Informe de Métodos Numéricos
Tarea 10, Ajustes de Parámetros Experimentales 3.0: Estadística Bayesiana

Leonardo Leiva

1. Resumen

2. Introducción

En este informe se mostrarán algunos procedimientos para la estimación de parámetros a partir de datos experimentales por medio de técnicas bayesianas.

2.1. Estadística Bayesiana

Es un área de la estadística que se basa en usar una creencia inicial para el cálculo de una probabilidad. Los distintos estados se presentan en grados de creencia.
En este caso se busca una serie de parámetros que ajusten de mejor forma a una serie de datos y obtener una probabilidad de que dichos datos se representen por estos parámetros. La probabilidad corresponde a:

$$P(\vec{\alpha}|\vec{x}, Modelo) = \frac{P(\vec{x}|\vec{\alpha}, Modelo)P(\vec{\alpha}|Modelo)}{P(\vec{x}|Modelo)} \tag{1}$$

Que corresponde al Teorema de Bayes. La ecuación anterior dice que la probabilidad de los parámetros dado los datos y el modelo es igual al cuociente: entre la probabilidad de los datos dado los parámetros y el modelo, multiplicado por la probabilidad de los parámetros dado el modelo; y la probabilidad de los datos dado el modelo.
En general, el término del denominador al lado derecho es dificil de calcular ya que se necesitaría tener una probabilidad del modelo con respecto a todo tipo de modelos, lo cual es bastante complejo porque primero sería necesario decidir que modelos entrarían en juego. De cualquier manera este problema resulta tener una solución abordable al poder ser reemplazada por una constante de normalización. Por otro lado, el primer término del numerador se llama verosimilitud y el segundo es la probabilidad a priori que se le asigna de forma inicial a los parámetros correspondiente a una adivinanza. Esto último es uno de los principales cuestionamientos del método bayesiano porque en ocasiones el resultado final depende de los parámetros a priori, pero si se le da una adivinanza suficientemente buena y un error asociado apropiado las influencias pueden ser reducidas.

2.2. Hamiltoniano Monte Carlo

Corresponde a un método de metropolis aplicable a espacios continuos. En especifico se refiere a "No U-turn Hamilitonian Monte Carlo (NUTS)". Para más detalles teóricos sobre el método de metrópolis se puede revisar la tarea 8 [2]. Lo fundamental de este método es el uso de gradientes para determinar la dirección de máximo desenso y reducir el comportamiento del paseo aleatorio.
El método que se dispone está previamente implementado, por lo que detalles más precisos sobre lo que hace teóricamente se pueden encontrar en la documentación del algoritmo[3].

2.3. Modelos

Se busca modelar mediante técnicas bayesianas un espectro de abosroción dado en los datos del archivo 'espectro.dat'. La idea es similar a la tarea anterior[1] pero en este caso se usará un método diferente, como se mencionó anteriormente. Además, los modelos que se usarán serán un poco diferentes: Una gaussiana para el primer modelo y la suma de dos gaussianas para el segundo modelo. Para simplificar en comparación con la tarea anterior, se dejará la recta con pendiente 0 y el coeficiente de posición igual a $n = 1e - 16$. y el centro de las gaussianas como $\mu = 6563\mathring{A}$. De esta manera, el modelo 1 tendrá 2 parámetros libres y el modelo 2 tendrá 4 parámetros libres.

3. Procedimiento

3.1. Primer Intento: NUTS

Inicialmente se intetó usar el método NUTS en base al demo visto en clases [4] usando la librería 'pymc3', pero se presentaron diversos problemas. En primer lugar, python reconoció la diferencia entre las variables de la forma arreglo de Numpy, mientras que la librería usada usa variables del tipo tensor.
Se buscó actualizar numpy ya que en versiones superiores se arreglaba el problema entre arreglos de numpy y tensores, pero no dió resultados.
Se reescribieron las variables en forma de tensores por medio de la librería 'Theano'. A pesar de que en este caso no arrojaba errores en el procedimiento, si existían errores en la implementación, porque los resultados aproximados entregados eran los mismos valores que se le daban como punto de partida.
Finalmente se instalaron algunos complementos extras que se sugerían en la documentación de la librería, pero seguían entregando errores incluso corriendo los mismos demos que estaban en la página. Se desiste de esta opción al no encontrar la manera de solucionarlo. Los procedimientos desarrollados se encuentrar en el archivo 'parte1.py', donde están solamente los algoritmos de prueba para NUTS.

3.2. Segundo Intento: A la antigua

Mediante el método de calcular todas las verosimilitudes y probabilidades a priori de la ecuación (1) se trató de obtener los parámetros óptimos.
Primero se definen los modelos con los cuatro parámetross móviles que se describen anteriormente. Luego se agregan las probabilidades a priori y verosimilitud muy similares al demo de la clase [5]. Se define también las funciones para llenar las grillas con los valores de la probabilidad de los datos. La idea es obtener una aproximación de dichos valores.
Cambiando las adivinanzas iniciales se intentó encontrar los parámetros apropiados. se usó la forma de graficar de los demos antes descritos. Con esto se fue cambiando la adivinanza y los valores maximos y minimos para ajustarlos de forma más cercana posible a los valores reales. Por conveniencia se reescaló la amplitud para que no fuera tan pequeña y se pueda trabajar de forma más detallada. El factor escogido fue de 10^15, pero queda pendiente decidir si era el factor más apropiado.
Una vez están llenas las grillas se puede calcular la evidencia de cada parámetro y las desviaciones de los mismos haciendo uso de la probabilidad marginal (al dejar fijo un parámetro y sumar todos los valores en los demás parámetros). Con estos resultados se obtiene la evidencia para cada modelo y el cuociente entre ellas permite decidir cual es el mejor modelo: Si la probabilidad de los datos dado el primer modelo es mayor que el del segundo modelo (su cuociente es mayor que 0), entonces el primer modelo es mejor que el segundo. Si es al revés, entonces, se prefiere el segundo modelo.

4. Resultados y Análisis

Del intento con NUTS no hubo muchos resultados concretos dado que no se logró hacer funcionar el método, pero se puede decir que hace falta aprender más para poder implementarlo, tanto del manejo de variables y librerias nuevas como de la instalación de estas. A pesar de ser un método muy llamativo e interesante por teoría (uso de Hamiltonianos en problemas con probabilidades) no se pudo llegar muy lejos.
Para el segundo caso se realizaron las pruebas con el modelo simple, porque se visualizaban de buena manera los datos que entrega. Inicialmente costó encontrar un gradiente de probabilidades en el cual guiarse para encontrar los parámetros óptimos, pero luego de cambiar la escala se pudo llegar a los valores en los cuales se observaba un comportamiento gaussiano (antes de eso solo se observaba una recta). Centrando los bordes su pudo obtener una aproximación de los valores. Por tiempo no se logran establecer los valores numéricos, además de que llama la atención que difieren mucho de los valores numéricos que se pueden obtener por 'leastsq': Para el primer modelo, $A = 7{,}61 * 10^{-17}$ y $\sigma = 3{,}701$, mientras que para el segundo $A_1 = 4{,}105 * 10^{-17}$, $A_2 = 4{,}854 * 10^{-17}$, $\sigma_1 = 2{,}44$ y $\sigma_2 = 8{,}419$, donde A corresponde a las amplitudes (A_1 y A_2 para la suma de gaussianas) y los σ a las desviaciones. Llama mucho la atención y permite dudar que el método esté bien implementado el que al cambiar la adivinanza para la amplitud (en el modelo de una sola gaussiana) el resultado

cambie bastante, corriendo la guassiana en dirección a la adivinanza. También puede deberse a que la grilla tiene pocos valores (50, en comparación a los 200 del demo). A pesar de esto, preocupa más que el valor central de lagaussiana para la amplitud esté cerca de -70 y no cerca de $0{,}07$ como debería ser según la aproximación vía leastsq, tomando en cuenta la escala que se usó (10^{15}). Se había probado con valores positivos para la amplitud, pero, como se mencionó anteriormente, pero al intentar acercarse al valor con más probabilidad se llegó a valores negativos. Se creyó inicialmente que podía haber cambiado el signo durante la aproximación, pero que era parte natural del algoritmo, pero se descarta esta opción porque el algoritmo de levenberg marquardt no mostró errores en ese sentido.
La figura (1) muestra la probabilidad de los parámetros para el primer modelo:

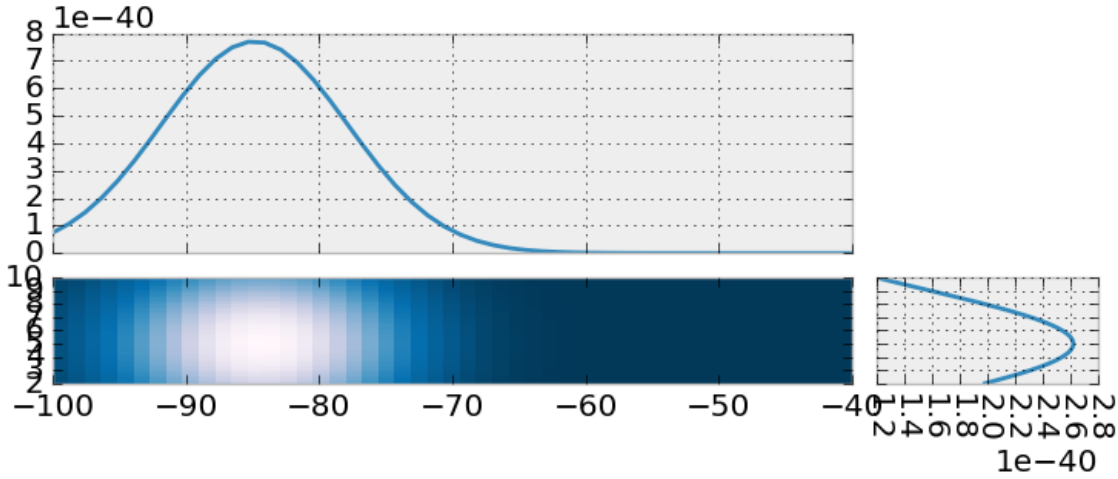

Figura 1: Función de densidad de probabilidad acumulada del modelo de Gauss (en rojo) y de los datos experimentales (en azul).

4.1. Conclusiones

Se concluye que hace falta tener más conocimientos y profundizar más sobre el tema para poder usar NUTS. Resulta llamativo la manera de abordar el problema visto desde ese método, pero no se logra implementar. Queda propuesto dedicar más tiempo a este algoritmo.
Se saca como aprendizaje pensar un poco más antes de aventurarse en un método tan diferente como es NUTS: El tiempo que se le dedicó impidió avanzar más con el otro método y entenderlo de mejor manera. Tal vez haber dedicado más tiempo desde un comienzo habría permitido obtener resultados en el otro método.
Para el intento de calcular las grillas se nota que la diferencia entre el tiempo que tarda para dos parámetros con respecto a cuatro parámetros es considerable. Utilizar herramientas de multiprocesing hubiera facilitado las cosas, pero tampoco se logró demostrar que el método funcionara apropiadamente para el modelo simple, dado que el valor numérico por otra aproximación es diferente al que se acerca este método. Se concluye que tiene que haber un

error en el método porque no debería dar un valor negativo y es por esto que no se toma demasiado en cuenta el optimizar los algoritmos que siguen después, porque ya en la parte inicial tendría un error. Queda pendiente encontrar la falla.

Referencias

[1] Informe 10 Metodos Numericos - Leonardo Leiva.
[2] Informe 8 Metodos Numericos - Leonardo Leiva.
[3] Documentación Hamiltonian Monte Carlo http://pymc-devs.github.io/pymc3/getting$_started/installation$
[4] Demo de la Clase: Hamiltonian Monte Carlo https://gist.github.com/thevalentino/f193d7bf6ef28bfa8a7b
[5] Demo de la Clase: Estadística Bayesiana https://gist.github.com/thevalentino/41335571feb0b8300953