Lab. 3, Estimación de parámetros

Martínez Hernández, Dafne. angelica@ciencias.unam.mx
Olvera Trejo, Alberto. alberto0410@ciencias.unam.mx

30 de junio de 2022

Resumen: En el presente trabajo se muestra el procedimiento que se llevó acabo para encontrar el plano que mejor se ajustara a la distribución de los hipocentros de las réplicas generadas por el terremoto del 07 de septiembre del 2017.

Palabras clave: Parámetros, plano, Grid Search.

1. Introducción

Dada la base de datos que contiene las coordenadas x, y y z de los hipocentros de las réplicas generadas por el terremoto del 07 de septiembre del 2017 (donde la entrada x, está dada en valores de latitud, la entrada y en valores de longitud y la entrada z en valores de profundidad), se busca un modelo matemático que represente el plano que mejor se ajuste los hipocentros utilizando los métodos vistos en clase, en este caso son el método *Grid Search* y el método del *Descenso del Gradiente*.

2. Metodología

Para poder encontrar un modelo que se ajuste a las observaciones dadas, se utilizó el método de Grid Search, empleando la norma $L1$, para optimizar la función de costo solicitada.
Dado que que el modelo propuesto es un plano y la fórmula general de estos es $z = ax + by + c$, entonces se llevó a cabo la búsqueda de tres parámetros, los cuales son a, b, c
El primer paso fue cargar la base de datos en Python haciendo la modificación de los valores de z multiplicándolos por -1, esto se hizo ya que los valores originales indicaban una profundidad positiva, pero queríamos representarlos en el espacio, por lo que necesitábamos los valores negativos.
Posteriormente se definió la función de la norma $L1$ y la función para obtener los valores del plano.
El siguiente paso fue crear las rejillas por las cuales se iban a mover los parámetros; lo anterior se logró con la función *linspace*. Hay que recordar que es sumamente importante el cómo se definen los rangos de las rejillas, pues si el valor mínimo buscado no está dentro de los rangos, nunca se va a alcanzar. Para poder solucionar dicho problema se establecieron los siguientes rangos iniciales:

- El rango para el parámetro a fue de -100 a 100 dividido en 30 secciones
- El rango para el parámetro b fue de -100 a 100 dividido en 30 secciones
- El rango para el parámetro c fue de -10000 a 10000 dividido en 50 secciones

Los rangos de a, b fueron elegidos así porque al momento de visualizar los datos nos percatamos de que la pendiente del plano de iba a ser tan elevada, por lo que seguramente no iba a tomar valores fuera del intervalo $[-100, 100]$, lo que sí podría tomar valores muy extremos es el parámetro c, lo cual explica que su rango inicial es $[-10000, 10000]$

Una vez hecho lo anterior, se procedió a iniciar el método de grid search. Primero se crearon dos listas, una para poder almacenar los parámetros y otra para ir guardando el historial de las evaluaciones de la función de costo Para ello se usaron tres ciclos *for*, uno por cada parámetro. Dentro de los tres ciclos se calcularon los valores de *z_pre* (los datos de la profundidad que modelaba el plano obtenido) con la función *plano*

ya definida; luego se calculó el error con los parámetros actuales y se anexaron a la lista de errores, finalmente se anexaron los parámetros a su correspondiente lista.

Fuera de los tres ciclos for se procedió a buscar en la lista de errores el índicie que daba el error mínimo, lo anterior se hizo con la función *np.where*. Finalmente, se evaluó la función *plano* con los parámetros que tenían el error mínimo.

Para la parte de la visualización se utilizó la herramienta $\%matplotlibnotebook$ ya que permite hacer la gráfica interactiva, en particular sirve para poder rotar el plano y visualizar el modelo desde varios puntos de vista.

Una vez obtenido el primer modelo, se precedió a restringir más los rangos de cada parámetro para ver si el modelo encontraba otro mínimo, se repitió el procedimiento 3-4 veces hasta que se obtuvo dos veces el mismo resultado, lo cual interpretamos como que el modelo había encontrado un mínimo local ahí. Las restricciones finales son las siguientes

- El rango para el parámetro a fue de -43 a 45 dividido en 30 secciones
- El rango para el parámetro b fue de -42 a -38 dividido en 30 secciones
- El rango para el parámetro c fue de -3470 a 3460 dividido en 50 secciones

3. Resultados y discusión

Los primeros parámetros obtenidos con el método Grid Search fueron $[-44{,}8276, -51{,}7241, -3469{,}3877]$. Por lo que el primer modelo matemático obtenido es el siguiente:

z_predicción$= -44{,}8276x - 51{,}7241y - 3469{,}3877$

Que al momento de graficarlo junto con los puntos obtenemos la figura 1, junto con la gráfica de la función de error de la imagen 2

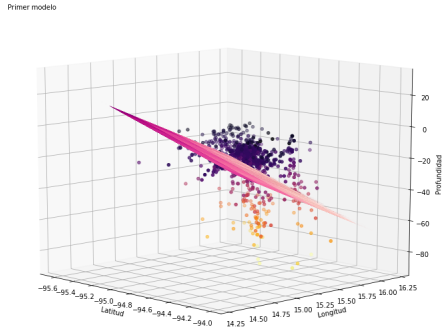

Figura 1: Primer plano

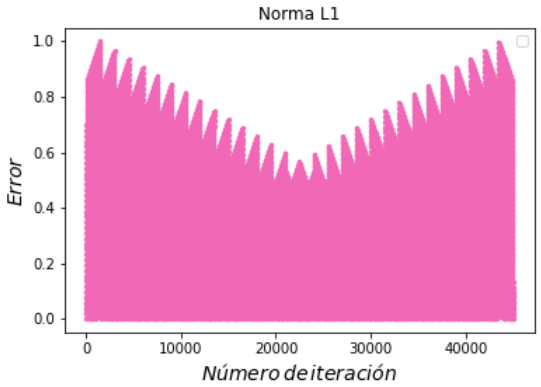

Figura 2: Minimizando la función de costo

Una vez que ajustamos los parámetros ingresados al Grid Search y se refinó la rejilla, los valores a, b, c que se obtuvieron son$[-43{,}0, -40{,}896, -3462{,}85]$, obteniendo así un modelo del plano donde:

z_predicción$= -43x - 40{,}896y - 3462{,}85$

Siendo éste el modelo final que se obtuvo para el plano que mejor se ajusta a la distribución de los hipocentros de las réplicas del terremoto, el cual se muestra en las imágenes 3, 4, 5.

La gráfica resultante de la función de error en el último grid search es la figura 6

Los aspectos relevantes del método Grid Search es que, aunque nos ayudó a encontrar los parámetros óptimos que se buscan, son muy importantes los parámetros iniciales que se ingresan en él, al igual que el refinamiento de la rejilla, esto debido a que si se dan ”malos”parámetros el modelo se distorciona completamente y aunque haya sido óptimo para ellos, no necesariamente es el óptimo del problema que se plantea al inicio. Asimismo,

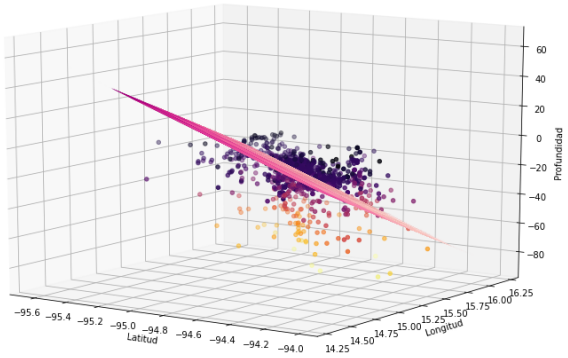

Figura 3: Segundo plano

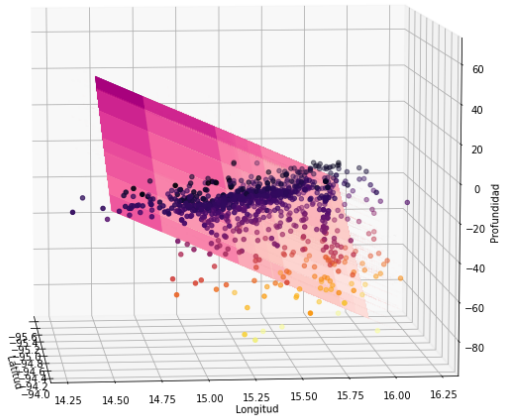

Figura 4: Segundo plano

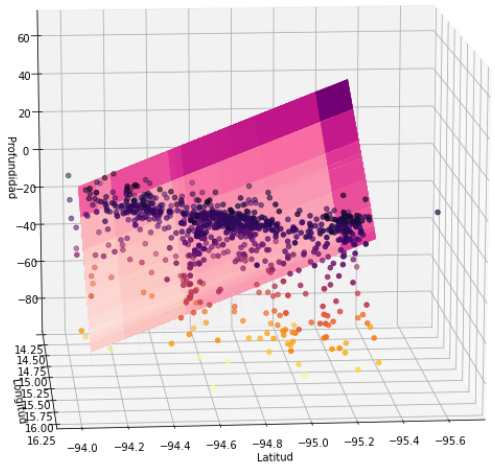

Figura 5: Segundo plano

el refinamiento de la rejilla juega un papel muy importante tanto en la búsqueda de la solución como en la complejidad del algoritmo, si se da una rejilla muy ”gruesa”, el margen de error será considerablemente grande, pero si se da una rejilla demasiado fina, como el código que se ocupó utiliza 3 ciclos *for* anidados que dependen de la longitud de los intervalos iniciales, la computadora tardará muchísimo más tiempo en efectuar las operaciones.

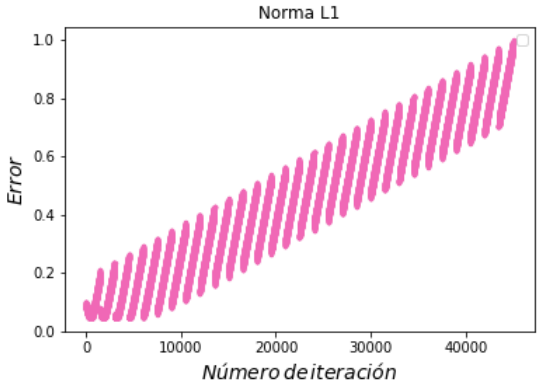

Figura 6: Función de costo

Como se mencionó anteriormente, la parte principal del código del algoritmo Grid Search que su¿e utilizó, hace uso de 3 ciclos *for* anidados, donde por cada uno de ellos se efectúa n, m, k veces respectivamente (dependiendo de la longitud de los intervalos iniciales), suponiendo sin pérdida de la generalidad que k es el mayor entre ellos, el grado de complejidad computacional queda acotado por k^3, por lo que podemos afirmar que el posible grado de complejidad del algoritmo pertenece al orden cúbico ($O(n^3)$).

4. Conclusión

Los métodos de optimización tienen un rango muy amplio de aplicaciones, pueden ir desde minimizar el costo de un viaje familiar hasta encontrar el plano que mejor se ajuste a la distribución de los hipocentros de un sismo. Lo anterior no sería posible sin la ayuda del computador, ya que al ser humano le tomaría mucho tiempo, mucho personal y mucho dinero poder hacer todos los cálculos que a la computadora le llevó un par de segundos (además de que no hubo errores). Con esto nos damos cuenta del potencial que tiene la modelación matemática