La relación entre algoritmos aleatorizados de tipo Montecarlo y Las Vegas

Bernardo Subercaseaux

18 de junio de 2020

Primero recordemos las definiciones.

- **Algoritmo tipo Montecarlo:** Son algoritmos que pueden *equivocarse*, pero que garantizan tardar un cierto tiempo. Nos interesa que la probabilidad de error sea baja.
- **Algoritmos tipo Las Vegas:** No pueden equivocarse, pero el tiempo que tardan no está determinado. Nos interesa que en valor esperado tarden poco tiempo.

Para afirmar las cosas, consideremos un ejemplo de algoritmo de tipo Las Vegas. El problema consiste en recibir una permutación p de los números del 1 al n, un número $1 \leq x \leq n$, y retornar en qué posición se encuentra x en p. Por ejemplo, si $p = 2, 4, 5, 3, 1$ y $x = 4$, la respuesta es 1, puesto que $p[1] = 4$. El algoritmo más simple es probar posiciones al azar hasta dar con el elemento buscado.

```
def find_position_LV(p, x):
  while True:
    i = random.randint(0, len(p)-1) # probabilidad uniforme
    if p[i] == x:
      return i
```

Listing 1: Algoritmo tipo Las Vegas para problema de búsqueda

Dado que el elemento buscado siempre está presente, el algoritmo no puede fallar, se trata pues de un algoritmo de tipo Las Vegas. El problema es que no sabemos cuánto se demorará. Intentemos calcular el tiempo esperado en que termina el algoritmo. Dado que el algoritmo tarda tiempo constante en cada iteración, y que el elemento x está en alguna posición k, desconocida, lo que queremos calcular es cuánto tardaremos en que la

posición al azar elegida entre 0 y $n-1$ sea exactamente k. Sea X la variable aleatoria que describe la cantidad de iteraciones que realiza el algoritmo, o equivalentemente, la cantidad de veces que hay que elegir un elemento al azar entre n posibilidades, uniforme e independientemente, hasta obtener una posibilidad posibilidad en particular.

$$\begin{aligned} E(X) &= \sum_{i=1}^{\infty} i \cdot Pr(X = i) && \text{por definición} \\ &= \sum_{i=1}^{\infty} i \cdot \left(\frac{n-1}{n}\right)^{i-1} \cdot \frac{1}{n} && \text{hay } i-1 \text{ fallos, y un acierto} \\ &= \frac{1}{n} \sum_{i=1}^{\infty} i \cdot \left(\frac{n-1}{n}\right)^{i-1} \end{aligned}$$

Definamos la variable $r = \frac{n-1}{n}$, y notemos que $\frac{1}{n} = 1 - r$. Entonces tenemos que

$$\begin{aligned} \text{Última expresión} &= (1-r) \sum_{i=1}^{\infty} i \cdot r^{i-1} \\ &= (1-r) \sum_{i=1}^{\infty} \frac{\mathrm{d}}{\mathrm{d}r} r^i && \text{puede que usar una derivada parezca raro, confiemos...} \\ &= (1-r) \frac{\mathrm{d}}{\mathrm{d}r} \left(\sum_{i=1}^{\infty} r^i \right) \end{aligned}$$

Notando que $|r| < 1$, y confiando[1] en que eso implica $\sum_{i=1}^{\infty} r^i = \frac{r}{1-r}$, obtenemos

$$\begin{aligned} \text{Última expresión} &= (1-r) \frac{\mathrm{d}}{\mathrm{d}r} \left(\frac{r}{1-r} \right) \\ &= \frac{1}{1-r} = n. \end{aligned}$$

Hemos concluido que el tiempo esperado del algoritmo es n, lo mismo que tarda en el peor caso probar cada posición una a una!

La siguiente pregunta que nos podemos hacer es si existe un algoritmo de tipo Montecarlo para este problema. La respuesta es que sí. Si probamos

[1] Esto se deduce de tomar la expresión de una suma geométrica y tomar el límite al infinito.

k posiciones al azar, sabemos que nos demoraremos tiempo k, la pregunta es cuál será la probabilidad de error.

```
def find_position_MC(p, x, k):
  for _ in range(k):
    i = random.randint(0, len(p)-1) # probabilidad uniforme
    if p[i] == x:
      return i
  return 0
```

Listing 2: Algoritmo tipo Montecarlo para problema de búsqueda

La probabilidad de error es la probabilidad de que ninguno de los k intentos de con la posición buscada. Esto tiene probabilidad $(1 - \frac{1}{n})^k$. Por ejemplo, si $n = 1000$ y $k = 700$, la probabilidad de error es de menos del 50 %.

Resulta que podemos generalizar esta idea, y obtener un algoritmo de tipo Montecarlo a partir de cualquier algoritmo de tipo Las Vegas.

Teorema 1. *Dado un algoritmo de tipo Las Vegas, que termina en tiempo esperado $T(n)$, y una constante $c > 0$, siempre se puede construir un algoritmo de tipo Montecarlo que tarda tiempo $cT(n)$ y cuya probabilidad de error es a lo más $1/c$.*

Antes de probar el teorema, probaremos una desigualdad muy importante en el análisis de algoritmos aleatorizados, la desigualdad de Markov. Vimos antes que nuestro algoritmo de tipo Las Vegas tardaba tiempo esperado n, pero en algunos casos podría demorar mucho más. Una pregunta interesante es cuál es la probabilidad de caer en uno de esos *casos malos*. Por ejemplo, cuál es la probabilidad de que nuestro algoritmo de tipo Las Vegas se demore más de n^2. Lo que nos interesa, entonces, es calcular cuál es la probabilidad de que una variable aleatoria no negativa (como es el tiempo de ejecución) sea mucho mayor que su esperanza. Esto es justamente lo que acota la desigualdad de Markov.

Teorema 2 (Desigualdad de Markov)**.** *Sea X una variable aleatoria no negativa, y $c > 0$ una constante real. Entonces se cumple que $Pr\big(X \geq cE(X)\big) \leq 1/c$.*

Demostración. Consideremos el caso en que el recorrido de la variable aleatoria es finito, para no meternos con integrales. Esto nos sirve dado que nuestros casos de interés son discretos. Es decir la variable aleatoria X toma valores en un conjunto finito $\Omega \subseteq R_0^+$. Primero probaremos que para

cualquier constante $a > 0$ se cumple que $Pr(X \geq a) \leq \frac{E(X)}{a}$. En efecto

$$\begin{aligned}
E(X) &= \sum_{r \in \Omega} r \cdot Pr(X = r) \\
&= \sum_{r \in \Omega, r > a} r \cdot Pr(X = r) + \sum_{r \in \Omega, r \leq a} r \cdot Pr(X = r) \\
&\geq \sum_{r \in \Omega, r \leq a} r \cdot Pr(X = r) \\
&\geq \sum_{r \in \Omega, r \leq a} a \cdot Pr(X = r) \\
&= a \cdot Pr(X \leq a)
\end{aligned}$$

Habiendo probado que para cualquier $a > 0$ se tiene que $Pr(X \geq a) \leq \frac{E(X)}{a}$, podemos tomar $a = cE(x) > 0$, con lo que obtenemos el resultado. □

Bueno, hemos demostrado la desigualdad de Markov, ahora la podemos utilizar para demostrar lo que realmente queríamos. Recordemos entonces el enunciado del Teorema 1, puesto que ya tenemos todas las herramientas para demostrarlo.

Teorema 1. *Dado un algoritmo de tipo Las Vegas, que termina en tiempo esperado $T(n)$, y una constante $c > 0$, siempre se puede construir un algoritmo de tipo Montecarlo que tarda tiempo $cT(n)$ y cuya probabilidad de error es a lo más $1/c$.*

Demostración. Consideremos que A es el algoritmo de tipo Las Vegas cuyo tiempo esperado es $T(n)$. Definamos B como el algoritmo que llama a A, pero tomando su tiempo. Si A se pasa de tiempo $cT(n)$, entonces B retorna cualquier cosa. Si A termina antes de $cT(n)$, entonces dado que A no se equivoca, B retorna lo mismo que retorna A, sin lugar a errores. Con esto garantizamos que B tarda a lo más $cT(n)$, siempre, y que su probabilidad de error es a lo más la probabilidad de que A tarde más de $cT(n)$. Sabemos entonces que B es un algoritmo de tipo Montecarlo, pero cuál es su probabilidad de error? Bueno, dado que A tiene tiempo esperado $T(n)$, y usando la desigualdad de Markov sabemos que la probabilidad de que el tiempo de A exceda de $cT(n)$, y por tanto B se pueda equivocar, es a lo más $1/c$. Con esto sabemos que la probabilidad de error de B es a lo más $1/c$, con lo que terminamos la demostración. □

Pregunta para reflexionar[2]: Cuándo podemos convertir un algoritmo de tipo Montecarlo en uno de tipo Las Vegas?

[2]It really makes you think.