Universidad Nacional de Ingeniería

Facultad de ciencias

Introducción a la estadística y probabilidades

Análisis probabilístico de algoritmos y uso de algoritmos aleatorios para determinar y mejorar la eficiencia en los procesos de búsqueda de resultados esperados en diferentes escenarios

Autores:
Daniel Hidalgo
Miguel Oviedo
Jhon Stuart
Nelson Sanabio

Supervisor:
Cesar Lara

1. Resumen

Los algoritmos aleatorios son muy útiles para evaluar el rendimiento de un algoritmo al querer obtener el peor de los casos posibles en determinados eventos. En el presente proyecto se implementarán diferentes algoritmos aleatorios para observar y concluir con datos el beneficio de estos al querer obtener un resultado esperado con determinada eficiencia. Además, se hará análisis y comparación de algoritmos (en algunos casos) de forma matematizada como en el caso de los algoritmos de Las Vegas y Monte Carlo donde se observará la diferencia en cuanto a la obtención de casos favorables y desfavorables dependiendo el algoritmo, como también el beneficio obtenido para ambos y sus respectivas eficiencias. Además veremos el algoritmo UCB, que supera todas las limitaciones de las estrategias basadas en la exploración y el compromiso, incluida la necesidad de conocer el horizonte y las brechas de suboptimidad.

También estan los árboles que son estructuras de datos que pueden organizar elementos con ciertas características básicas como elementos padre, hijos, antescesores o sucesores. Además se pueden implementar variar operaciones como INSERCIÓN, ELIMINACIÓN, BÚSQUEDA, etc. Presentan una gran cantidad de variantes y su uso dependen en gran medida de la aplicación que se necesite. En este apartado presentamos en concreto la inserción aleatoria de elementos en los árboles de búsqueda binaria que es uno de los árboles más básicos. Analizamos la esperanza matemática de dichos elementos equiprobables asi como una implementación de su código en R.

Un Bloom Filter consiste de un vector A de tama˜no n, A[0] a A[n-1], inicializados en 0. Un Bloom Filter usa k funciones hash h1, h2, ..., hk con un rango de 0,...,n1. Una vez definido esto, comencemos con el análisis matemático

2. Introducción

Los algoritmos aleatorios que usaremos nos mostrarán lo siguiente:

- Comparación de los algoritmos **Las Vegas** y **Monte Carlo** en el juego Find the lady, donde se obtendrá la cantidad de jugadas requeridas para obtener determinada cantidad de victorias y el dinero que nos queda luego de hacer las jugadas.
- Un árbol de búsqueda binaria es una estructura de datos para contener datos ordenados en forma de aŕbol binario. Un árbol binario está vacío o consta de un nodo raíz que contiene un elemento y apunta a los subárboles izquierdo y derecho. La propiedad fundamental en árboles binarios es que tanto en un árbol como un subárbol, todos los elementos del subárbol izquierdo son menores que el elemento raíz, mientras que todos los elementos del subárbol derecho son mayores que que el elemento raíz. Esta distribución implica que podemos buscar un elemento particular haciendo

una búsqueda binaria: si un elemento no está en la raíz, podemos recurrir en el subárbol izquierdo o derecho dependiendo de si es más pequeño o más grande que el elemento de la raíz.

- Bloom Filters es una estructura de datos compacta para representar probabilísticamente un conjunto para admitir consultas de la presencia de elementos sobre un conjunto, es decir, se verificará si un elemento X se encuentra en el conjunto Y.
- Analizar la independencia como tambien la dependencia de las instancias.

2.1. Estado del arte

En los siguientes artículos citados veremos como se analiza y determina la eficiencia de los algoritmos de Las Vegas y Monte Carlos:

`https://www2.cs.duke.edu/courses/spring16/compsci330/Notes/LVMC.pdf`

`https://www2.cs.duke.edu/courses/fall15/compsci532/scribe_notes/lec10.pdf`

Para Bloom Filters:

`http://www-db.disi.unibo.it/~fgrandi/papers/IPL2017_accepted.pdf`

3. Implementación en R

Para la evaluación y comparación de los algoritmos Las Vegas y Monte Carlo se hará la implementación de ambos algoritmos en R para el juego *Find the lady*. Un juego muy conocido donde el jugador debe escoger entre 3 cartas, si la carta escogida es una reina de corazones ganará 2 dolares, en caso contrario perderá 1 dolar. Para esto, se usará el objeto llamado "my"donde se almacenará variables, como las cartas a ser usadas (una reina de corazones y dos jockers), el dinero obtenido luego de una partida o una ronda de muchas partidas, las veces que jugó por ronda, la cantidad de dinero obtenido al final y por ronda, las victorias en determinada cantidad de partidas como máximo hasta ganar una cantidad de veces finita, las veces que se ganó y la cantidad de rondas.

Al final, se mostrará una gráfica donde se visualizarán las veces jugadas versus el dinero obtenido para ambos algoritmos, también se mostrará la media del dinero obtenido luego de todas las jugadas y la varianza entre los resultados al final de cada ronda. Es interesante observar como es que apesar de que la esperanza es 0 (asumiendo que cada partida es justa, es decir, no hay cartas truncada o la existencia de alguna clase de trampa) se puede obtener una media

cercana a este, también se puede ver como la varianza llega a estar en un rango entre 45-55.

En el caso de Bloom Filters, consideremos un conjunto $A = \{a_1, a_2, ..., a_n\}$ de n elementos. Bloom filters describe información de pertenencia de A utilizando un vector V de tamaño m. Para esto, tenemos k funciones hash, $h_1, h_2, ..., h_k$ con $h_i : X \to \{1..m\}$, se usan como se describe a continuación: El siguiente procedimiento crea un Bloom filters de m bits, que corresponde a un conjunto de A y utiliza las k funciones hash $h_1, h_2, ..., h_k$

En consecuencia, si a_i es está en A, en el Bloom filter V resultante todos los bits obtenidos correspondientes a los valores hasheados de a_i su valor es 1. La prueba de pertenenecia a un elemento $elem$ es equivalente a probar que todos los bits correspondientes de V son el conjunto.

4. Pruebas

Para el juego de Find the lady el lector puede ejecutar varias veces el código otorgado en el cuadernillo de jupyter se podrá visualizar y comprobar dichos datos referidos en la implementación, la media y la varianza por otro lado nos dará un resultado muy curioso y en un rango determinado. Este código está modulado por partes para que pueda ejecutar cada prueba por separado, además de unas pruebas iniciales para luego entrar más a fondo en el código.

5. Conclusiones

- Al comparar los algoritmos de Las Vegas y Monte Carlo se puede verificar que en el caso de Monte Carlo no se llega a obtener la cantidad de victorias deseadas, pero la cantidad de veces jugadas es menor a comparación del otro algoritmo. Esto debido a que en el algoritmo de Las Vegas siempre se va a querer obtener el resultado deseado sin importar cuánto nos tome de tiempo para lograrlo. Para nuestro objeto de estudio se puede concluir que con Monte Carlo se puede tener una aproximación acerca de cuánto podremos obtener y cuantas jugadas requeriremos a lo menos para determinada cantidad de victorias que se pida como requerimiento.
- Para el experimento con Quicksort, según los cálculos obtenidos de diferentes pruebas no hay un punto de comparación computaciones para un rango de hasta cien mil elementos debido que la potencia de cómputo de una computadora éstandar es mayor al igual como en el algoritmo de quicksort, pero matemáticamente se demuestra que la inserción de elementos aleatorizados tiene tiempo computacional aceptable.

- Definimos cada instancia en la cual nos muestra las políticas simples y eficientes para el problema del bandido (un problema particular) en cualquier conjunto de distribuciones de recompensas con soporte limitado conocido con arrepentimiento logarítmico uniforme.
- La cantidad de funciones hash del Bloom Filter, solo depende de la probabilidad del falso positivo p_{err}. Además se puede a diferenciar falso positivos y falsos negativos y que existe estructuras de datos más eficiente que las tablas hash. Dicha estructura de datos se puede usar para grandes conjuntos de datos (BigData), como lo usan actualmente Google Chrome, Etherium, Squid Web, entre otros.

Referencias

[1] James Aspnes,Notes on Randomized Algorithms,CPSC 469/569: Fall 2016, 2018, pp. 1-10, 44-53, 103-132

[2] Peter Auer, Nicolo Cesa-Bianchi y Paul Fischer. Análisis en tiempo definido del problema de bandido múltiple , (2002), pp. 124-129.

[3] Karl Stratos, C.C. (2016).IEOR 8100-001, UCB Algorithm, Worst-Case Regret Bound. Learning and Optimization for Sequential Decision Making

[4] Mitzenmacher, M., Upfal, E. (2017). Probability and computing: Randomized and probabilistic techniques in algorithms and data analysis. Cambridge; New York ; Port Melbourne ; Delhi ; Singapore: Cambridge University Press.

[5] Kirsch, Adam; Mitzenmacher, Michael (2006), "Less Hashing, Same Performance: Building a Better Bloom Filter", in Azar, Yossi; Erlebach, Thomas, Algorithms – ESA 2006, 14th Annual European Symposium , Lecture Notes in Computer Science, 4168, Springer-Verlag, Lecture Notes in Computer Science 4168, pp. 456–467, doi:10.1007/11841036, ISBN 978-3-540-38875-3, archived from the original on 2009-01-31

[6] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest and Clifford Stein. Introduction to Algorithms. The MIT Press, Cambridge, Massachusetts, 2009.

[7] PKS Prakash, Achyutuni Sri Krishna Rao R Data Structures and Algorithms. Packt Publishing, 2016