PRÁCTICA 1 – APRENDIZAJE AUTOMÁTICO

GRADIENTE DESCENDENTE Y REGRESIÓN LINEAL

Antonio José Blánquez Pérez
45926869D
3ºCSI

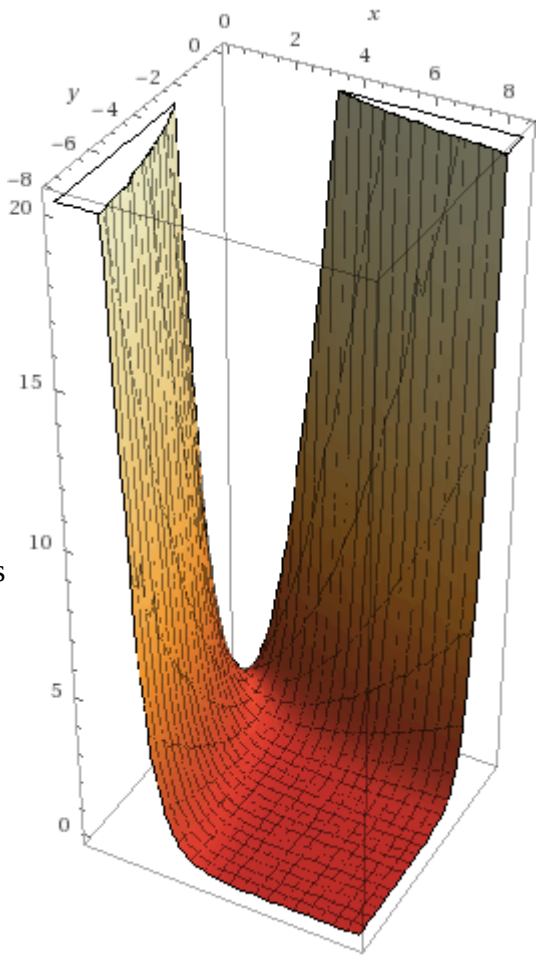
1 GRADIENTE DESCENDENTE

En este ejercicio se pide implementar el gradiente descendente para poder aplicarlo minimizando diferentes funciones. Se ha implementado el código con la ayuda del esqueleto que se ha proporcionado, basándose en el pseudocódigo explicado en teoría y consultando dudas concretas sobre Python en las páginas web que se listan en el apartado de bibliografía

1.1 MINIMIZACIÓN DE LA FUNCIÓN E

Primero, se ha planteado minimizar, mediante gradiente descendente, la siguiente función:

$$E(u,v)=(ue^{v}-2ve^{-u})^{2}$$



Dicha función tiene la forma que podemos ver en la imagen de la derecha(extraída con ayuda de la página web Wolfram Alpha), de la que se puede deducir que es al menos aparentemente convexa y que por lo tanto el gradiente descendente encontrará un buen óptimo local o incluso un óptimo global. Aunque no sirva de nada estudiar la función, ya que en casos reales la mayoría de veces no se va a poder ni representar ni intuir como esta se va a comportar, me ha parecido interesante para este ejercicio para entender mejor como funciona la técnica del descenso del gradiente

El gradiente de la función será el vector formado por sus dos derivadas parciales, las cuales también han sido calculadas por con ayuda de Wolfram Alpha y son las siguientes:

$$\frac{\partial}{\partial u}((ue^{v}-2e^{-u}v)^{2})=2e^{-2u}(ue^{u+v}-2v)(e^{u+v}+2v)$$

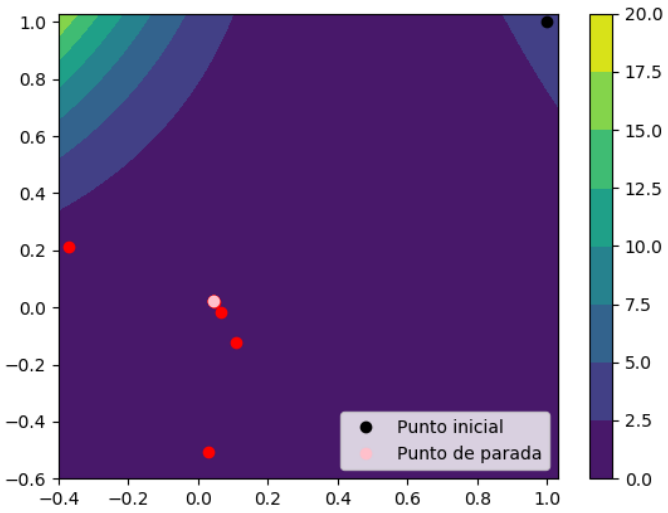
$$\frac{\partial}{\partial v}((ue^{v}-2e^{-u}v)^{2})=2e^{-2u}(ue^{u+v}-2)(ue^{u+v}+2v)$$

La cuestión que se nos plantea es cuántas iteraciones tarda el algoritmo en obtener por primera vez un valor menor a 10^{-14} y en que coordenadas lo hace. Para obtener fácilmente una respuesta a ésta pregunta se ha establecido como criterio de parada para el gradiente descendente que el valor de la función en el punto donde acaba la iteración sea menor a 10^{-14}, de tal manera que obtenemos los siguientes resultados tras la ejecución del primer apartado:

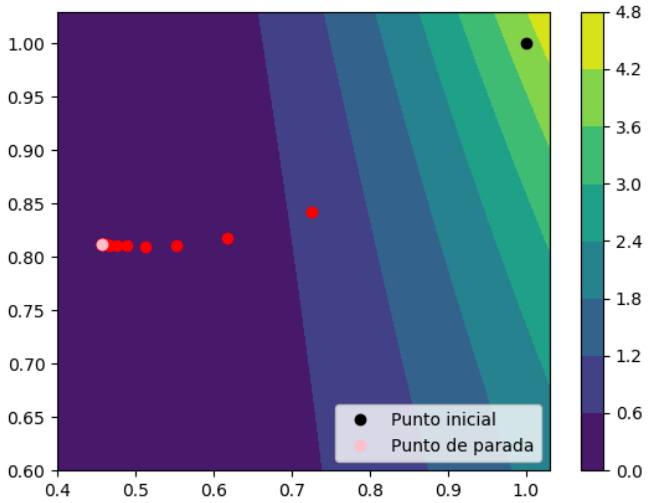
Número de iteraciones: 10

Coordenadas obtenidas: (0.04473629039778203, 0.02395871409914173)

Sin embargo, hay un pequeño detalle que quiero comentar. Con el objetivo de mostrar como funciona el gradiente descendente, me propuse representar su funcionamiento de la mejor manera posible, para lo cual, tras buscar e investigar, decidí hacerlo mediante una gráfica de contorno. En el guión se establece que el ratio de aprendizaje es de 0.1, sin embargo se puede comprobar mediante la gráfica como éste learning rate es demasiado alto, ya que va oscilando alrededor de la zona de mínimo absoluto.



Aunque en el código se ha dejado así, ya que se pide explícitamente un ratio de aprendizaje de 0.1, adjunto también la misma gráfica con un learning rate adecuado para poder ver el camino sin que se produzcan oscilaciones, donde sí que se puede percibir de manera muy visual el comportamiento correcto del gradiente descendente.

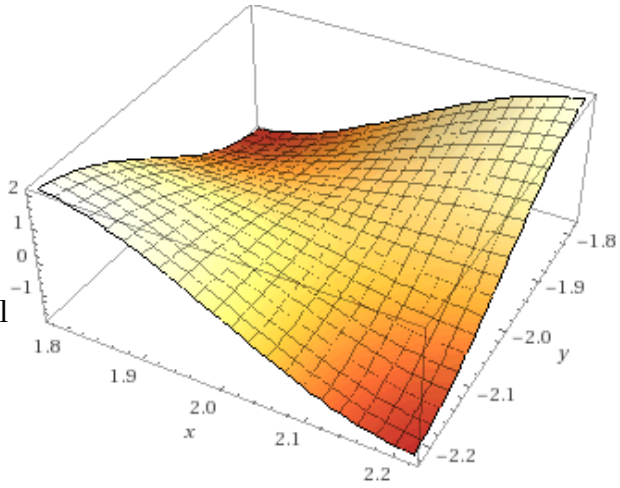


1.2 MINIMIZACIÓN DE LA FUNCIÓN F

Durante este apartado se considerará la siguiente función:

$$f(x,y)=(x-2)^2+2(y+2)^2+2\sin(2\pi x)\sin(2\pi y)$$

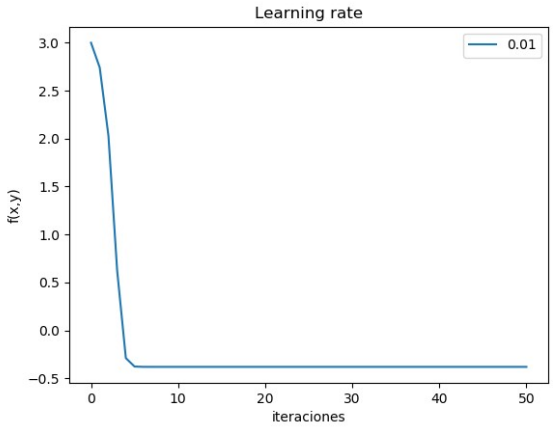
También se han obtenido la forma, la cual se puede ver en la imagen de la derecha, y las derivadas de la función con ayuda de Wolfram Alpha. De la forma y la ecuación se puede deducir que en esta función sí que va a haber óptimos locales y muchos, ya que depende de funciones trigonométricas. Su gradiente está formado, al igual que en la anterior función, por sus dos derivadas parciales que son las siguientes:

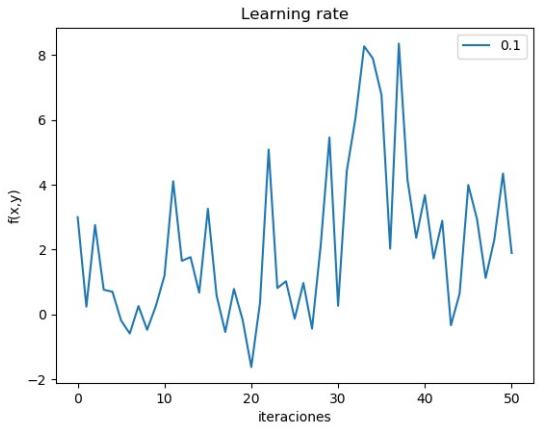


$$\frac{\partial}{\partial x}\left(2\sin(2\pi x)\sin(2\pi y)+(x-2)^2+2(y+2)^2\right)=2\left(2\pi\cos(2\pi x)\sin(2\pi y)+x-2\right)$$

$$\frac{\partial}{\partial y}\left(2\sin(2\pi x)\sin(2\pi y)+(x-2)^2+2(y+2)^2\right)=4\left(\pi\sin(2\pi x)\cos(2\pi y)+y+2\right)$$

Empezando por el estudio del learning rate, veamos las gráficas obtenidas con los valores 0.1 y 0.01 y pasemos a comentar los resultados:





A través de éstas gráficas se puede apreciar perfectamente la diferencia entre usar un ratio de aprendizaje adecuado y usar uno no adecuado. Cuando usamos 0.01 va directamente hacia el óptimo cercano más profundo y se queda con él, sin embargo, cuando usamos 0.1, que para ésta función es demasiado alto, podemos ver cómo fluctúa. Estas fluctuaciones son debidas a que, encontrado el gradiente y estando en dirección al óptimo objetivo, el learning rate es lo suficientemente alto como para que el paso sea más largo que la distancia hasta ese óptimo, de manera que nos pasamos, entonces el gradiente vuelve a apuntar en la dirección de un óptimo y sigue ese proceso en bucle, de manera que nunca llegará a un mínimo.

De esto podemos comprobar que elegir un learning rate es una decisión delicada; con uno muy bajo el gradiente descendente necesitará un número de iteraciones muy alto que conllevará una gran cantidad de tiempo, problema que con un valor más alto podemos subsanar. Sin embargo, si el ratio de aprendizaje es demasiado alto los valores empezarán a oscilar alrededor del mínimo local y perderemos ese tiempo que buscábamos ahorrarnos oscilando alrededor de la solución.

Por otra parte pasemos a estudiar la tabla de valores para los distintos puntos iniciales planteados:

Coordenadas de inicio	Coordenadas de parada	Valor obtenido
(2.1, -2.1)	(2.08, -1.24)	0.1666
(3, -3)	(1.1, -2.29)	-0.1884
(1.5, 1.5)	(1.2, -1.67)	2.5405
(1, -1)	(1.09, -2.56)	1.8937

A través de ésta tabla podemos ver como el resultado de aplicar el gradiente descendente es muy dependiente de la posición inicial, ya que variando ésta entre valores no demasiado alejados conseguimos resultados muy diferentes entre sí, tanto en coordenadas como en mínimos.

1.3 CONCLUSIONES

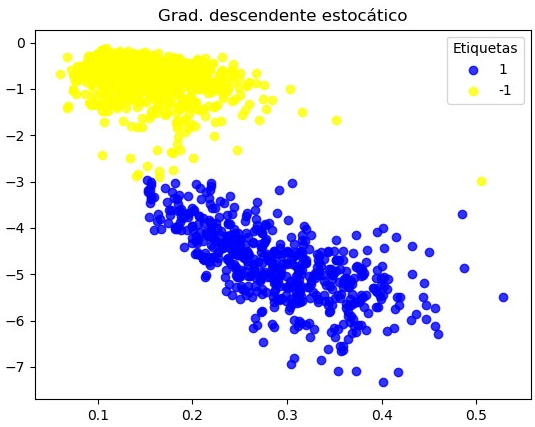
Teniendo en cuenta todo lo estudiado anteriormente, podría concluir que la verdadera dificultad de encontrar el mínimo global de una función arbitraria es que es muy difícil saber en qué parte de la función estamos, por lo que es muy difícil escapar de óptimos locales en las funciones no convexas, que son la gran mayoría. Así, todas las mejoras y alternativas a este algoritmo no van tan enfocadas a llegar a un óptimo ya que ya se resuelve de manera eficiente(aún con el problema de tener que elegir el learning rate), si no que van enfocadas a evitar óptimos locales ya que este es el caso crítico y difícil de resolver. Prueba de ello es la tabla del apartado anterior, donde se puede ver claramente esa dificultad, ya que desde distintos puntos iniciales obtenemos óptimos distintos y ni siquiera sabemos si alguno de ellos es el óptimo global. Es por esto por lo que podemos afirmar que la verdadera dificultad de encontrar un óptimo global es escapar de los óptimos locales.

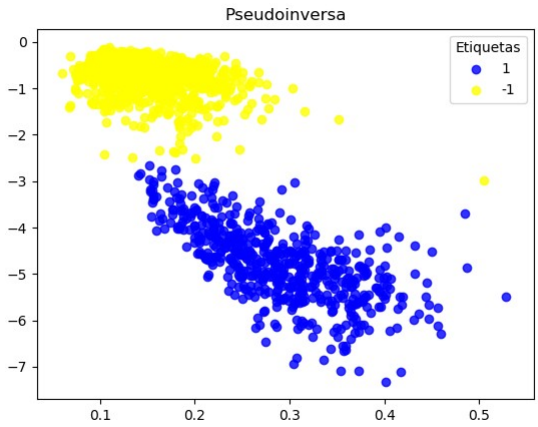
2 REGRESIÓN LINEAL

2.1 MODELO DE REGRESIÓN LINEAL PARA DÍGITOS MANUSCRITOS

Para este ejercicio tenemos un conjunto de datos que sabemos que son dígitos manuscritos, en este caso unos y cincos, etiquetas -1 y 1 respectivamente, y que se han extraído dos características de ellos: la intensidad promedio y la simetría. Para intentar clasificarlos se va a estimar un modelo de regresión lineal utilizando tanto el gradiente descendente estocástico(en adelante SGD) como el de la pseudoinversa.

Veamos a continuación la clasificación resultado de ambas técnicas y la bondad de esas soluciones:





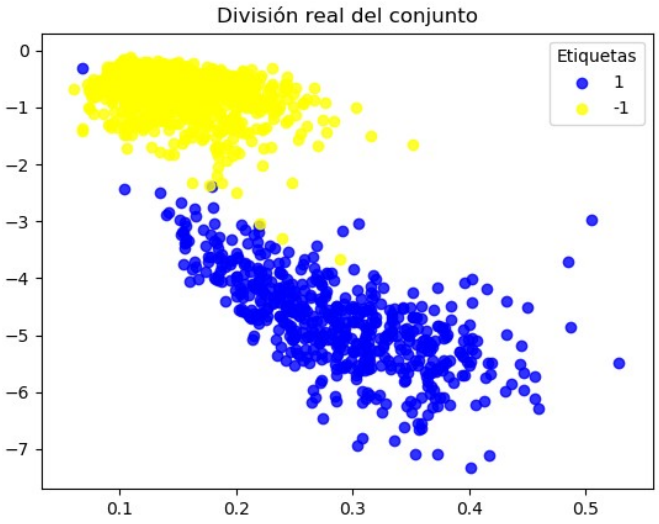
E_{in}: 0.0865

E_{out}: 0.1332

E_{in}: 0.0791

E_{out}: 0.1309

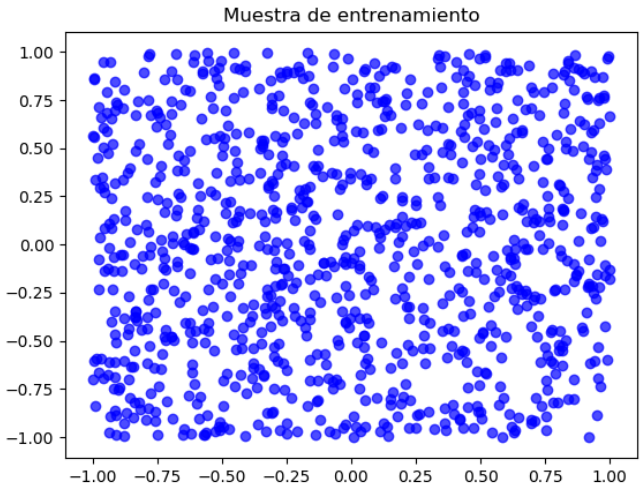
Primero aclarar que los resultados de SGD varían algunas centésimas en el error y algunas asignaciones en la clasificación en cada ejecución debido a su naturaleza aleatoria(como su propio nombre indica). Dicho esto podemos ver que ambos resultados son parecidos, aunque ningún resultado de SGD llega a ser tan bueno como el que proporciona el algoritmo de la pseudoinversa; aun así ambos resultados son buenos, teniendo un Ein bastante bueno y un Eout muy aceptable. Por ello podemos decir que con ambas técnicas hemos obtenido una buena solución, de hecho ambas gráficas mantienen pocas diferencias con la división real, como se puede comprobar en el gráfico.



También es cierto que los datos, a simple vista, son fácilmente separables por una línea recta asumiendo un error aceptable, por lo que este conjunto de datos no es excesivamente complejo y es por ello que no debemos esperar unos errores demasiado altos, lo cual se cumple en la práctica.

2.2 EXPERIMENTACIÓN CON MODELOS MÁS COMPLEJOS

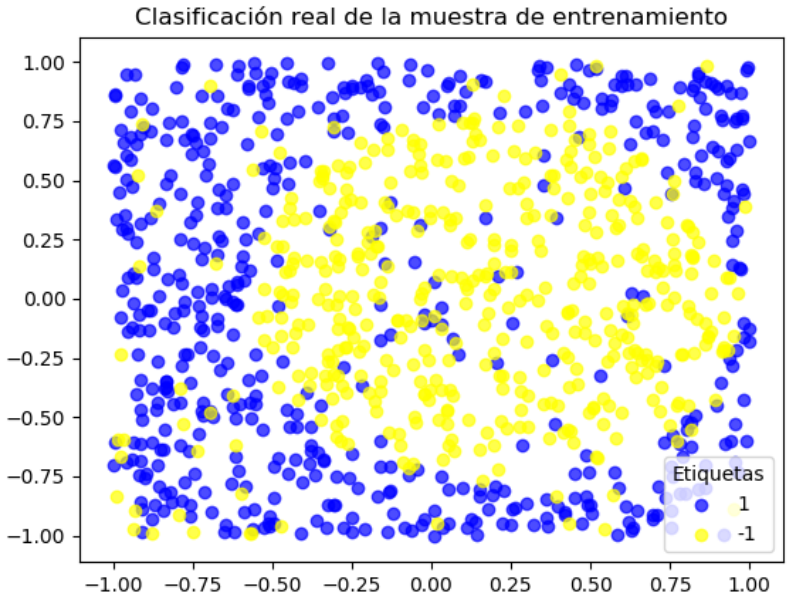
En este ejercicio vamos a explorar como se transforman los errores cuando aumentamos la complejidad del modelo lineal usado, para lo que vamos a crear 1000 puntos aleatorios uniformemente distribuidos en el cuadrado X =[-1,1] x [-1,1].



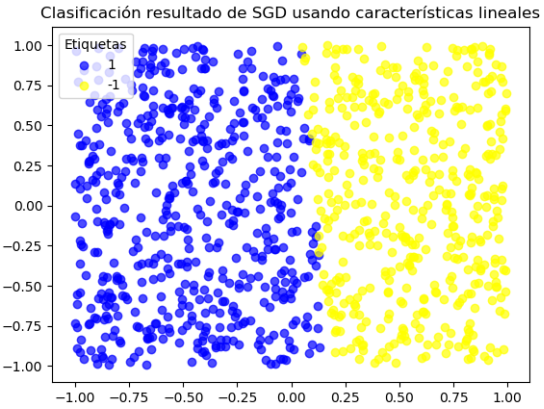
A partir de ésta muestra vamos a clasificar los puntos de nuestra muestra de entrenamiento según la siguiente función:

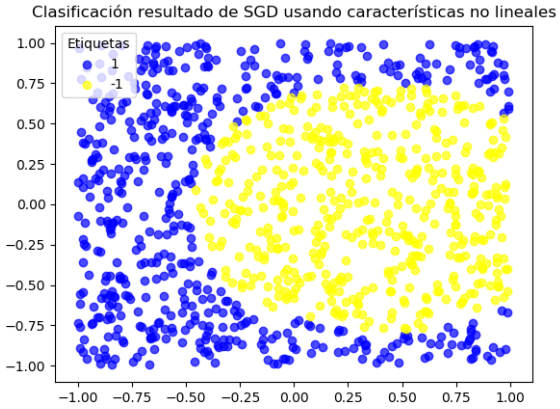
$$f(x_1, x_2) = sign((x_1 - 0.2)^2 + x_2^2 - 0.6)$$

añadiendo además un ruido del 10%, por lo que obtenemos la siguiente gráfica que tendremos como referencia para los siguientes apartados.



Ahora, mediante SGD, vamos a intentar clasificar la muestra de entrenamiento anterior usando como vector de características $(1, x_1, x_2)$, características lineales, así como $(1, x_1, x_2, x_1x_2, x_1^2, x_2^2)$, características no lineales, con lo que obtendremos unos resultados como los siguientes:





E_{in}: 0.9096

E_{in}: 0.6686

Como ya se ha visto anteriormente ahora estamos trabajando con un conjunto más difícil de clasificar, de hecho no es separable mediante un línea recta. Debido a esto si usamos características lineales vemos una división muy diferente a la real y, asociado a esta, un error enorme de 0.9096; mientras que mediante características no lineales obtenemos un resultado que, a pesar de su error de 0.6686, al menos presenta una forma parecida a la real. Hay que tener en cuenta también que el ruido del 10% hace que, aunque SGD nos diera la elipse perfecta, tuviéramos algo de error.

Por último vamos a hacer el experimento con 1000 muestras distintas y otras 1000 asociadas a cada una para obtener un E_{out}, de manera que obtengamos una medida del error fiable para el caso de ajustar un modelo de regresión a este conjunto clasificado según f. Se representan los resultados en la siguiente tabla para poder estudiarlos.

Error según características	E_{in}	E_{out}
$(1,x_1,x_2)$	0.9299	0.9012
$(1,x_1,x_2,x_1x_2,x_1^2,x_2^2)$	0.6608	0.5417

Dados estos resultados, podemos deducir lo siguiente:

- E_{out} es perceptiblemente menor que E_{in}, lo cual tiene sentido ya que la muestra usada para calcularlo no contiene ruido.

- El resultado obtenido con características no lineales es marcadamente mejor, lo cual también es lógico, ya que podemos observar en la clasificación real que la solución es una elipse, y una elipse es imposible de dibujar con características lineales, por lo que la única manera de acercanos a ella es usar características no lineales.

Por ello podemos concluir que el modelo más adecuado es el modelo que usa características no lineales, ya que el otro no es capaz de separar la elipse, debido a que ambas clases no son separables mediante una línea recta, y por tanto es totalmente imposible que nos de buenas resultados por muy bueno que sea el algoritmo que use para encontrarlos. Sin embargo, el modelo no lineal es capaz de amoldarse de manera bastante apropiada al resultado, devolviéndonos una solución que me parece bastante aceptable.

3 BIBLIOGRAFÍA

Dudas y problemas con la implementación:

- https://www.analyticslane.com/
- https://realpython.com/
- https://docs.scipy.org/doc/numpy/index.html
- https://numpy.org/doc/1.18/index.html

Ideas de representación y refuerzo de los conceptos teóricos:

- https://matplotlib.org/
- https://www.wolframalpha.com/
- https://www.youtube.com/dotcsv
- https://www.youtube.com/playlist?list=PLLssT5z_DsK-h9vYZkQkYNWcItqhlRJLN