Concentración de Medida en Algoritmos Aleatorizados.

Nicolás Rivera

23 de Junio de 2011

Pontificia Universidad Católica de Chile

Índice

Índice

Índice

¿Qué es la Concentración de Medida?
¿Ley Grandes Números, Teorema Límite Central, algo más?

Vamos a introducir la desigualdad de Chernoff que nos será útil en el análisis de algoritmos. Para ellos recordemos la desigualdad de Markov.

Teorema

Sea X una variable aleatoria no negativa y sea $a > 0$, entonces

$$\mathbb{P}(X > a) = \frac{\mathbb{E}(X)}{a}.$$

Sean $X_i, i = 1..n$ variables aleatorias independientes y sea $X = \sum_{i=1}^{n} X_i$. Consideremos la función generadora de momentos de X_i esta es

$$f_X(t) = \mathbb{E}(\exp(tX)), t \in \mathbb{R},$$

Así, por desigualdad de Markov, se tiene que

$$\mathbb{P}(X > m) = \mathbb{P}(\exp(tX) > \exp(tm)) \leq \frac{\mathbb{E}(\exp(tX))}{\exp(tm)}, t \geq 0$$

Supongamos que todas las variables X_i tenen la misma distribución, entonces

$$\mathbb{E}(\exp(tX)) = \mathbb{E}(\prod_{i=1}^{n} \exp(tX_i)) = \mathbb{E}(\exp(tX_1))^n,$$

luego

$$\mathbb{P}(X > m) \leq \frac{\mathbb{E}(\exp(tX_1))^n}{\exp(tm)},$$

donde sale la desigualdad

Teorema

$$\mathbb{P}(X > m) \leq \min_{t \in \mathbb{R}} \frac{\mathbb{E}(\exp(tX_1))^n}{\exp(tm)},$$

La técnica anterior permite encontrar muchas cotas dependiendo del tipo de variables aleatorias que consideremos. Consideremos X_i variables aleatorias Bernoulli donde $X_i = 1$ con probabilidad p y 0 con probabilidad $q = 1 - p$. Entonces

$$\mathbb{E}(\exp(tX_1)) = p\exp(t) + q,$$

luego el teorema anterior queda

$$\mathbb{P}(X > m) \leq \min_{t\in\mathbb{R}} \frac{(p\exp(t) + q)^n}{\exp(tm)},$$

Tomando $m = (p + x)n$ y minimizando se obtiene

$$\mathbb{P}(X > (p+x)n) \leq \exp(-(p+x)\log(\frac{p+x}{p}) - (q-x)log(\frac{q-x}{q}))^n,$$

Supongamos que $X_i \in [0,1]$. Es fácil ver la función $f(x) = \exp(tx)$, $t \geq 0$ en $x \in [0,1]$ queda bajo la recta que pasa por los puntos $(0,1)$ y $(1, \exp(t))$. Dicha recta es $y = \alpha x + \beta$ con $\beta = 1$ y $\alpha = \exp(t) - 1$. Así

$$\mathbb{E}(\exp(tX_i)) \leq \mathbb{E}(\alpha X_i + \beta) = \alpha \mathbb{E}(X_i) + \beta = \exp(t)\mathbb{E}(X_i) + 1 - \mathbb{E}(X_i),$$

con lo anterior podemos sacar cotas para casos generales y no solo para X_i con distribución Bernuolli.

Teorema

Sea $X = \sum_{i=1}^{n} X_i$ *donde* X_i, $i = 1..n$ *son independientes e identicamente distribuidas en* $[0, 1]$, *entonces*

- *Para todo* $t > 0$,

$$\mathbb{P}(X > \mathbb{E}(X) + t) = \mathbb{P}(X < \mathbb{E}(X) - t) \leq \exp(\frac{-2t^2}{n}).$$

- *Para todo* $\varepsilon > 0$,

$$\mathbb{P}(X > (1 + \varepsilon)\mathbb{E}(X)) \leq \exp(-\frac{\varepsilon^2 \mathbb{E}(X)}{3}).$$

- *Para todo* $0 < \varepsilon < 1$,

$$\mathbb{P}(X < (1 - \varepsilon)\mathbb{E}(X)) \leq \exp(-\frac{\varepsilon^2 \mathbb{E}(X)}{2})$$

Aplicaciones

1. Amplificación de Probabilidad

Sea f un algoritmo de computa x y $f(x)$ entrega la respuesta correcta probabilidad $p > 3/4$. Supongamos, por simplicidad que $f(x) \in \{0, 1\}$. Supongamos que ejecutamos este algoritmo n veces y sea X el valor más repetido y sea Y el número de veces que aparece la respuesta correcta, así el valor X es la respuesta incorrecta si $Y < \frac{n}{2}$. Asumamos, por simplicidad que la respuesta correcta es 1. Sabemos que $\mathbb{E}(Y) = pn > \frac{3}{4}n$. Sea R_i la respuesta en la $i-$esima ejecución del algoritmo, entonces definimos

$$Y_i = 1_{R_i=1},$$

luego $Y = \sum_{i=1}^{n} Y_i$.

Claramente todos los Y_i son i.i.d. y $\mathbb{E}(Y_1) = \mathbb{P}(R_i = 1) = p$, luego

$$\mathbb{P}(X = 0) = \mathbb{P}(Y < \frac{n}{2}) = \mathbb{P}(Y < pn - (p - \frac{1}{2})n)$$

$$\leq \exp(-2(p - \frac{1}{2})^2 n) \leq \exp(-\frac{n}{8}),$$

Más o menos.. $\exp(-\frac{50}{8}) = 0{,}0019$.

2. Quicksort

Sea $X = [x_1, x_2, ..., x_n]$ un arreglo con $x_i \in \mathbb{R}$. Recordemos que Quicksort selecciona un pivote $p \in [n]$ y luego considera el arreglo

$$[Y, x_p, Z] = [y_1, ..., y_j, x_p, z_1, ..., z_k],$$

donde $y_i \leq x_p$ para todo $i \in [j]$ y $z_i > x_p$ para todo $i \in [k]$ y actuamos recursivamente en Y y en Z. Notemos que estas llamadas recursivas empiezan a formar un árbol binario. Notemos que crear la recursión toma tiempo $O(n)$ (asumiento que elegir el pivote toma tiempo O(1)). Por lo tanto el número de acciones totales al realizar el algoritmo es $O(Hn)$ donde H es la altura del árbol creado. En el mejor caso se obtiene un árbol binario balancedo con altura a lo más $\log_2(n)$ y por lo tanto el algoritmo es super rápido.

Veamos que ocurre si elegimos el pivote de manera aleatoria. Consideremos un arreglo de números de largo n y sea $T(n)$ el número de acciones que ejecuta QuickSort eligiendo los pivotes de forma aleatoria, es decir $T(n)$ es variable aleatoria. Es claro que

$$T(n) = n - 1 + T(X) + T(n - 1 - X),$$

donde X es el tamaño de alguno de los arreglos que quedan despues de elegir el pivote. Es fácil ver que $\mathbb{P}(X = i) = \frac{1}{n}$ con $i = 0, 1, ..., n - 1$. Ahora tomamos esperanza y

$$\mathbb{E}(T(n)) = n - 1 + 2\sum_{i=0}^{n-1} \mathbb{E}(T(i)),$$

Llamando $f(n) = \mathbb{E}(T(n))$ la ecuación anterior se escribe como una recurrencia y se puede obtener (por inducción) que

$$\mathbb{E}(T(n)) \leq 2nlog(n)$$

Notemos que si elegimos el pivote de manera aleatoria generamos un árbol (de la manera no aleatoria también se genera un árbol). Se puede probar usando técnicas de Análisis Amortizado que el tiempo de ejecución de Quicksort la suma de las profundidades de cada nodo del árbol.

Probaremos que la probabilidad de encontrar un vértice del árbol con profundidad $24\ln(n)$ es muy baja. Sea $\mathcal{P}$ el conjunto de todos los caminos que unen un vértice con la raíz del árbol, sea v un vértice cuya y profundidad es $24\ln n$ y sea $P \in \mathcal{P}$ el camino que une v con la raíz. Para cada $i \in P$ sea X_i la variable aleatoria que toma valor 1 si el pivote x_i estaba en la mitad del arreglo que particionaba, osea entre el primer y 3er cuarto del arreglo. Si el arreglo en el vértice v tiene más de un elemento (y por lo tanto debemos usar v como pivote y actuar recursivamente) entonces, si:

$$\alpha = \alpha_P = \sum_{i \in P} X_i,$$

se tiene

$$(\frac{3}{4})^{\alpha} > 1,$$

Aplicando logaritmo se tiene $\alpha < 3\ln(n)$. Se concluye que si hay más de $3\ln(n)$ de nodos que pivotean en entre el primer y el tercer cuarto del arreglo, entonces el camino formado por esos pivotes no excede los $24\log(n)$. Notemos que $\mathbb{P}(X_i = 1) = \frac{1}{2}$, luego $\mathbb{E}(\alpha) = 12\lg n$. Finalmente sea $P \in \mathcal{P}$, entonces

$$\mathbb{P}(|P| > 24\ln n) \leq \mathbb{P}(\alpha_P < 3\ln n),$$

usando Chernoff

$$\mathbb{P}(\alpha_P < 3\ln n) = \mathbb{P}(\alpha_P < (1-\frac{3}{4})12\ln n) \leq \exp(-\frac{9(12\ln n)}{32}) \leq \frac{1}{n^3},$$

Finalmente, se sigue que

$$\mathbb{P}(\exists P \in \mathcal{P}, |P| > 24\lg n) \leq n\mathbb{P}(|P| > 24\ln n) \leq \frac{1}{n^2}$$

También se puede demostrar que con probabilidad $1 - \frac{1}{n}$ ningun nodo del árbol tiene tamaño $4 \lg n$.

Randomized Rounding

Sea $G = (V, E)$ un grafo no dirigido. Sea D un conjunto de pares de vértices, es decir, $D = \{(s_i, t_i), i = 1..n\}$. Sea $C > 0$. El problema consiste encontrar caminos $\{P_i\}$ tal que P_i es un camino entre s_i y t_i y tal que ningúna arista $e \in E$ aparezca en más de C caminos.

Un caso particular es elegir los caminos $\{P_i\}$ tal que $C = \text{máx}_{e \in E}\, cong(e)$ sea mínimo, dond $cong(e)$ es la cantidad de caminos P_i que contienen a e. El problema es NP-hard (obvio). C es la congestión del grafo.

Para atacar el problema escribamos el problema a lo programación entera.

Para expresarlo como programación entera sea $\mathcal{P}_i$ todos los caminos que unen s_i con t_i, f_P^i la variable indicatriz que nos dice si elegimos $P \in \mathcal{P}_i$ o no. Sea C la congestión del grafo.
Minimizar C sujeto a

- Exactamente un camino que une s_i a t_i.

$$\sum_{P \in \mathcal{P}_i} f_P^i = 1 \forall i$$

- La congestión en cada arista es menor que C

$$\sum_i \sum_{P : e \in P} f_P^i \leq C, \forall e$$

-

$$f_P^i \in \{0, 1\}, \forall i, P$$

Oviamente el programa anterior es NP-completo, pero podemos relajar la condición $f_P^i \in \{0,1\}$ a $f_P^i \in [0,1]$. Transformando el programa Entero a uno Lineal. Sea C^* la solución relajada, entonces es claro que $C^* \leq C$ (C es la solución del programa Entero). La idea ahora es utilizar que $f_P^i \in [0,1]$ y $\sum_{P \in \mathcal{P}_i} f_P^i = 1$, es decir, son una distribución de probabilidad sobre P_i. Así, ahora, para cada i elegimos $P_i \in \mathcal{P}_i$ con probabilidad $f_{P_i}^i$.

Analizemos la congestión en $e \in E$. Notemos que la probabilidad de que el camino P_i contenga a e es

$$\mathbb{P}(e \in P_i) = \sum_{P \in \mathcal{P}_i, e \in P} f_P^i,$$

Sea $X_i^e = 1$ si $e \in P_i$, 0 en otro caso, luego la congestión en e es $X^e = \sum_i X_i^e$. luego

$$\mathbb{E}(X_i^e) = \sum_{P \in \mathcal{P}_i, e \in P} f_P^i,$$

se sigue

$$\mathbb{E}(X^e) = \mathbb{E}(\sum_i X_i^e) = \sum_i \mathbb{E}(X_i^e) = \sum_i \sum_{P \in \mathcal{P}_i, e \in P} f_i^p \leq C^* \leq C$$

Sea $k = 1 + \varepsilon$, luego la idea es tomar k para que $\mathbb{P}(X^e > kC^*) \leq \frac{1}{n^3}$, luego al unir sobre todas las aristas tenemos que elegir aleatoriamente es una $k-1$-aproximación. Se sabe que si $C >> \ln(n)$ entonces la aproximación es buena, si C es pequeño la aproximación es logaritmica.

Teorema de Karp

Supongamos que tenemos un algoritmo aleatorizado que toma un input X de tamaño x realiza un trabajo que $a(x)$ operaciones y produce un subproblema de tamaño $H(x)$ que puede ser resuelto recursivamente, Sea $T(x)$ el tiempo que se demora el algoritmo en realizar el trabajo sobre el input X, entonces se tiene que

$$T(x) = a(x) + T(H(x)),$$

Supongamos que $H(x)$ es aleatorio. Veamos que podemos hacer.

En general Encontrar cotas para $E(H(x))$ no es tan difícil. Supongamos que

$$0 \leq E(H(x)) \leq m(x) \leq x,$$

para alguna función m. Consideremos la versión determinística del problema, es decir

$$u(x) = a(x) + u(m(x)),$$

Cuya solución es $u(x) = \sum_{i \leq 0} a(m^i(x))$, donde $m^0(x) = x$, $m^{i+1}(x) = m(m^i(x))$.

Teorema (Primer Teorema de Karp)

Supongamos que $\mathbb{E}(H(x)) \leq m(x)$ *con* $0 \leq m(x) \leq x$ *y* $a(x)$ *continua y estrictamente creciente en* $\{x|a(x) > 0\}$ *y* , $m(x)$ *continua, entonces para todo real* x *y toda instancia de tamaño* x *y todo* t *positivo se tiene que*

$$\mathbb{P}(T(x) > u(x) + ta(x)) \leq \left(\frac{m(x)}{x}\right)^t .$$

Generalicemos lo anterior, supongamos ahora que nuestro algoritmo toma un input de tamaño x realiza un trabajo que toma $a(x)$ operaciones y luego divide el problema en k problemas de tamaño $H_i(x)$, $(i = 1..k)$. La recurrencia ahora queda como

$$T(x) = a(x) + \sum_{i=1}^{k} T(H_i(x)),$$

Para lo anterior se tiene lo siguiente

Teorema (Segundo Teorema de Karp)

Suponga que, para todo x y todos los posibles valores $(y_1, ..., y_k)$ de la tupla $(H_1(x), ..., H_k(x))$ se tiene que $\mathbb{E}(T(x)) \geq \sum_{i=1}^{k} \mathbb{E}(T(H_i(x)))$, entonces para todo t positivo,

$$\mathbb{P}(T(x) \geq (t+1)\mathbb{E}(T(z)) < \exp(-t),$$

k-ésimo Estadístico de Orden

Consideremos un conjunto S de n números distintos y queremos encontrar el k-ésimo elemento más pequeño de S, Una algoritmo posible es elegir un elemento $r \in S$ al azar y luego comparando cada elemento de $S \setminus r$ con r crear dos conjuntos L y U donde $L = \{y \in S : y < r\}$ y $U = \{y \in S : y > r\}$, luego si:

- $|L| \geq k$, entonces recursivamente encontrar el k-ésimo más pequeño en L;
- $|L| = k - 1$ retornar r;
- $|L| < k - 1$, entonces recursivamente, encontrar el $k - 1 - |L|$-ésimo más pequeño en U.

Es fácil ver que particionar requiere $n-1$ comparaciones. Se puede probar que $\mathbb{E}(H(x)) \leq \frac{3x}{4}$, y luego $u(x) \leq 4x$. Sea $T(n,k)$ el número de comparaciones que realiza el algoritmo en un arreglo de tamaño n para encontrar el k-ésimo estadístico de orden, así

$$\mathbb{P}(T(n,k) > 4n + t(n-1)) \leq (\frac{3}{4})^t,$$

Hint: $(\frac{3}{4})^{20} = 0{,}003$

QuickSort

Ya vimos que

$$T(n) = n - 1 + T(X) + T(n - 1 - X),$$

por lo tanto es claro que $\mathbb{E}(T(n)) \geq \mathbb{E}(T(X)) + \mathbb{E}(T(n-1-X))$ y vimos que $\mathbb{E}(T(n)) < 2n \ln n$,se sigue que

$$\mathbb{P}(T(n) \geq (t+1)2n \ln n) \leq \mathbb{P}(T(n) \geq (t+1)\mathbb{E}(T(n))) \leq \exp(-t),$$

Hint: $\exp(-6) = 0{,}0024$.

Haciendo un análisis local se puede encontrar una mejor cota. Supongamos que tenemos un elemento v en el arreglo que queremos ordenar, entonces contemos cuantas veces debemos comparar v con un pivote. Llamemos C_v a esta cantidad, entonces la cantidad de comparaciones de QuickSort es $\sum_v C_v$. Sea $C_v(n)$ el número de veces que comparamos v con un pivote al usar QuickSort en un arreglo de largo n, entonces

$$C_v(n) = 1 + C_v(H(n)),$$

donde $H(l)$ es la lista en donde queda v al separar l respecto al pivote.

Luego,

$$\mathbb{E}(C_v(n)) = 1 + \frac{1}{n-1}\sum_{i=1}^{n-1}\mathbb{E}(C_v(m)) \leq \ln(n),$$

luego por el primer teorema de Karp se tiene

$$\mathbb{P}(C_v(n) \geq (1+t)\ln(n)) \leq (\frac{\ln(n)}{n})^t,$$

finalmente

$$\mathbb{P}(T(n) \geq (1+t)n\ln(n)) \leq \mathbb{P}(\exists v, C_v(n) \geq (1+t)\ln(n)) \leq n(\frac{\ln(n)}{n})^t,$$

Esto funciona con $t > 1$. Hint: $n = 1000, t = 2 \Rightarrow 0,008$, Hint2: $n = 13, t = 5 \Rightarrow 0,003$.