ParallelFischerScoring_binomial

Manual de Usuario

Tabla de contenido

1. Descripción 2
2. Disponibilidad 2
3. Requisitos 2
4. Iniciar aplicación 2
5. Realización de Pruebas 3
   Tabla 1. Características del Servidor 3
   5.1. Descripción datos de entrada. 4
   Figura 1. Datos de entrada al algoritmo Fisher Scoring 4
6. Muestra de resultados 5
   Figura 2. Comparación resultados temporales entre CPU y GPU. (Datos de entrada en miles) 5
   Tabla 2. Tiempos de ejecución en CPU y GPU 5
   Figura 3. Resultados ejecución del programa 6
7. Detección de errores 6
   7.1. Error por fichero inexistente 6
   Figura 4. Error al encontrar fichero de datos. 6
   7.2. Error por no proporcionar dirección de archivo 6
   Figura 5. Error al no proporcionar la dirección de archivo. 6

1. Descripción

ParallelFischerScoring_binomial es un software que permite estimar los parámetros de un modelo de Regresión Logística Binaria a través del algoritmo de Fischer Scoring mediante procesamiento heterogéneo CPU/GPU.

Este manual no solo enseña las funcionalidades del software, sino que también evalúa su capacidad de estimación y convergencia para un modelo de regresión logística binaria de dos variables de la forma $Y = 1 + 2X_1 + 3X_2$ estimando los parámetros (Betas), de tal forma que la variable dicotómica (Y) es explicada por medio de las dos variables (X1, X2). Se realizan pruebas utilizando CPU y GPU comparando el tiempo de respuesta para diferentes cantidades de datos multivariados simulados de las variables antes mencionadas, simulados entre los 10.000 a los 30.000 datos.

2. Disponibilidad

El código fuente y la documentación de encuentran disponibles en el siguiente repositorio:

https://github.com/Parall-UD/ParallelFischerScoring_binomial

3. Requisitos

Para el funcionamiento de este software, es necesario el siguiente hardware:

1. Tarjeta gráfica NVIDIA

También es necesario el siguiente software:

1. Python 2.7: https://www.python.org/download/releases/2.7/
2. Numpy 1.14.2: https://pypi.org/project/numpy/
3. Pycuda 2017.1.1: https://pypi.org/project/pycuda/
4. Skcuda 0.5.2: https://scikit-cuda.readthedocs.io/en/latest/install.html
6. Scikit-image 0.14.2: https://pypi.org/project/scikit-image/

4. Iniciar aplicación

La ejecución de la aplicación presenta dos opciones de ejecución, la primera desde una versión de CPU, la segunda desde una versión de GPU. Para iniciar la aplicación, ubique la carpeta en la cual usted clono el repositorio a través de la consola.

Ahora, utilice el siguiente comando para realizar la ejecución de la aplicación en CPU:

```
python fisher_cpu.py 'parametro_de_entrada'
```

Y el siguiente comando para realizar la ejecución de la aplicación en GPU:

```
python fisher_gpu.py 'parametro_de_entrada'
```

A continuación, se describen los parámetros y sugerencias:

- **Parámetro_de_entrada:** String con la ruta absoluta del archivo .csv que contiene los datos, el archivo .csv debe estar compuesto de 4 columnas que representan:
    - **y:** Respuesta dicotómica (0-1)
    - **r:** Es la columna asociada al parámetro de intercepto (B0)
    - **x1:** Conjunto de datos aleatorios en una distribución normal
    - **x2:** Conjunto de datos aleatorios en una distribución normal
- **Salida:** Esta compuesta de 3 partes:
    - Numero de Iteraciones
    - Lista de Betas obtenidos
    - Tiempo en escala de segundos de cuanto se demoró la ejecución en la aplicación
- Si tiene más de una versión de Python se recomienda especificar que se usara la versión de Python 2.7 de la siguiente manera:

```
Python2.7 fisher_gpu.py 'parametro_de_entrada'
```

5. Realización de Pruebas

Las pruebas se realizaron implementando una instancia de este software en un servidor del Centro de Cómputo de Alto Desempeño de la Universidad Distrital (CECAD). Las características de este servidor se encuentran en la Tabla 1.

Tabla 1. Características del Servidor.

Procesador	Intel (R) Xeon (R) CPU E-52697 v3 @ 2.60GHZ
Tarjeta Grafica	NVIDIA Tesla K80
Memoria RAM	128Gb

Al realizar la ejecución de la aplicación se ejecutará un comando como el siguiente:

```
Python2.7 fisher_cpu.py '/home/nvera/andres/fisher_scoring/datos/datos_fisher20.csv'
```

En este comando se pueden evidenciar los parámetros anteriormente descritos haciendo uso del script en CPU.

5.1. Descripción datos de entrada.

A continuación, se muestra una descripción general de los datos de entrada, estos deben estar contenidos dentro de un archivo .csv, la figura 1 muestra un ejemplo de la distribución de los datos:

y	r	x1	x2
1	1	-33.100.529.062.	202.200.761.832
1	1	-520.444.850.66	104.346.919.549
0	1	-990.007.802.90	-35.438.163.000.
1	1	293.363.238.354	6.159.660.201.2
1	1	-373.090.961.38	255.164.894.823
0	1	-148.051.921.94	-212.040.535.86
0	1	-167.630.273.90	-31.352.544.089.
1	1	-158.470.048.16	151.018.838.329
0	1	-122.392.877.89	313.229.021.228
1	1	526.041.722.795	511.035.867.294
0	1	-286.936.938.29	-448.636.895.52
1	1	615.752.899.788	12.365.306.711.3
1	1	162.236.503.468	-833.941.846.28
0	1	-129.070.619.75	-8.386.241.779.6
1	1	5.097.258.335.8	188.573.811.506
1	1	-644.672.328.38	-25.069.121.468.
1	1	962.706.711.203	978.867.014.193
0	1	16.950.500.628.8	-12.677.571.505.
1	1	-13.708.708.885.	5.181.418.042.69
1	1	-10.198.317.852.	481.734.828.416
1	1	14.936.854.846.2	296.162.366.190
0	1	655.211.661.790	-188.250.663.85
1	1	-47.697.880.201.	100.505.640.362

Figura 1. Datos de entrada al algoritmo Fisher Scoring

Como se observa los datos de entrada están repartidos en 4 columnas que serán explicadas a continuación:

- **Y:** Variable Dicotómica que es explicada por las variables x1, x2, también es la variable de la probabilidad de la función binaria su rango es de [0-1]
- **r:** Variable auxiliar que multiplica al termino independiente del modelo binario de tal forma que $Y = \boxed{1} + 2X_1 + 3X_2$ pueda cumplirse.
- **x1, x2:** Valores aleatorios dentro de una distribución normal que explican la función dicotómica.

Las pruebas realizadas se hicieron con tamaños de 10.000, 15.000, 20.000, 25.000 y 30.000 datos correspondientemente.

6. Muestra de resultados

La siguiente figura presenta los resultados obtenidos de tiempo de respuesta en segundos al realizar la prueba del algoritmo de Fisher Scoring simulado en CPU y en GPU con datos de entrada de distintos tamaños de 10.000, 15.000, 20.000, 25.000 y 30.000 datos correspondientemente y así comparar la efectividad de tiempo entre ambos métodos:

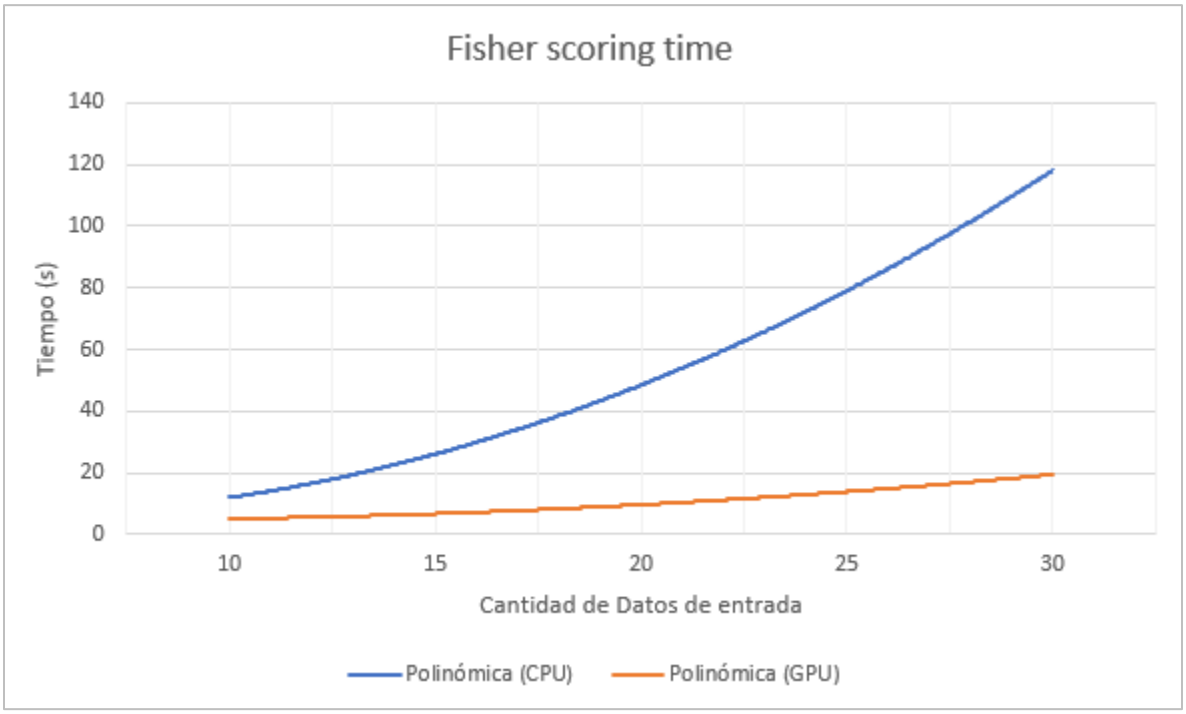

Figura 2. Comparación resultados temporales entre CPU y GPU. (Datos de entrada en miles)

A partir de la gráfica anterior, se puede evidenciar que tanto el tiempo de procesamiento en GPU como CPU aumentan conforme aumenta el tamaño de la cantidad de datos, pero los tiempos en GPU no varían significativamente, como si lo hacen los tiempos en CPU, por lo tanto, se puede determinar que tras realizar la implementación del algoritmo de Fisher Scoring en un contexto de computación heterogénea CPU/GPU se optimizan los tiempos de respuesta. A continuación, la Tabla 2 presenta los valores consolidados en la gráfica anterior, con el fin de entender el comportamiento plasmado:

Tabla 2. Tiempos de ejecución en CPU y GPU

Tamaño (# de entradas)	Tiempo (S)		Speed-Up
	CPU	GPU	
10000	11,89	4,88	2,23x
15000	28,34	7,65	3,70x
20000	41,71	7,89	5,28x
25000	85,78	15,22	5,63x
30000	115,8	18,99	6,09x

Se puede observar en la última columna el orden de velocidad que es obtenida al ejecutar el algoritmo mediante la interfaz gráfica con respecto a la ejecución desde el procesador. Los resultados obtenidos al ejecutar el programa se ven reflejados en la figura 3, donde se muestran 3 factores importantes:

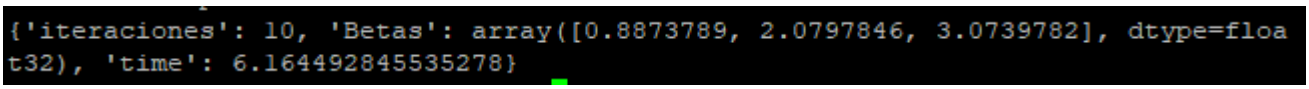

Figura 3. Resultados ejecución del programa

Los resultados indican:

- Numero de iteraciones realizados para la estimación de parámetros.
- Lista de Betas estimados, estos deben ser aproximados a 1, 2 y 3 respectivamente para indicar que se ha realizado una estimación correcta.
- Tiempo de ejecución que estará indicada en segundos.

7. Detección de errores

La detección de errores hace referencia a aquellas situaciones en las que las funcionalidades del sistema no se ejecutarán si no se cumplen con ciertas condiciones.

7.1. Error por fichero inexistente

La figura 4 describe una ejecución fallida del programa debido a que no encontró el fichero de datos desde la dirección que fue proporcionada:

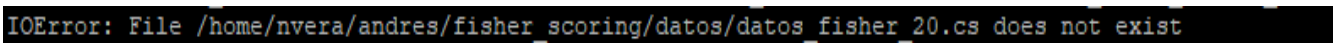

Figura 4. Error al encontrar fichero de datos.

Esto puede ocurrir si se proporciona mal la dirección donde se encuentra el archivo .csv o no existe el fichero en la dirección escrita.

7.2. Error por no proporcionar dirección de archivo

La figura 5 describe una ejecución fallida del programa debido a que el usuario ejecuto la aplicación sin proporcionarle una dirección del fichero que contiene los datos:

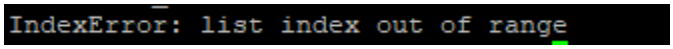

Figura 5. Error al no proporcionar la dirección de archivo.