Proyecto Final de Programación Básica: Cadenas de Markov Chain Montecarlo con el método Metrópolis.

Universidad de Guanajuato, División de Ciencias e Ingenierías
Programación Básica
Profesora: Dra. Alma Xochitl González Morales
Alumnos: Guadalupe Sinaí Florián Landa, Ian Barrera Salas

December 8, 2018

1 Descripción del Programa

El método de ajuste de parámetros conocido como minimización de X^2 es un caso particular. En el programa y_i^{obs} y x_i^{obs} son un conjunto de coordenadas recolectadas de un experimento de péndulo oscilatorio dond x=el periodo de oscilación τ y y=el largo del cable l, los cuáles tienen su error σ_x σ_y respectivamente. Para encontrar el valor de la pendiente ocupamos un ajuste de minimos cuadrados expresada por la ecuación:

$$X^2 = \sum_i \frac{(y_i^{obs} - y_i^{mod}(x_i, m, b)}{\sigma_i^2} \tag{1}$$

El problema en el programa a resolver es la probabilidad de que cierto valor de la pendiente sea el valor verdadero que se ajuste a los datos. Esto se puede resolver con el teorema de bayes:

$$P(\overrightarrow{\theta}\,|\,\overrightarrow{y}) \propto L(\overrightarrow{y}\,|\,\overrightarrow{\theta}) * Pr(\overrightarrow{\theta}) \tag{2}$$

$$P(A_i/B) = \frac{P(A_i) \cdot P(B/A_i)}{P(B)}$$

Donde:
$P(A_i)$ = Probabilidad a priori
$P(B/A_i)$ = Probabilidad condicional
$P(B)$ = Probabilidad Total
$P(A_i/B)$ = Probabilidad a posteriori

Figure 1: Ecuación del teorema de Bayes, especificando el significado de cada variable.

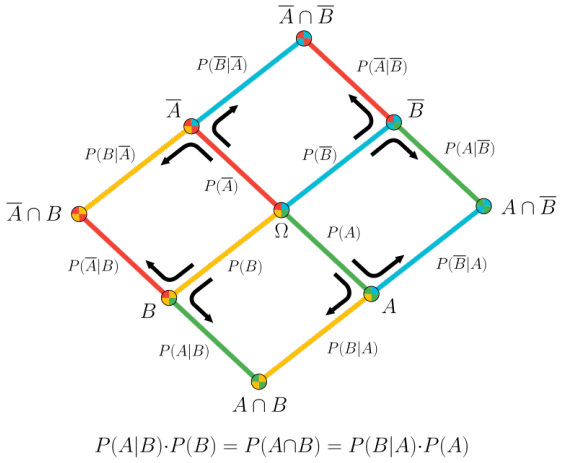

Figure 2: Visualización del treorema de Bayes por superposición de dos árboles de decisión.

P es la distribución de probabilidad prosterior, L es Likelihood o probabilidad de de verosimilitud, Pr es la distribución a priori. θ representa los parámetros a ajustar en el modelo inicial. Lo que se busca es encontrar el máximo de distribución posteriior, es de decir, los parámetros que suponen mayor probabilidad de describir correctamente los datos iniciales $\vec{\theta} = (m, b)$ que son la pendiente y ordenada al origen. Lo más útil en este caso es suponer una distribución Gaussiana para L que se relacione con X^2:

$$ln(P(y|m)) = 0.5 \sum_i ((y_i^{obs} - y_i^{mod}(x_i, m)) - \sigma_i^2) \tag{3}$$

Por lo que el teorema de Bayes se expresaría como:

$$ln(P(y|m)) \approx ln(L(\vec{y}|\vec{\theta})) + ln(Pr(\vec{y}|\vec{\theta})) \tag{4}$$

También se usará la distribución a priori plana, considerando los parámetros m,b y su probabilidad a priori de describir los datos en intervalos $m_{min} < m < m_{max}$ y $b_{min} < b < b_{max}$, si lo anterior se cumple el valor a priori será igual a zero. El método montercarlo de maximización es Metropolis-Hasting, que genera parámetros m,b de forma aleatoria y calcula la L a priori de dichos parámetros. Cada conjunto nuevo que se genera depende del valor de L y que tan mayor o menor al anterior sea.

A continuacón se describirá el conjunto de programas escritos en lenguaje C, que nos permitió encontrar la pendiente con los datos (x,y) y error dados desde un archivo.

2 Comandos usados en el Programa

perce.h

Este programa nos ayudó a crear una libería con funciones que se definirán en el sigueinte programa escrito:

Primero se indicaron las librerías que se usarán en el programa, las cuales incluyen definiciones comunmente usadas en algoritmos. En este programa se usaron: dos librerías estandar y la libería con varias funciones matemáticas:

$$include < stdio.h > \tag{5}$$

$$include < math.h > \tag{6}$$

$$include < stdlib.h > \tag{7}$$

float nos ayuda a declarar variables con valores de punto flotante, percentx() y percenty() nos ayudan a declarar las funciones que estarán en la liberría perce.h

$$float\ percenty\ (float\ m0,\ float\ b0); \tag{8}$$

$$float\ percentx\ (float\ m0,\ float\ b0); \tag{9}$$

func.c

Este programa nos ayudó a definir las acciones que realizan las funciones que se encuentran dentro de la libería perce.h:

Se declara la función percentx(), la cual es sin argumentos de entrada pero con argumentos de salida . El nombre de la función debe ser único y exclusivo para dicha función.

$$float\ percentx\ (float\ m0, float\ b0) \tag{10}$$

int, float y FILE ayudan a delcar las variables de tipo entero, punto flotante y archivo respectivamente. result [10] crea un arreglo con capacidad de 10 datos que se podrán guardar dentro de él. *xobs crea una variable de tipo apuntador con tamaño de float.

$$inti, j, n = 10; \tag{11}$$

$$float\ *xobs,\ *yobs,\ *sigmam,\ *sigmab,\ result[10]; \tag{12}$$

$$FILE * datos; \tag{13}$$

Se reserva el espacio en la mamoria usando malloc, que asigna un número de bytes indicados y devuelve un apuntador al primer byte del espacio asignado (un solo bloque), y se le indica que el total de bytes son el valor num por el tamaño de una variable entera

$$xobs = (float*)malloc(10 * sizeof(float)); \tag{14}$$

Abro el archivo dónde se encuentran los datos de (x,y):

$$info = fopen\ (\text{"}datos\text{"},\text{"}r\text{"}); \tag{15}$$

Se inicia un ciclo dónde el contador de las coordenadas i inicia en cero, la condición de seguir realizando el ciclo es que se repita un un numero menor al valor de la variable "n" y que al coontador se le aumente una unidad cada vez que se completa la serie de instrucciones. fopen lee los datos ya formateados del stdin y los guarda los valroes en las variables indicadas con "":

$$for(i = 0; i < n; i++) \tag{16}$$

$$fscanf(datos,\text{"}\%f\ \text{"},\ xobs[i]); \tag{17}$$

Cierro el archivo con datos:

$$fclose(datos); \tag{18}$$

Se inicia un ciclo dónde el contador de las coordenadas i inicia en cero, la condición de seguir realizando el ciclo es que se repita un un numero menor al valor de la variable "n" y que al coontador se le aumente una unidad cada vez que se completa la serie de instrucciones.

$$for\ (i = 0; i < 10; i++) \tag{19}$$

Se calcula la L=likelihood por el método metrópolis, el cual indica que el arreglo xobs[i] se multiplica por la m0 y se suma a b0, y luego se resta al arreglo yobs [i], esto se guarda en result [i]. En la segunda los arreglos de sigma se multiplican y se restan con la raíz del arreblo result [i], dicho valor es cuardado den result [i]. Se procede a relaizar una sumatoria con los valores del arreglo result y se multiplicar por -0.5. De dicho resultado se obtiene la exponencial que se guarda en res2.

$$result[i] = yobs[i] - (xobs[i] * (m0)) + b0; \tag{20}$$

$$result[i] = pow(result[i], 2) - (sigmam[i] * sigmam[i]); \tag{21}$$

$$suma+ = result[i]; \tag{22}$$

$$suma* = -0.5; \tag{23}$$

$$res2 = pow(2.718282, suma); \tag{24}$$

La segunda función es percenty, que se asemeja con con la función percentx, la única diferencia es que en la función percenty no se obtiene el exponencial de su respectiva sumatoria.

mainp.c

Este es el programa principal que calculará la pendiente m y la ordenada b que más se aproxime de acuerdo con las coordenadas (x,y) del experimento.

Se declaran las variables int, float y double que definen variables de tipo archivo, entero, punto flotante con 6 cifras y punto flotante con 12 cifras respectivamente.

$$FILE\ *resultados; \tag{25}$$

$$int\ i,\ j,\ n = 10; \tag{26}$$

$$float\ m0,\ b0,\ mn,\ bn; \tag{27}$$

$$double\ px0,\ pxn = 0,\ pxn2 = 0,\ px02,\ ale; \tag{28}$$

Se le pide al usuario, imprimiento en la pantalla con printf, entrar los valores de la pendiente y ordenada iniciales, además del número de veces que desea correr el ciclo para encontrar la pendiente y la ordenada.

$$printf(\text{"}Ingrese\ el\ valor\ de\ la\ pendiente\ y\ el\ valor\ de\ las\ ordenadas\ iniciales\text{"}); \tag{29}$$

Aquí se guarda el dato ya formateado del stdin en la variable m0.

$$scanf(\text{"}\%f\text{"}, m0); \tag{30}$$

Se abre el archivo donde se escribirán los resultados obtenidos del cáulculo:

$$resultados = fopen(\text{"}resultados\text{"},\text{"}w\text{"}); \tag{31}$$

Se inicia un ciclo dónde el contador j inicia en cero, la condición de seguir realizando el ciclo es que se repita un un numero menor al valor de la variable "delta" y que al coontador se le aumente una unidad cada vez que se completa la serie de instrucciones. Dicho ciclo encontrará la pendiente y la ordenada el número de veces que le haya indicado el usuario.

$$for(j = 0; j < delta; j++) \tag{32}$$

Se generan números al azar entre -1,1 y se le restan a los valroes iniciales guardados en m0,b0 respectivamente. Dicho valor se guarda en la variable mn,b0 respectivamente. El signo "*" indica una multiplicación y el signo "/" una división.

$$mn = m0 - (1) + (2) * (rand()/(double)RAND\_MAX); \tag{33}$$

Se llaman y se hace uso de las funciones percentx y percenty declaradas y descritas en el programa funcion.c y los valores de los cálculos de dichas funciones se guardan en px0, pxn y pxn2.

$$px0 = percentx(m0, b0); \tag{34}$$

$$pxn = percentx(m0, b0); \tag{35}$$

$$pxn2 = percenty(m0, b0); \tag{36}$$

Se crea una condición if donde si el valor guardado en pxn es mayor al valor guardado en px0 se igualaran los nuevos valores de mn y bn en m0 y b0 respectivamente.

$$if(px0 < pxn)\{ \tag{37}$$

$$m0 = mn; b0 = bn \tag{38}$$

else indica que si las condiciones anteriores dentro del if no se cumplen, se buscará un nuevo número aleatorio, y si dicho número es menor que pxn2 (ale¡pxn2) entonces dicho valor nuevo mn,bn ser guardará en las varialbes m0,b0. RAND_MAX es una constante que representa el valor más grande al azar que una función puede regresar.

$$else\{ \tag{39}$$

$$ale = rand()/(double)RAND_MAX; \tag{40}$$

$$if(ale < pxn2) \tag{41}$$

$$m0 = mn; b0 = bn \tag{42}$$

Con fprintf se imprimen los valores de m0,b0 obtenidos a cada término de ciclo, y resultados indica que en dicho archovo se imprimirán.

$$fprintf(resultados, "\%f\ \%f\ ",\ m0,\ b0); \tag{43}$$

Se cierra el archivo de resultados

$$fclose(resultados); \tag{44}$$

Indica si la secuencia de instrucciones sucedio correctamente, de lo contrario enviará signo de error

$$return0; \tag{45}$$

Makefile

El Makefile nos ayuda compilar un programa desde el orígen del código.
Se incluyen los nombres de los programas que se desean compilar simultáneamente.

$$mainp: \ func.c\ mainp.c\ libs \tag{46}$$

Gcc compila los programas func.c y mainp.c, junto con la libería libs en un programa llamado mainp.out, el -lm indica que se usó la libería con funciones matemáticas

$$gcc\ func.c\ mainp.c\ -o\ mainp.out\ -lm\ -I\ ./libs \tag{47}$$

Clean limpia el ejecutable, que es el programa mainp.out

$$clean: \tag{48}$$

$$rm\ mainp.out \tag{49}$$

Cuando se coloca la palabra "make" en la terminal, dentro de la carpeta donde se encuentran los programas, liberías y documentos que forman parte del ejecutable; y se presiona enter "mainp.out"se compilará y ejecutará.

graph1.ipynb

Este programa se usó para graficar los datos obtenidos de los resultados del programa ejecutado con Makefile, llamado datos.txt, minetras se encunetre dividido en 2 columnas de datos numericos.

Se llaman las librerías que se desean usar, en este caso serán las estandar y para funciones matemáticas.

$$import\ matplotlib.pyplot\ as\ plt \tag{50}$$

$$import\ numpy\ as\ np \tag{51}$$

Se leen los datos del archivo "restultados", se grafica con mathplotlib y se guarda la gráfica con plt.savefig.

$$file = "resultados" \tag{52}$$

$$data\_img = np.loadtxt(file) \tag{53}$$

$$plt.savefig('frecuencias\ absoluta\ pendietes.png') \tag{54}$$

Se leen los datos con los comandos mencionados anteriormente y se guardan en dos arreglos.

$$plt.savefig('frecuencias\ absoluta\ pendietes.png') \tag{55}$$

$$a = data_img[:, 0] \tag{56}$$

$$b = data_img[:, 1] \tag{57}$$

$$x = np.linspace(-5, 5, 100) \tag{58}$$

Se grafican en un rango de -5 a 5.

$$for\ i\ in\ range\ (1, 1000) : \tag{59}$$

$$plt.plot(x, a[i * 10] * x + b[i * 10]) \tag{60}$$

Se grafican como pendiente y eje respectivamente y los resultados se guardan en archivos ".png"

$$foriinrange(1, 1001) \tag{61}$$

$$file = name[i] \tag{62}$$

$$plt.plot(x, a[i * 10] * x + b[i * 10]) \tag{63}$$

$$plt.savefig(file + '\ .png') \tag{64}$$

3 Resultados

El porograma se ejecutó con éxito y realizó el proceso de buscar la pendiente y la ordenada en "y" que más concordaba con el set de 10 de coordenadas (x,y) dadas. Dicho proceso se realizó un número de 10000 veces, que fué el valor de delta dado por el usuario. Las gráficas realizadas fueron las siguientes:

Los datos obtenidos en python nos muestran la tendencia de los datos obtenidos por el programa. En seguida se muestran los datos reales de la pendiente m y la ordenada b para comparar que tan cerca estuvo el programa de calcular los valores reales buscados:

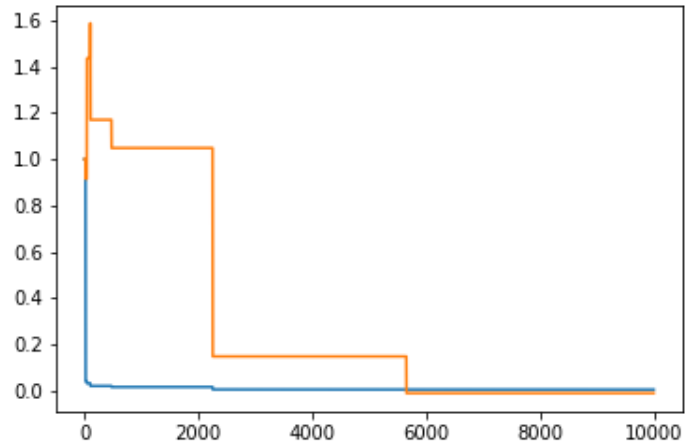

Figure 3: Se observa la tendencia de los valores de la pendiente y la ordenada.

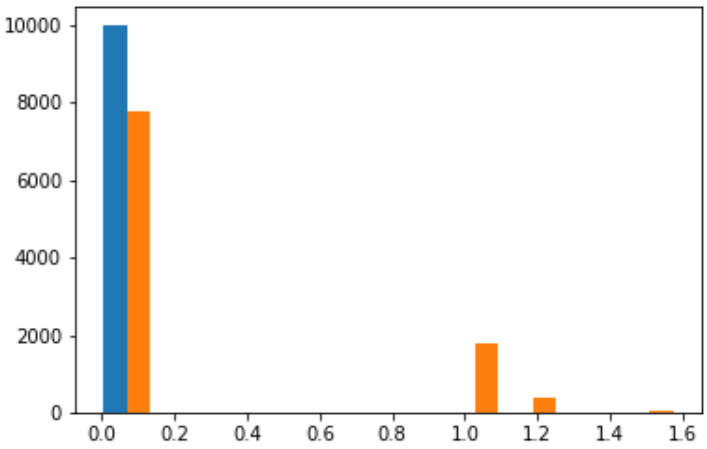

Figure 4: Histograma que representa el número de veces que se buscó el valor que mas se acercaba a la pendiente de los datos (x,y) ingresados y el valor de la misma.

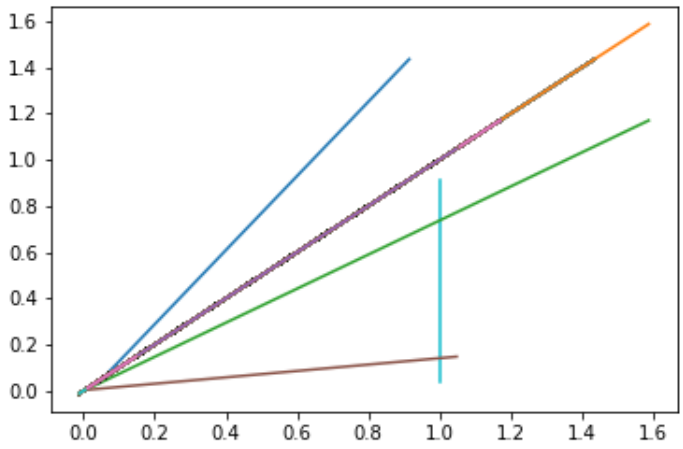

Figure 5: Pendientes encontradas.

```
10000    0.003527 -0.011711
```

Figure 6: Se observan: el número de veces que se buscó la pendiente y la ordenada, y el valor de la pendiente y valor de la ordenada encontrados.

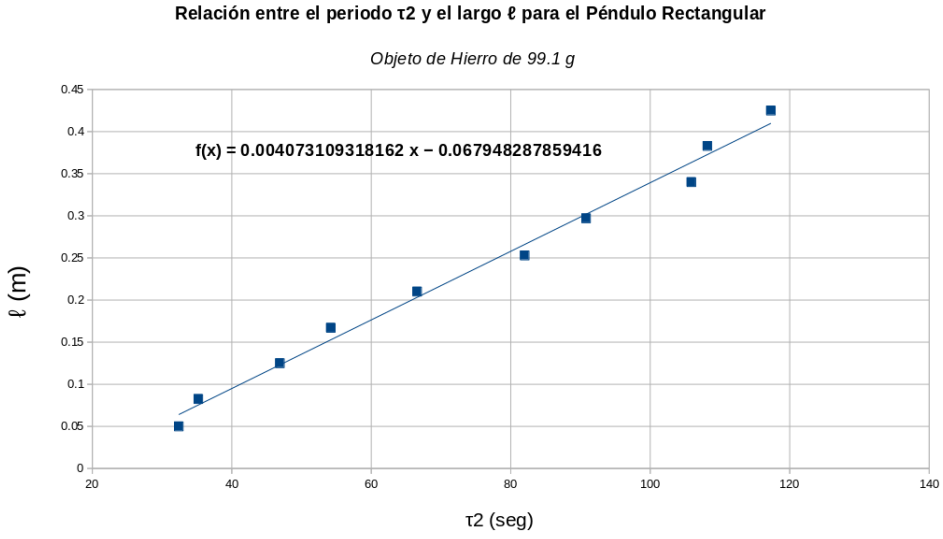

Figure 7: Se observan los puntos (x,y) con los cuales se ejecutó el programa y los valores reales de la pendiente m y la ordenada b.

4 Conclusiones

El programa usó el teorema de Bayes para encontrar la probabilidad de que el valor la pendiente de una gráfica se ajustara a los datos de 10 coordenadas dadas en x,y respectivametne. Al final, se notó la gran similitud entre los valores de los resultados obtenidos de la pendiente m y la ordenada b con el programa realizado, comparado con el valor real de la pendiente m y la ordenada b. Dichos resultados también se pueden obtener con el método de mínimos cuadrados, sin embargo, se escogió este programa debido a que el teorema de Bayes es una de las herramientas que ayuda a llegar al porcentaje de veracidad de un dato tomando en cuenta información ya conocida, también llamada información "a priori". Dicha herramienta tiene varias aplicaciones no solo para el área de la ciencia, también para el área de estadística y economía. El teorema de Bayes fué un tesoro que él dejó enterrado y que por desgracia, fué descubierto hasta después de su muerte. Ahora nos es de gran ayuda en sus aplicaciones para el uso y manejo de datos.