GF7013 - Métodos Inversos Avanzados

Tarea 3

Ajuste Ortogonal de una Linea Recta usando Métodos de Muestreo Bayesianos: Algoritmos de Transitional Markov Chain Monte Carlo (TMCMC) + Remuestreo

Fecha Publicación: Miércoles 2 de julio de 2025
Fecha Última de Entrega: Sábado 19 de julio de 2025 (último día de exámenes)

En esta tarea se agregará funcionalidades al paquete de Python 3 llamado GF7013, que comenzaron a desarrollar en las Tareas 1 y 2. En particular se agregará el algoritmo de Transitional Markov Chain Monte Carlo (TMCMC) y se aumentará la utilidad de éste agregándole un paso de remuestreo. PARA ESTA TAREA DEBEN ENTREGAR EL PAQUETE GF7013 ACTUALIZADO CON LOS CODIGOS A DESARROLLAR. RECUERDEN QUE SI NECESITAN UN SERVIDOR PARA CORRER LA TAREA LO PUEDEN SOLICITAR ENVIANDO UN CORREO.

Se provee una carpeta llamada Tarea3, con un archivo que se referencia mas adelante. Reemplaze la carpeta Tarea3 que se encuentra al interior de la carpeta `GF7013/bin/` con la que se provee con este enunciado.

Correcciones Tarea 2

En esta tarea se hará uso extensivo de lo desarrollado en las Tareas 1 y 2. Por lo que el primer paso en su desarrollo debe ser hacer las correcciones necesarias para que el algoritmo de Metropolis en Paralelo funcione correctamente. Para ello use las instancias de clases auxiliares y horas de consultas con el profesor auxiliar y profesor de cátedra.

P1. Resolución del problema inverso: Programación del Algoritmo TMCMC

En esta etapa de la tarea se programará y verificará el algoritmo TMCMC (la versión sin remuestreo). Copie la carpeta `tmcmc` proveída en esta tarea en su carpeta `GF7013/sampling/` reemplazando las existentes. Lea la documentación de los módulos de python y las definiciones de las funciones al interior, le darán mayor claridad para definir los siguientes pasos de la tarea.

Enel módulo `tmcmc.py` se encuentra un prototipo de algoritmo TMCMC que usted debe completar. Para el correcto funcionamiento del algoritmo también debe programar la función que permite calcular los $\Delta\beta$ en cada iteración del algoritmo en el módulo `calc_dbeta.py`

P1.1. (1.0 pts) Módulo calc_dbeta.py

Una parte esencial del algoritmo TMCMC es la transición de un valor del exponente β al siguiente. De acuerdo a lo visto en clases, el valor de β para la siguiente iteración del algoritmo se calcula de manera tal que, de las muestras obtenidas para el valor actual de β, el 50 % de estas sea representativa de la fdp definida para el valor actualizado de β. Note que debe considerar ambos casos: a) cuando se trabaja directamente con los valores de verosimilitud (`use_log_likelihood = False` en el ensemble) y b) cuando se trabaja con el logaritmo natural

de los valores de verosimilitud (`use_log_likelihood = True` en el ensemble). Ver definiciones del ensemble en la Tarea 2.

P1.2. (1.0 pts) Módulo tmcmc.py

Complete la programación del algoritmo TMCMC en el módulo `GF7013/sampling/tmcmc/tmcmc.py`. Note que se contempla ambos casos: 1) en que se calcula las verosimilitudes de la distribución de probabilidad f_{prior} y de la función de verosimilitud ($\mathcal{L}$), asi como 2) en que se calcula el logaritmo natural de dichas verosimilitudes. Note que en la definición de la función `tmcmc_pool` hay una variable llamada `use_resampling`, ignórela por el momento.

P1.3. (0.5 pts) Verificación del algoritmo

Cree una copia de `Tarea2/tests/test_metropolis_paralelo_POOL.py` desarrollado en su Tarea 2 en la carpeta `Tarea3/tests/` y cambie el nombre del archivo a `Tarea3/tests/test_tmcmc.py`. Modifique este último para verificar que su algoritmo TMCMC funciona correctamente (considere ambos casos de `use_log_likelihood`). Comente sus resultados comparándolos con los del test de la Tarea 2, `Tarea2/tests/test_metropolis_paralelo_POOL.py`.

P2. Resolución del problema inverso: Programación del Algoritmo TMCMC + Remuestreo

En esta etapa de la tarea se programará y verificará el algoritmo de TMCMC al cual se ha agregado un remuestreo de las muestras después de la actualización del valor del exponente β. Para ello, se utiliza como base el algoritmo TMCMC recién programado.

P2.1. (1.0 pts)Algoritmo de Remuestreo

Programe los códigos del algoritmo de remuestreo, en el módulo `GF7013/sampling/tmcmc/resampling.py`, que se utilizará en cada transición del algoritmo TMCMC. El algoritmo de remuestreo a programar es el visto en clases, que toma un conjunto de muestras obtenidas para la iteración del TMCMC con un valor del exponente β dado, y las remuestrea para que sean muestras de la distribución de la siguiente iteración del TMCMC donde el valor del exponente igual a $\beta + \Delta\beta$. Note que el algoritmo de muestreo debe contemplar ambos casos: 1) en que se calcula las verosimilitudes de la distribución de probabilidad f_{prior} y de la función de verosimilitud ($\mathcal{L}$), asi como 2) en que se calcula el logaritmo natural de dichas verosimilitudes.

P2.2. (0.5 pts)TMCMC + Remuestreo

Modifique la programación del algoritmo TMCMC en el módulo `GF7013/sampling/tmcmc/tmcmc.py` para que tenga la opción de aplicar el algoritmo de remuestreo justo después de actualizar el valor de β. Para ello debe usar la variable booleana llamada `use_resampling` codificada en la definición de la función `tmcmc_pool`.

P2.3. (0.5 pts) Verificación del algoritmo

Cree una copia de `Tarea3/tests/test_tmcmc.py` en `Tarea3/tests/test_tmcmc_resampling.py`. Modifique este último para verificar que su algoritmo TMCMC + Remuestreo funciona correctamente (considere ambos casos de `use_log_likelihood`). Comente sus resultados comparándolos con los del test de la P1.3 de esta tarea.

P3. (1.5 pts) Aplicación al Problema de Ajuste Ortogonal a la Recta

Resuelva el problema de Ajuste Ortogonal a la recta, (ver datos sintéticos para esta pregunta en P32.py de la Tarea 2). Para ello asuma que los parámetros que definen la recta quedan restringidos a los intervalos siguientes: $a \in [-15, 15]$ y $\theta \in [-360, 360]$. Genere los modelos iniciales como muestras de f_{prior}. Haga un gráfico mostrando los histogramas que aproximan las fdp marginales para a y θ y otro gráfico en donde se muestra la fdp conjunta (para este último use matplotlib.pyplot.hist2d) tanto para las muestras de f_{prior} como para las muestras de f_{post}. Comente sus resultados comparando con los obtenidos en la Tarea 2, y justificando su elección del valor de la variable booleana `use_log_likelihood`.

Desarrollar esta parte de la tarea en la carpeta `Tarea3/P3`.

Instrucciones

- La tarea se realizará en grupos (ya definidos en Ucursos).
- Se prohíbe la colaboración entre grupos.
- Copiar la estructura de carpetas que se entrega en esta tarea dentro del paquete GF7013 que desarrollaron para la Tareas 1 y 2.
- Al entregar la tarea deben entregar, el informe debe ser escrito en word, latex, o en un Jupyter notebook (donde puede cargar los resultados los diferentes procesos y hacer los gráficos, o cargar dentro del notebook las imagenes de sus resultados) esto de manera ordenada y se entienda claramente los gráficos y las respuestas a cada parte de la tarea. A u-cursos se debe subir un archivo comprimido que tenga la estructura de carpetas del paquete `GF7013` con los códigos e informe al interior.
- Asimismo es una excelente práctica el escribir códigos que sean modulares de manera de poder reutilizar la mayor cantidad de veces ese código (obviamente sin abusar).

Exito!!!..