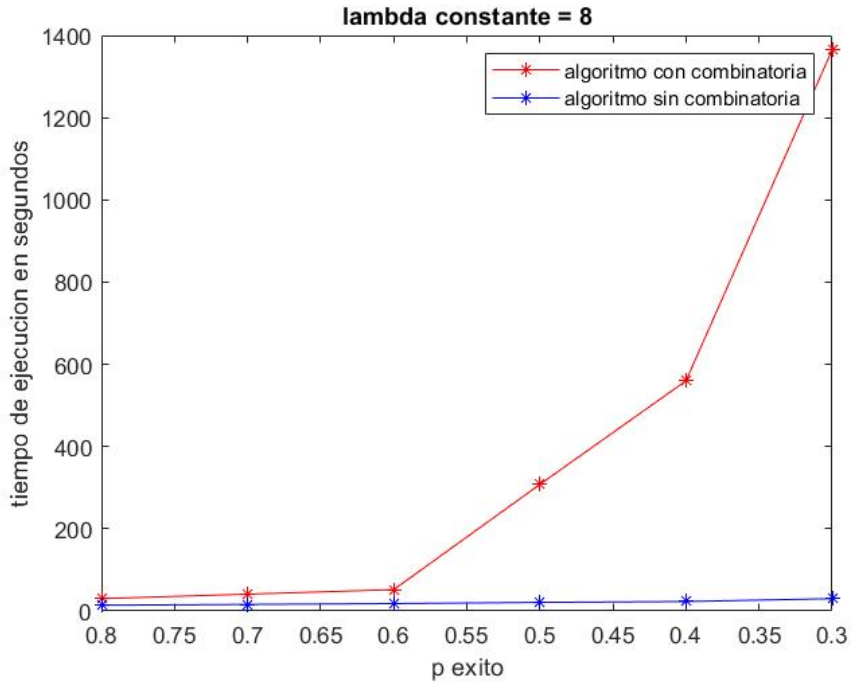


Figura 6.7. Comparativa entre algoritmos con variación de $p_{éxito}$ para una red de 200 agentes

Como se observa en la imagen, el efecto de variación de $p_{éxito}$ (es decir, el parámetro que indica la probabilidad de que un agente proceda a la adopción o no lo haga una vez tiene vecinos adoptantes suficientes) tiene un efecto parecido al de variación de λ; a medida que disminuye su valor, en el caso del algoritmo con combinatoria, el tiempo de ejecución aumenta drásticamente.

$\boldsymbol{\lambda = 8, p_{éxito} = 0.8}$		
Tiempo de ejecución – Algoritmo con combinatoria (s)	**Tiempo de ejecución – Algoritmo sin combinatoria (s)**	
13	8	**Red con 100 agentes**
30	14	**Red con 200 agentes**
88	26	**Red con 300 agentes**
240	45	**Red con 500 agentes**
1515	164	**Red con 1000 agentes**
14786	973	**Red con 2000 agentes**

Tabla 6.5. Comparativa entre algoritmos para diferentes valores de tamaño de red

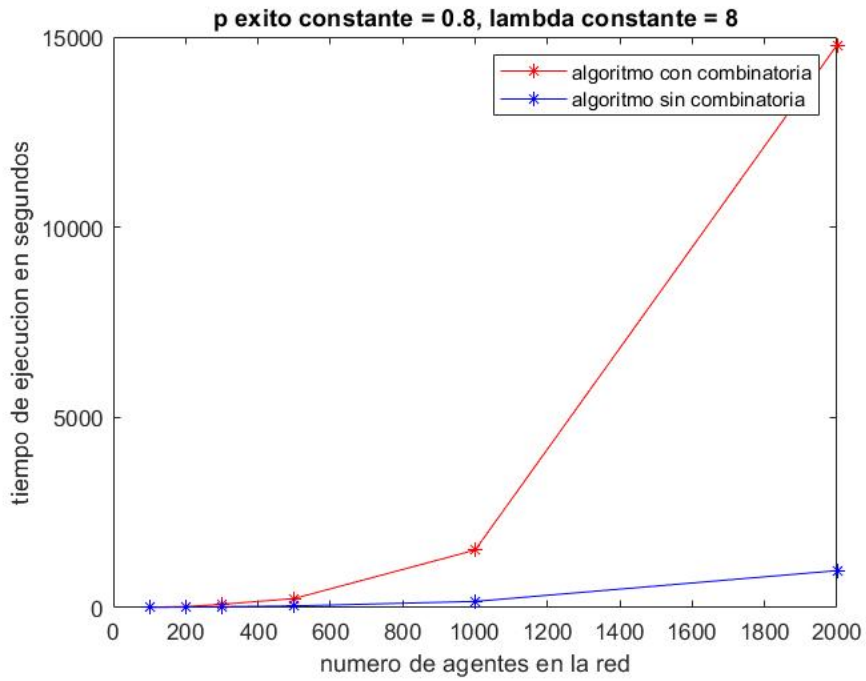


Figura 6.8. Comparativa entre vecinos con variación del número de agentes de la red

De la comparativa anterior se puede observar que, al aumentar el número de agentes de la red, el efecto sobre el algoritmo sin combinatoria es más notable que al disminuir $p_{éxito}$ o λ. Esto es debido a que, a medida que se aumenta el tamaño de la red (número de agentes) se tienen que realizar más cálculos de intervalos e integraciones de la función de densidad de probabilidad, lo cual no era necesario hacerlo en las simulaciones comparativas anteriores ya que el número de agentes era constante. Sin embargo, este efecto de ralentización al aumentar el tamaño de la red afecta en mucha mayor medida al algoritmo con combinatoria, por lo que también supone una mejora el nuevo algoritmo en términos de tamaño de la red y velocidad de ejecución.

Durante el transcurso de este capítulo, se ha explicado por qué el algoritmo utilizado en el capítulo anterior presentaba tiempos de ejecución muy elevados y se ha detallado la solución a este problema mediante la eliminación de la combinatoria que crecía de forma exponencial con el instante de adopción. Posteriormente, se realizaron varias simulaciones que permiten comprobar el correcto funcionamiento del algoritmo, y por último, se realizaron comparativas entre ambos algoritmos en términos de tiempo de ejecución, poniendo de manifiesto las grandes ganancias en cuanto a rapidez tras haber mejorado el algoritmo.

El siguiente capítulo versa sobre la introducción de una nueva variable en el modelo de red: la probabilidad de que un agente adoptante pase a ser no adoptante. Durante la introducción de dicho capítulo, se expondrán ejemplos en los que este fenómeno podría tener lugar justificando así la introducción de la variable. Una vez explicada la modificación, se explicará el algoritmo de máxima verosimilitud implementado para hallar el valor de la nueva variable (así como los valores de los demás parámetros) y se incluirán ejemplos de simulaciones.

7 ALGORITMO DE MÁXIMA VEROSIMILITUD CON CAMBIO DE OPINIÓN DE LOS AGENTES

En este capítulo se describirá una nueva modificación al modelo de red que conlleva la adición de un nuevo parámetro, así como la implementación de un algoritmo de máxima verosimilitud para hallar los parámetros que gobiernan el problema.

7.1. Motivación y objetivos

La implementación de la variable que modelaba la probabilidad de éxito en la adopción constituyó un importante paso a la hora de otorgar realismo al modelo de comportamiento social en que se basa este trabajo. En este capítulo, se explica la introducción de una variable adicional que incrementa dicho realismo aun más, y permite el modelado de problemas que antes no eran posibles sin ella.

La variable recién mencionada es la probabilidad de pasar de ser adoptante a no adoptante. Como se explicó en capítulos anteriores, el modelo original solo contempla el cambio de estado desde no adoptante hasta adoptante, pero no al revés; gracias a esta nueva variable, los agentes podrán pasar de ser adoptantes a no serlos, de forma que no influirán a sus vecinos a la hora de decidir.

Son varios los ejemplos de la vida real en los que puede haber un cambio como el mencionado en el párrafo anterior. Uno de ellos sería una enfermedad infecciosa. Un individuo puede estar sano; al entrar en contacto con individuos enfermos puede contagiarse, con mayor o menor facilidad según lo susceptible que sea a la infección (esto se modela con el parámetro α), y las medidas de precaución que tome (lo cual podría modelarse con la probabilidad de éxito). Una vez contagiado, permanecería enfermo durante un tiempo, pero finalmente se curaría y dejaría de ser un agente infeccioso. Esto último lo modela la variable descrita en el párrafo anterior. Otro ejemplo es el de una persona que decide comprar un determinado producto influido por sus vecinos, pero que tiene una mala experiencia con su uso, por lo que decide venderlo o desecharlo.

En los apartados siguientes se explicará la modificación al modelo de red que permite el cambio de adoptante a no adoptante de un determinado agente, la aplicación del algoritmo de máxima verosimilitud para el cálculo del valor de la probabilidad de decadencia, y varios ejemplos de simulación que muestran el funcionamiento y resultados otorgados por el algoritmo.

7.2. Modificación del modelo de red

La implementación de la probabilidad de decadencia se hizo de un modo muy parecido al de la implementación de la probabilidad de éxito descrito en capítulos anteriores.

Durante cada iteración, una vez se han calculado los nuevos adoptantes, se genera un número aleatorio comprendido entre 0 y 1 para cada uno de los agentes de la red. Posteriormente, si el agente es adoptante, se comprueba si el número aleatorio generado para él es menor que la probabilidad de decadencia previamente declara; si lo es, se procede a asignar a cada una de las componentes de la matriz historia correspondiente a su fila para las iteraciones restantes un valor de -1. El motivo de usar el valor -1, es evitar que pueda volver a pasar de no adoptante a adoptante (ya que esto solo puede hacerlo en caso de que en el instante presente de la simulación su valor de historia sea 0). Una vez terminada la simulación, los valores -1 de la matriz historia se sustituyen por ceros. A continuación, se muestra un ejemplo de la matriz historia donde ocurren decadencias entre los agentes:

historia =

1	0	0	1	1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1	1	1	1	1
3	0	0	1	1	1	1	1	1	1	1	1
4	0	0	0	1	1	1	0	0	0	0	0
5	0	0	1	0	0	0	0	0	0	0	0
6	0	0	1	1	1	1	1	1	1	1	1
7	0	1	1	1	1	1	1	1	1	1	1
8	1	1	1	1	1	1	1	1	1	1	1
9	0	0	1	1	1	1	0	0	0	0	0
10	0	0	0	1	1	1	1	1	1	1	1
11	0	0	1	1	1	1	1	1	1	1	1
12	0	0	1	1	1	1	0	0	0	0	0
13	1	1	1	1	1	1	1	1	1	1	1
14	0	0	1	1	1	1	1	1	1	1	1
15	0	0	1	1	1	1	1	0	0	0	0
16	1	1	1	1	1	1	1	1	0	0	0
17	0	0	1	1	1	0	0	0	0	0	0
18	0	1	1	1	1	1	1	1	1	1	1

Figura 7.1. Matriz historia con decadencia

En el ejemplo de la figura anterior, los agentes 4, 5, 9, 12, 15, 16 y 17 presentan decadencias.

7.3. Algoritmo de máxima verosimilitud

Debido a que el parámetro característico de la distribución estadística y el valor de la probabilidad de éxito en la adopción están relacionados únicamente con el proceso de adopción, mientras que el valor de la probabilidad de decadencia está asociado al proceso de decadencia sin estar relacionado con el de adopción, el algoritmo de máxima verosimilitud puede obtener de forma independiente por un lado los dos primeros parámetros, y por otro lado el último de los mencionados; de este modo además, se permite su representación: los dos primeros en una gráfica de 3 ejes ($p_{éxito}$, λ y $\log(p)$)como se ha mostrado en los dos capítulos anteriores, y el último en una gráfica de 2 ejes ($p_{decadencia}$ y $\log(p)$)

El algoritmo de máxima verosimilitud que calcula el valor del parámetro característico de la distribución de la probabilidad de éxito es muy parecido al presentado en el capítulo anterior (es decir, sin la combinatoria), ya que éste solo dependía de los instantes previos al primer 1 en la matriz historia (antes de haber tenido oportunidad de decaer).

La única diferencia que existe es que se eliminan aquellos intervalos en los que el límite inferior calculado según el proceso descrito en el capítulo anterior es mayor al límite superior de α, de este modo se evitan probabilidades negativas.

El algoritmo de máxima verosimilitud que calculará el valor de la probabilidad de decadencia se implementó de la siguiente manera:

1. Se recorre la matriz historia por filas. Si el valor de la última columna es un cero, pero la suma de los valores correspondientes a cada instante de simulación es mayor de cero, significa que ha ocurrido una decadencia.
    a. Si ha ocurrido esto, se recorre columna a columna la fila de historia en la que se ha dado el caso. Se cuenta el número de unos que existen.
    b. La probabilidad de la observación, en lo que a decadencia se refiere se calcula mediante:

$$(1 - p_{decadencia})^{n_{unos}-1} \cdot p_{decadencia}$$

Siendo n_unos el número de unos que existen en dicha fila. Dicho de otra manera, por cada uno que exista en la fila, significa que no ha habido decadencia (probabilidad asociada 1-$p_{decadencia}$), hasta el último, que lleva asociada $p_{decadencia}$.

2. Si no ha habido decadencia, la probabilidad de la observación asociada al agente es $(1 - p_{decadencia})^{n_{unos}-1}$

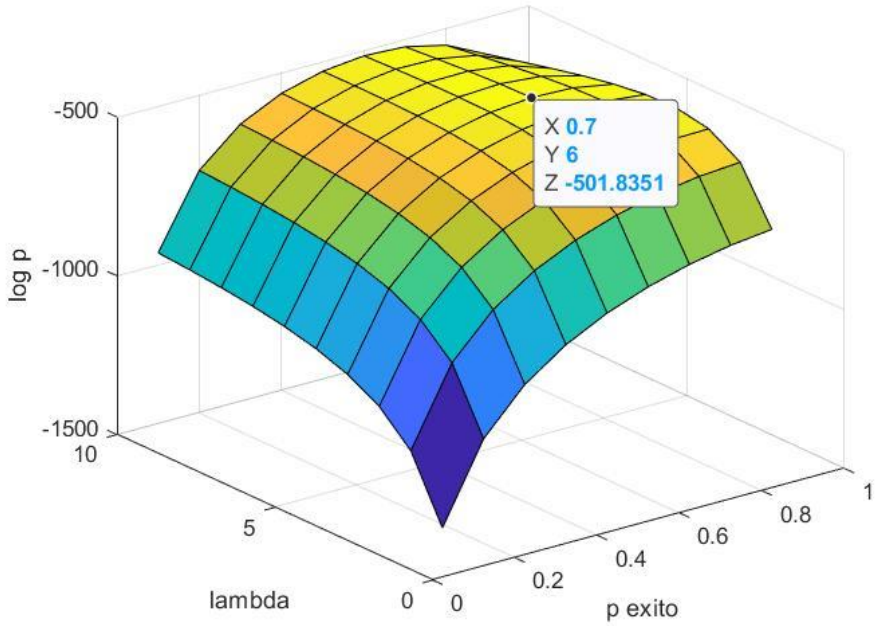
3. La probabilidad asociada a cada agente se guarda en un vector auxiliar. Una vez se han recorrido todas las filas de la matriz historia, la probabilidad de la observación completa se obtiene multiplicando todas las componentes del vector antedicho, o bien sumando el logaritmo de las componentes del mismo.

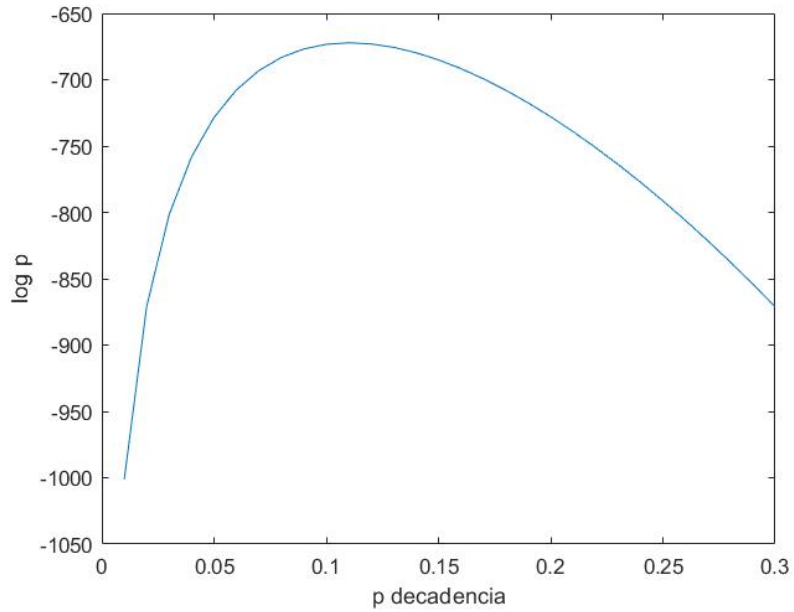
4. El proceso anterior se realiza para varios valores de $p_{decadencia}$. Los valores obtenidos se guardan en un vector para su posterior representación gráfica y cálculo del valor que maximiza la verosimilitud.

7.4. Ejemplos de ejecución del algoritmo.

En el presente apartado se muestran varios ejemplos de la ejecución del algoritmo, donde se puede apreciar tanto la gráfica con su máximo marcado, que corresponde a los valores del parámetro característico de la distribución de la probabilidad de éxito que maximizan la verosimilitud, como la gráfica que muestra el valor de la probabilidad de decadencia que maximiza la verosimilitud. Asimismo, también se muestra la solución numérica que otorga el programa de Matlab para cada uno de los parámetros del problema. Todas las gráficas que se muestran a continuación se han realizado utilizando una distribución exponencial, un número de agentes igual a 500, y un número de iteraciones igual a 10.

7.4.1 Ejecución para $\lambda = 5, p_{éxito} = 0.8, p_{decadencia} = 0.1$

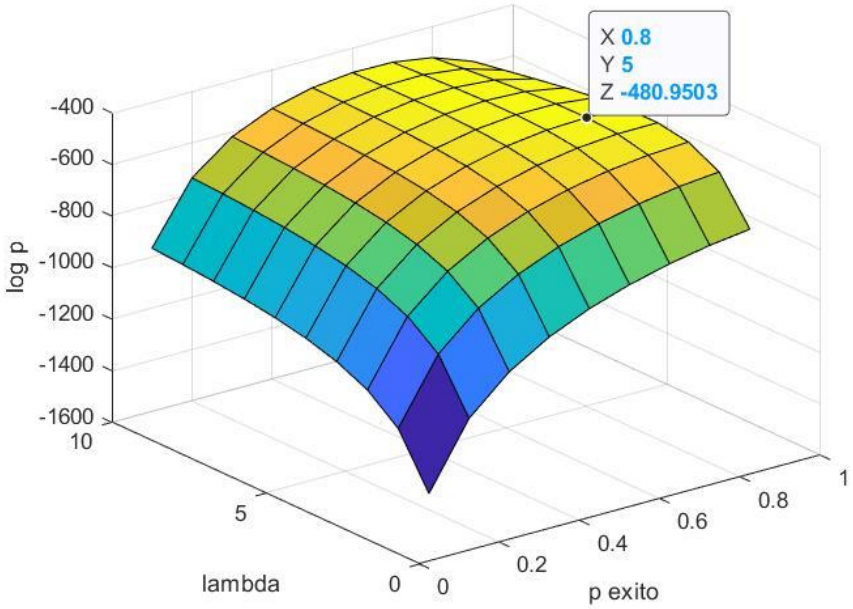


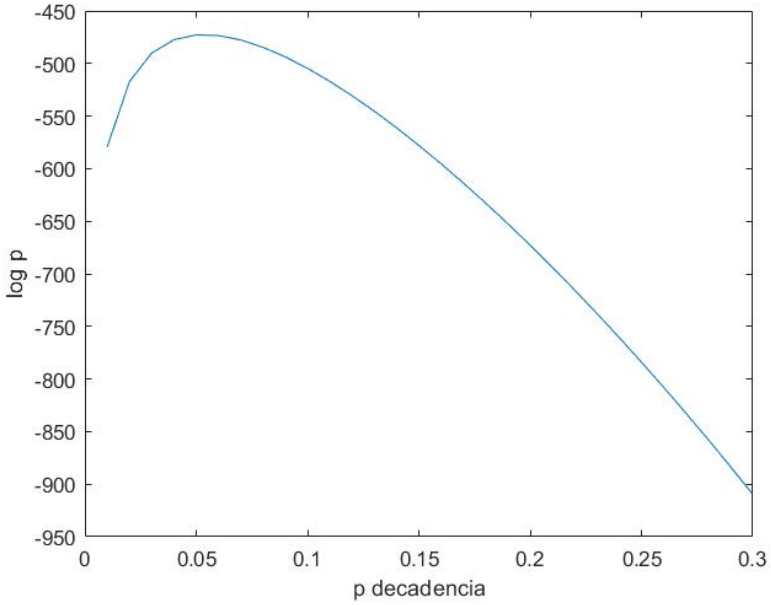



```
lambda_solution =

     6


p_exito_solution =

    0.7000


p_decaer_solution =

    0.1100
```

Figura 7.2. Ejecución para $\lambda = 5, p_{éxito} = 0.8, p_{decadencia} = 0.1$

7.4.2 Ejecución para $\lambda = 5, p_{éxito} = 0.8, p_{decadencia} = 0.05$

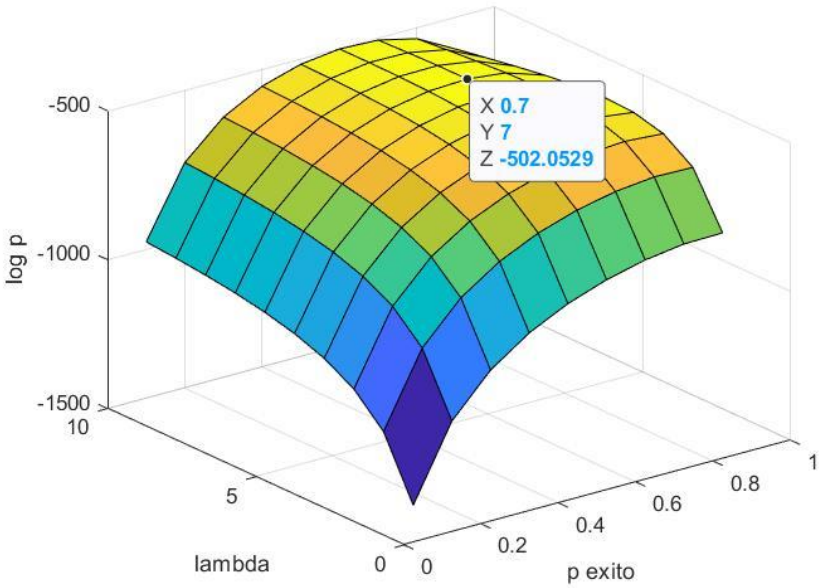


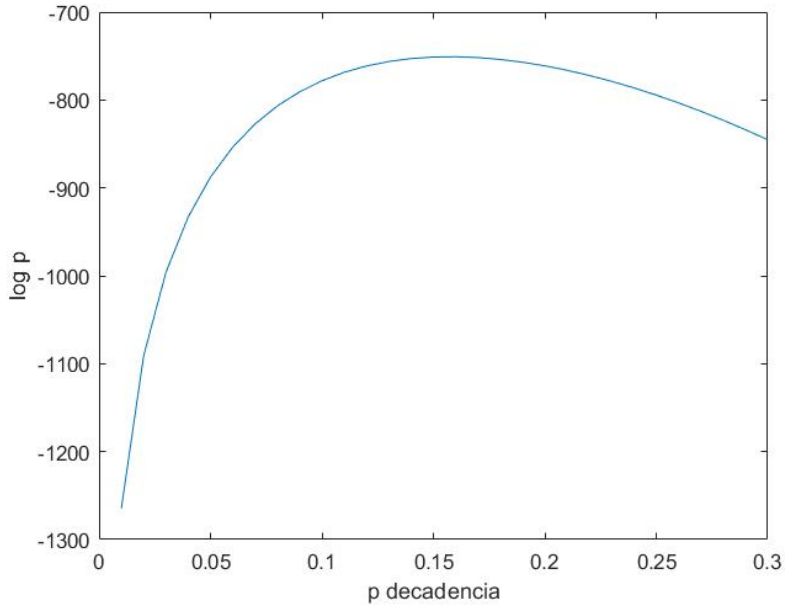



```
lambda_solution =

     5


p_exito_solution =

    0.8000


p_decaer_solution =

    0.0500
```

Figura 7.3. Ejecución para $\lambda = 5, p_{éxito} = 0.8, p_{decadencia} = 0.05$

7.4.3 Ejecución para $\lambda = 7, p_{éxito} = 0.7, p_{decadencia} = 0.15$

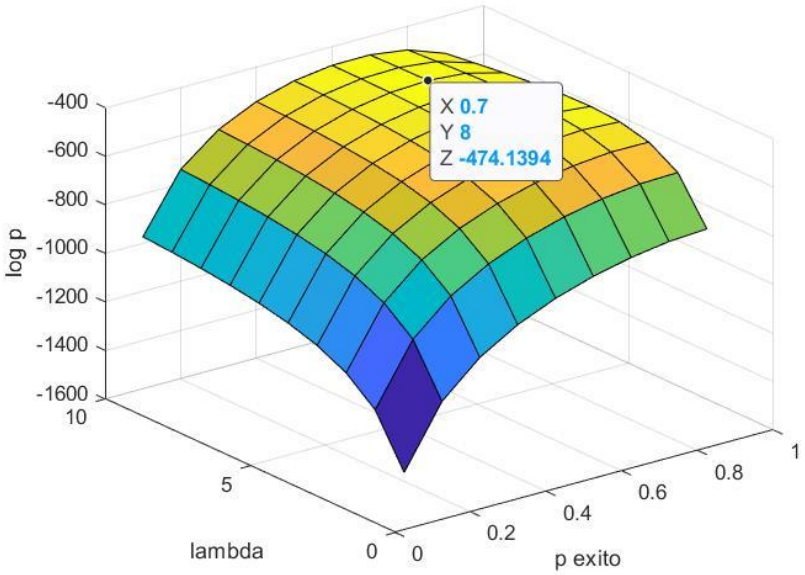


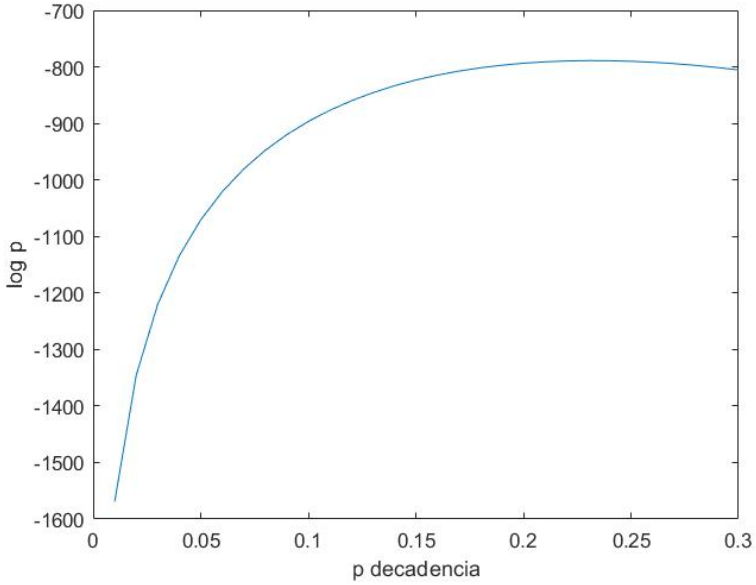



```
lambda_solution =

     7


p_exito_solution =

    0.7000


p_decaer_solution =

    0.1600
```

Figura 7.4. Ejecución para $\lambda = 7, p_{éxito} = 0.7, p_{decadencia} = 0.15$

7.4.4 Ejecución para $\lambda = 7, p_{éxito} = 0.7, p_{decadencia} = 0.2$

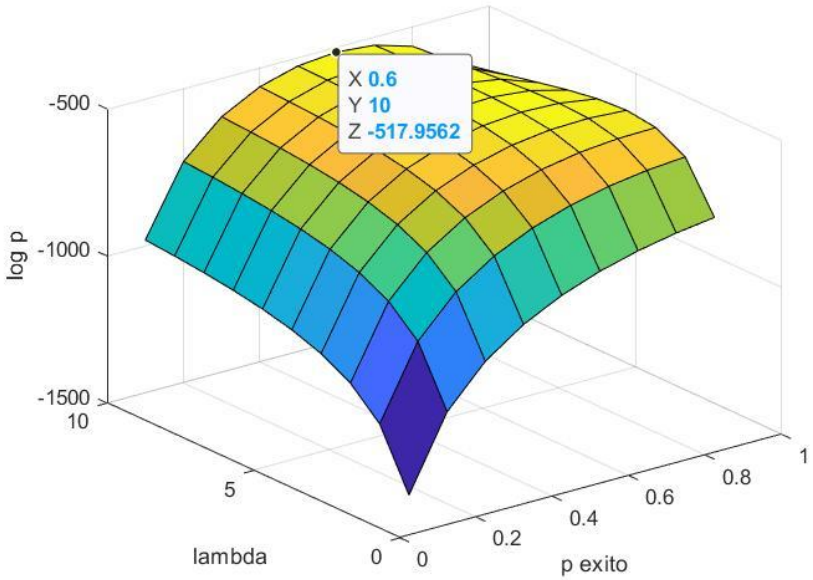


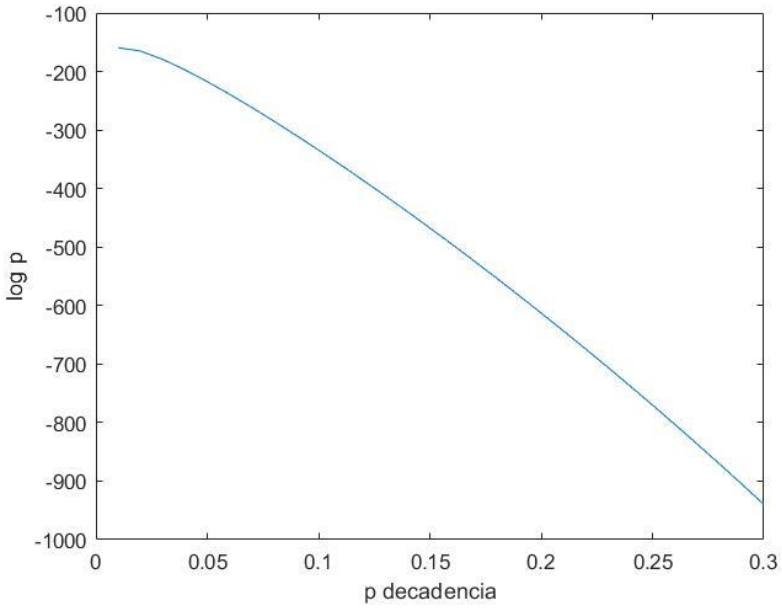



```
lambda_solution =

     8


p_exito_solution =

    0.7000


p_decaer_solution =

    0.2300
```

Figura 7.5. Ejecución para $\lambda = 7, p_{éxito} = 0.7, p_{decadencia} = 0.2$

7.4.5 Ejecución para $\lambda = 9, p_{éxito} = 0.6, p_{decadencia} = 0.01$

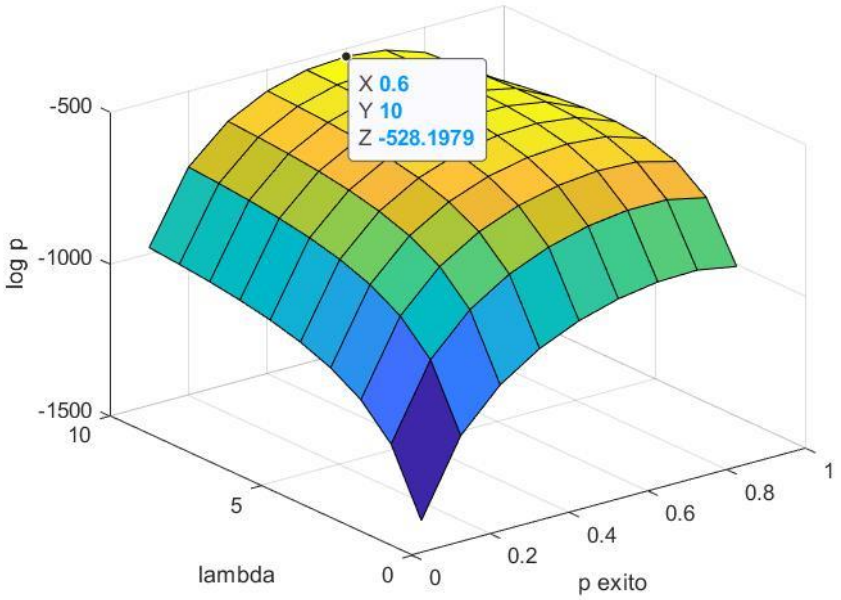


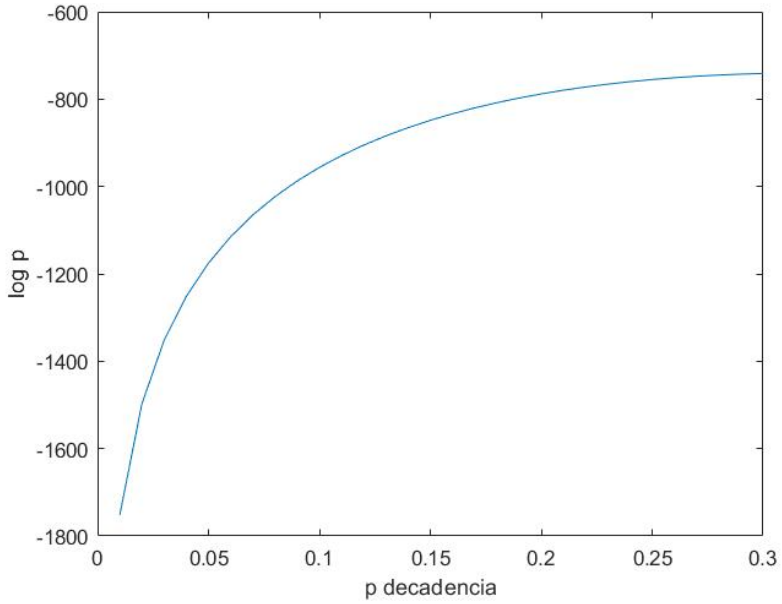



```
lambda_solution =

    10


p_exito_solution =

    0.6000


p_decaer_solution =

    0.0100
```

Figura 7.6. Ejecución para $\lambda = 9$ $p_{éxito} = 0.6, p_{decadencia} = 0.01$

7.4.6 Ejecución para $\boldsymbol{\lambda = 9, p_{éxito} = 0.6, p_{decadencia} = 0.3}$






```
lambda_solution =

    10


p_exito_solution =

    0.6000


p_decaer_solution =

    0.3000
```

Figura 7.7. Ejecución para $\lambda = 9\ p_{éxito} = 0.6, p_{decadencia} = 0.3$

A lo largo de este capítulo se ha explicado la implementación de la nueva variable (probabilidad de decadencia), la justificación de por qué dicha variable es interesante de cara a mejorar el realismo del modelo, y el proceso que se siguió para su programación en Matlab. También se explicaron las modificaciones necesarias al algoritmo de máxima verosimilitud para poder estimar el valor del nuevo parámetro. Como en los anteriores capítulos, una vez explicado el concepto de la nueva variable y su implementación en el modelo, se realizaron varias simulaciones del algoritmo que ponen de manifiesto su correcto funcionamiento. Al igual que ocurrió en capítulos anteriores, el algoritmo de máxima verosimilitud proporciona resultados en ocasiones exactos, y cuando no son exactos, los valores proporcionados son bastante cercanos.

Con este capítulo quedan finalizadas las modificaciones del modelo y las implementaciones de algoritmos estadísticos para la estimación de parámetros en este trabajo; no obstante, en el siguiente capítulo se describen diversas mejoras futuras que se pueden realizar a lo desarrollado en este trabajo, así como una recopilación de las conclusiones más importantes.

8 CONCLUSIONES Y MEJORAS FUTURAS

Durante el transcurso de este trabajo, realizado en parte durante la estancia del autor en Politecnico di Torino y con la importante colaboración de Fabrizio Dabbene y Chiara Ravazzi, se ha implementado un modelo para estudiar comportamientos sociales. A pesar de que el modelo era sencillo y funcionaba bien, presentaba el problema de que consideraba el mismo valor del parámetro α para cada uno de los agentes de la red. Esto provocaba un doble problema; por un lado, es poco realista asumir que todos los individuos de una comunidad son igual de sensibles a la influencia ajena. Por otro lado, como se vio en el capítulo 3, para generar dinámicas que fueran considerablemente diferentes entre ellas, debían elegirse valores de α entre 0.3 y 0.5, ya que valores menores siempre resultaban en adopciones muy rápidas en las que incluso se llegaba a la adopción por parte de todos los agentes tras 2 o 3 iteraciones, y valores mayores resultaban en dinámicas en las que prácticamente ningún agente se convertía en adoptante.

El hecho de haber cambiado el valor estático de α e igual para todos los agentes por una distribución estadística, permite no solo un mayor realismo al considerar que puedan existir individuos más susceptibles que otros al cambio, sino que permite un juego mucho mayor con el uso de los parámetros característicos de la distribución resultando en dinámicas mucho más variadas.

Adicionalmente, las diversas adiciones de variables extra en el modelo de red aumentan aún más el realismo del modelo, al poder considerar el hecho de que factores externos retrasen la adopción aún teniendo un agente influencia ajena suficiente, o el hecho de que un agente pueda abandonar su estado de adoptante y dejar así de influir en sus vecinos.

En cada modificación del modelo de red en que se basa este trabajo, se han incluido algoritmos de máxima verosimilitud para resolver problemas inversos, esto es, en lugar de elegir ciertos parámetros y generar una dinámica, hallar los parámetros que, con mayor probabilidad, otorgarían una cierta dinámica dada. Se ha mostrado que el funcionamiento de dichos algoritmos es correcto, y que, a pesar de ser la máxima verosimilitud un método que no siempre es exacto, en muchas ocasiones el valor de los parámetros era exactamente el correcto; en los casos en los que no lo era, el error era bastante pequeño.

La comparativa realizada en el capítulo 6 sobre la mejora en la velocidad de ejecución del algoritmo al eliminar la combinatoria utilizada en el algoritmo del capítulo 5, pone de manifiesto las grandes ganancias en términos de tiempo de ejecución, y permite justificar la decisión de utilizar el algoritmo mejorado en el capítulo posterior.

8.1. Mejoras futuras

En trabajos futuros pueden implementarse las siguientes mejoras del modelo de red y/o los algoritmos estadísticos empleados en este trabajo:

- Incluir búsquedas locales para reducir el tiempo de ejecución. Es una opción interesante, en lugar de realizar un barrido de los parámetros y después hallar la probabilidad máxima, hacer ese mismo barrido, pero en torno a un determinado valor que se sabe que posee una probabilidad más alta que su entorno, de este modo, se ahorran iteraciones y por consiguiente tiempo de ejecución.

- Considerar redes cuyo número de agentes sea variable. Las redes empleadas en este trabajo poseen durante toda la simulación un número constante de agentes; una posible mejora futura sería programar redes en las que agentes puedan entrar y/o salir de ellas, teniendo influencias temporales sobre el resto de vecinos.

- Considerar redes en las que los agentes varíen su posición. En este trabajo, cada agente tenía asignada una posición fija, de forma análoga a las casas de una ciudad. Una modificación que abriría las puertas a la resolución de nuevos problemas, es considerar redes en los que los agentes se muevan, de este modo, cambiarían de vecinos con el tiempo pudiendo influir a otros agentes y ser influidos por nuevos vecinos.

- Implementar valores variables de α, de este modo se simularía un cambio en la susceptibilidad con el tiempo.

9 BIBLIOGRAFÍA

[1] V. Breschi, M. Tanelli, C. Ravazzi, S. Strada and F. Dabbene, "Social network analysis of electric vehicles adoption: a data-based approach," 2020 IEEE International Conference on Human-Machine Systems (ICHMS), Rome, Italy, 2020, pp. 1-4

[2] International Energy Agency, Global EV Outlook 2019 - Scaling-up the transition to electric mobility, 2019.

[3] McKinsey & Company, Electrifying insights: How automakers can drive electrified vehicle sales and profitability, 2017.

[4] Electric vehicles. (2020). Retrieved 26 October 2020, from https://www2.deloitte.com/uk/en/insights/focus/future-of-mobility/electric-vehicle-trends-2030.html

[5] Ministerio de Sanidad, Consumo y Bienestar Social. 2020.

[6] Hald, Anders. On the history of maximum likelihood in relation to inverse probability and least squares. Statist. Sci. 14 (1999), no. 2, 214--222.

[7] Maximum likelihood estimation. (2020). Retrieved 26 October 2020, from https://en.wikipedia.org/wiki/Maximum_likelihood_estimation#History

[8] Brooks-Bartlett, J. (2019, 10 diciembre). Probability concepts explained: Maximum likelihood estimation. Medium. https://towardsdatascience.com/probability-concepts-a explained-maximum-likelihood-estimation-c7b4342fdbb1

[9] Wikipedia contributors. (2020, 21 octubre). Exponential distribution. Wikipedia. https://en.wikipedia.org/wiki/Exponential_distribution

10 ANEXOS

A continuación, se adjuntan los códigos del modelo implementado y los algoritmos de máxima verosimilitud, en el siguiente orden:

- Programa que genera la red de agentes
- Programa que ejecuta la dinámica de influencia en función de los parámetros dados
- Programa que contiene el algoritmo de máxima verosimilitud para hallar los parámetros.

Programa que genera la red de agentes

```
clear;
clc;
close all;

n=100;

%GENERA AGENTES

long_v=10+(10.5-10).*rand(n,1);
lat_v=44.5+(45-44.5).*rand(n,1);
dist=100+(400-100).*rand(n,1);

h_v=[long_v,lat_v,dist];
long_mu=mean(h_v(:,1));
lat_mu=mean(h_v(:,2));

%REPRESENTA AGENTES GENERADOS

figure
scatter(h_v(:,1),h_v(:,2),'filled');
xlabel('longitud');
ylabel('latitud');
title('Agentes generados');


%CALCULA DISTANCIA ENTRE AGENTES

distance_matrix=zeros(n);
distance_to_avg=zeros(n,1);

for i=1:n
    for j=1:n
    distance_matrix(i,j)=deg2km(distance(h_v(i,1:2),h_v(j,1:2)));
    end
    distance_to_avg(i)=deg2km(distance(h_v(i,1:2),[long_mu,lat_mu]));
end

sigma_h=std(distance_to_avg);

%PROCESO DE ELIMINACION DE AGENTES

prune_1=distance_matrix<60;
prune_2=distance_matrix<5;
elimina=[];
sum=0;


for i=1:n
    for j=1:n
        sum=sum+prune_2(i,j);
        if prune_1(i,j)==0
            elimina=[elimina,i];
        end
    end
```

```
     if sum==1
        elimina=[elimina,i];
     end
    sum=0;
end

h_v(elimina(:),:)=[];
n=length(h_v);


%REPRESENTA AGENTES TRAS EL PROCESADO DE RED

figure
scatter(h_v(:,1),h_v(:,2),'filled');
xlabel('longitud');
ylabel('latitud');
title('Agentes generados');

%CALCULA VECINOS DE CADA AGENTE

total_edges=0;
neighbors=0;
elimina=[];
for i=1:n
    for j=1:n
        if deg2km(distance(h_v(i,1:2),h_v(j,1:2))) <= 5
            neighbors=neighbors+1;
        end
    end
    if neighbors < 4
        elimina=[elimina,1];
    end
    neighbors=0;
end

h_v(elimina(:),:)=[];
n=length(h_v);

figure
xlim([10,10.5]);
ylim([44.5,45]);
scatter(h_v(:,1),h_v(:,2),'filled','r')
hold on
xlabel('longitud');
ylabel('latitud');
title('Red post-tratamiento');

total_edges=0;
neighbors=[];
k=1;

for i=1:n
    for j=1:n
        if deg2km(distance(h_v(i,1:2),h_v(j,1:2))) <= 5
            if i~=j
            neighbors=[neighbors;i,j];
            k=k+1;
```

```
            end
        end
    end
end

A=h_v;
```

Programa que ejecuta la dinámica de influencia en función de los parámetros dados

```
clc
figure
counter=0;
number_of_iterations=10;
historia=zeros(n,number_of_iterations+2);
historia(:,1)=1:n;
n=length(h_v);
h_v=A;


%GENERA VECTOR DE ALPHA

adopter=zeros(n,1);
lambda=9;
umbral_exito=0.6;
p_decaer=0.3;

f_alpha=@generate_alpha_vector;

alpha=f_alpha(n);


h_v=[h_v,alpha,adopter];
S_0=[];
decaidos=[];


%CALCULA ADOPTANTES INICIALES

for i=1:n
    if h_v(i,3)<130
        scatter(h_v(i,1),h_v(i,2),'filled','g')
        counter=counter+1;
        h_v(i,5)=1;
        S_0=[S_0,i];
        hold on
        historia(i,2:number_of_iterations+2)=1;
    else
        scatter(h_v(i,1),h_v(i,2),'filled','r')
        h_v(i,5)=0;
        hold on
    end
end

percent_s0=100*counter/n;

disp('percentage of vehicles belonging to S_0');
disp(percent_s0);

hold on

m=length(neighbors);
```

```
S_t=S_0;
current=1;
adopting_neighbors=0;
total_neighbors=0;
coefficient=0;
t=0;

adopters=[length(S_t)];

%DINÁMICA DE ADOPCIÓN

while(t<number_of_iterations)

aux=[];

%ASIGNA VARIABLE ALEATORIA PARA ÉXITO EN LA ADOPCIÓN

for i=1:n
    x=rand;
    if x<umbral_exito
      aux=[aux;1];
    else
      aux=[aux;0];
    end
end

h_v=[h_v,aux];

%CALCULA NUEVOS ADOPTANTES

for i=1:m
    if i>1
        if neighbors(i,1)~=neighbors(i-1,1) || i==m
            coefficient=adopting_neighbors/total_neighbors;
            if coefficient>=h_v(neighbors(i-1,1),4) && ...
            h_v(neighbors(i-1,1),t+6)==1 && ...
            ismember(neighbors(i-1,1),decaidos)==0
                S_t=sort([S_t,neighbors(i-1,1)]);
            end
            adopting_neighbors=0;
            total_neighbors=0;
        end
    end

    if ismember(neighbors(i,1),S_0)==0
        if ismember(neighbors(i,2),S_0)==1
            adopting_neighbors=adopting_neighbors+1;
        end
        total_neighbors=total_neighbors+1;
    end
end

[a,b]=size(historia);


%ACTUALIZA MATRIZ HISTORIA CON LOS NUEVOS ADOPTANTES
```

```
S_0=unique(S_t);
historia(S_0,t+3:number_of_iterations+2)=1;


%GENERA AGENTES QUE DECAEN ALEATORIAMENTE Y MODIFICA MATRIZ HISTORIA

if t>1
    for i=1:a
        if historia(i,t)==1
            x=rand;

            if x<p_decaer
                historia(i,t+1:b)=-1;
                decaidos=[decaidos;i];

                S_t(S_t==i)=[];
            end
        end
    end
end


h_v(S_0,5)=1;


adopters=[adopters,length(S_t)];

t=t+1;
current=1;

end

historia(historia==-1)=0;

B=h_v;
```

Programa que contiene el algoritmo de máxima verosimilitud para hallar los parámetros.

```
clc;
clear sum;
close all;

%CALCULA MATRIZ PARA CONOCER LÍMITES DE INTERGACIÓN

[adopting_neighbors_matrix,total_neighbors_matrix] = ...
calculate_neighbors(historia,neighbors);
[a,b]=size(historia);
intervals_matrix=adopting_neighbors_matrix./total_neighbors_matrix;
intervals_matrix(:,1)=1:a;
f=@PDF_Exponential;
p_alpha=zeros(10);
lambda_index=1;

%REALIZA BARRIDO DE PARÁMETROS LAMBDA Y P_EXITO

for lambda=1:10
    p_exito_index=1;
    for p_exito=0.1:0.1:1
        p_fracaso=1-p_exito;
        p_alpha_vector=zeros(a,1);

        %RECORRE MATRIZ HISTORIA CALCULANDO PROBABILIDAD DE CADA AGENTE

        for i=1:a
            for j=b:-1:2

                %AGENTES QUE NUNCA ADOPTAN

                if historia(i,j)==0 && j==b && sum(historia(i,2:b))==0
                    intervals=zeros(j-1,2);
                    for k=1:j-1
                        if k==1
                            intervals(k,1)=0;
                            intervals(k,2)=intervals_matrix(i,2);
                        end
                        if k>1
                        intervals(k,1)=intervals_matrix(i,k);
                        intervals(k,2)=intervals_matrix(i,k+1);
                        end
                    end
                        [c,d]=size(intervals);
                    aux=zeros(c,1);
                    for l=1:c
                        aux(l)=integral(@(alpha) f(lambda,alpha), ...
                        intervals(l,1), intervals(l,2))*p_fracaso^(c-l+1);
                    end

                    %CALCULA PROBABILIDAD DE OBSERVACIÓN DEL AGENTE

                    p_alpha_vector(i)=sum(aux)+integral(@(alpha)...
```

```
                    f(lambda,alpha), 0, ...
                    intervals_matrix(i,b))*p_fracaso^(b-2)+...
                    integral(@(alpha) f(lambda,alpha),...
                    intervals_matrix(i,b), Inf);
                    break;
                end

                % AGENTES ADOPTANTES INICIALES

                if historia(i,2)==1
                    p_alpha_vector(i)=-1;
                    break;
                end

                %AGENTES QUE ADOPTAN DURANTE LA DINÁMICA

                if j<b && historia(i,j+1)==1 && historia(i,j)==0
                    intervals=zeros(j-1,2);
                    for k=1:j-1
                        if k==1
                            intervals(k,1)=0;
                            intervals(k,2)=intervals_matrix(i,2);
                        end
                        if k>1
                        intervals(k,1)=intervals_matrix(i,k);
                        intervals(k,2)=intervals_matrix(i,k+1);
                        end
                    end


                        [c,d]=size(intervals);
                        for k=1:c
                            if intervals(k,1)>intervals(k,2)
                                intervals(k,:)=0;
                            end
                        end

                    aux=zeros(c,1);
                    for l=1:c
                        aux(l)=integral(@(alpha) f(lambda,alpha), ...
                        intervals(l,1), ...
                        intervals(l,2))*p_exito*p_fracaso^(c-l);
                    end
                    p_alpha_vector(i)=sum(aux);
                    break;
                end

            end
        end

        %GUARDA LA PROBABILIDAD DE OBSERVACIÓN EN LA MATRIZ CORRESPONDIENTE

        p_alpha_vector(p_alpha_vector==-1)=[];
        p_alpha(lambda_index,p_exito_index)=sum(log10(p_alpha_vector))
        p_exito_index=p_exito_index+1;
    end
    lambda_index=lambda_index+1;
```

```
end

%CÁLCULA EL MÁXIMO DE LA MATRIZ DE PROBABILIDAD

maximum=max(max(p_alpha));

[I,J]=find(p_alpha==maximum);

%IMPRIME POR PANTALLA LA SOLUCIÓN NUMÉRICA

lambda_solution=I
p_exito_solution=J*0.1

%REPRESENTA LA GRÁFICA DE MÁXIMA VEROSIMILITUD

figure;
p_exito_v=0.1:0.1:1;
lambda_v=1:1:10;
[X,Y]=meshgrid(p_exito_v,lambda_v);
surf(X,Y,p_alpha)
xlabel('p exito');
ylabel('lambda');
zlabel('log p');


[a,b]=size(historia);

p_decaer_vector=zeros(30,1);

%ALGORITMO PARA CALCULAR PROBABILIDAD DE DECADENCIA

index=1;

%REALIZA UN BARRIDO DEL PARÁMETRO P_DECAER

for p_decaer=0.01:0.01:0.3
    aux=[];
    for i=1:a

        %AGENTES QUE HAN DECAÍDO EN ALGÚN MOMENTO

        if historia(i,b)==0 && sum(historia(i,2:b))>0 && ismember(i,decaidos)==1
            count=0;
            flag=0;
            for j=2:b-1
                if historia(i,j)==0 && historia(i,j+1)==1
                    flag=1;
                end

                if historia(i,j)==1 && flag==1
                    count=count+1;
                end

            end
            aux=[aux;(1-p_decaer)^(count-1)*p_decaer];

        end
```

```
        %AGENTES QUE NO HAN DECAÍDO

        if historia(i,b)==1 && sum(historia(i,2:b))>1
            count=0;
            flag=0;
            for j=2:b-1
                if historia(i,j)==0 && historia(i,j+1)==1
                    flag=1;
                end

                if historia(i,j)==1 && flag==1
                    count=count+1;
                end

            end
            aux=[aux;(1-p_decaer)^(count-1)];

        end

    end

    %GUARDA LA PROBABILIDAD DE OBSERVACIÓN EN UN VECTOR

    p_decaer_vector(index)=sum(log10(aux));
    index=index+1;
end

%CALCULA EL MÁXIMO DEL VECTOR DE PROBABILIDAD

maximum=max(p_decaer_vector);

%IMPRIME POR PANTALLA LA SOLUCIÓN NUMÉRICA

[I]=find(p_decaer_vector==maximum);

p_decaer_solution=I/100

%REPRESENTA LA GRÁFICA DE MÁXIMA VEROSIMILITUD

figure
plot([0.01:0.01:0.3], p_decaer_vector);
xlabel('p decadencia');
ylabel('log p');
```