Algoritmos, Azar y Autómatas
Resolución Ejercicios 4 a 6

Manuel Panichelli

October 19, 2021

Resultados previos

Lema 1 (Pequeño truco de límites)**.** *Relación entre lim inf / lim sup y lim*
Sean $(x_{1,n})_{n\geq 0}, (x_{2,n})_{n\geq 0}, \dots (x_{k,n})_{n\geq 0}$ una secuencia de números reales tales que $\sum_{i=1}^{k} x_{i,n} = 1$ y $c_1, c_2, \dots c_k$ numeros reales tales que $\sum_{i=1}^{k} c_i = 1$. Luego,

1. $\forall i \ \liminf_{n\to\infty} x_{i,n} \geq c_i \implies \forall i \ \lim_{n\to\infty} x_{i,n} = c_i$
2. $\forall i \ \limsup_{n\to\infty} x_{i,n} \leq c_i \implies \forall i \ \lim_{n\to\infty} x_{i,n} = c_i$

Vamos a aplicar esto para $k = b^\ell$, $x_{n,i} = \frac{\|x[1\dots n]\|_{w_i}}{b^\ell}$ con $i = 1, 2, \dots b^\ell$.
Notar que

$$\sum_{i=1}^{b^\ell} \frac{\|x[1\dots n]\|_{w_i}}{b^\ell} = 1,$$

ya que es la suma de las frecuencias, donde

- n = # palabras de ℓ caracteres (n posiciones que quiero mirar)
- w_i = palabras de longitud ℓ.

0.1 Bad

Def. 1 (Malas palabras)**.**

$$\text{Bad}(A, k, w, \varepsilon) = \left\{ v \in A^k : \left| |v|_w - \frac{k}{b^{|w|}} \right| > \varepsilon k \right\}$$

Lema 2 (Hardy & Wright, Theorem 148)**.** Sean $b \geq 2$ y k enteros positivos. Si $6/k \leq \varepsilon \leq 1/b$ entonces para cada símbolo d en A,

$$|Bad(A, k, d, \varepsilon)| < 4e^{-b\varepsilon^2 k/6} b^k.$$

0.2 Turing

Prop. 1. Sea E_n el conjunto de candidatos del algoritmo de Turing,

Para cada n, E_n es una unión finita de intervalos abiertos con bordes racionales, y para $n \geq n_0$, $\mu E_n > 1 - 1/N_n^2$.

Corolario. $\bigcap_{n\geq 0} E_n$ tiene medida positiva.

Prop. 2 (Medida de intersección)**.**

$$\mu(A \cap B) = \mu(A \setminus B^c) \geq \mu(A) - \mu(B^c) = \mu(A) - (1 - \mu(B))$$

Ejercicio 4 - Piatetski-Shapiro

Demostrar la versión de ocurrencias alineadas de Piatetski-Shapiro

Teorema 1 (Piatetski-Shapiro)**.** Sea x un número real, $b \geq 2$ un entero y $A = \{0, \ldots, b-1\}$. Las siguientes son equivalentes

1. x es normal a base b
2. Existe una constante C tal que para infinitas longitudes ℓ y para todo $w \in A^\ell$

$$\limsup_{n\to\infty} \frac{|x[1\ldots n]|_w}{n} < C \cdot b^{-\ell}.$$

3. Existe una constante C tal que para infinitas longitudes ℓ y para todo $w \in A^\ell$

$$\limsup_{n\to\infty} \frac{\|x[1\ldots n\ell]\|_w}{n} < C \cdot b^{-\ell}.$$

Lema 3. Para todo $w \in A^*, n$ y k, contar la cantidad de apariciones de w en $x[1\ldots n\ell]$ es lo mismo que partir x en bloques de longitud ℓk alineados y contar dentro de cada uno.

$$\|x[1\ldots n\ell k]\|_w = \sum_{v\in A^{\ell k}} \|x[1\ldots n\ell k]\|_v \|v\|_w$$

No puede suceder que aparezca de forma alineada un w en el medio de dos bloques v, ya que como los bloques son de tamaño ℓk están alineados con las ocurrencias de w.

Def. 2. Defino Bad', un análogo a Bad pero para ocurrencias alineadas.

$$\text{Bad'}(A, k, w, \varepsilon) = \left\{ v \in A^k : \left| \|v\|_w - \frac{k}{b^{|w|}} \right| > \varepsilon k \right\}$$

Lema 4. Sea v una *buena palabra*, es decir $v \in A^{\ell k} \setminus \text{Bad'}(A^\ell, k, w, \varepsilon)$. Se que la frecuencia de apariciones alineadas de w en v se parece mucho al esperado, k/b^ℓ. Por lo tanto, puedo multiplicar la cantidad de apariciones por un factor de achique pequeño y seguirá valiendo:

$$\|v\|_w \geq \frac{k}{b^\ell}(1-\varepsilon) \Leftrightarrow \frac{\|v\|_w}{k} \geq (1-\varepsilon)b^{-\ell}$$

Dem. de Piatetski-Shapiro. Voy a demostrar que 3. $\Rightarrow$ 1.

Sea ℓ una de las infinitas longitudes que cumple Piatetski Shapiro,

$$\limsup_{n\to\infty} \frac{\|x[1\ldots n\ell]\|_w}{n} < C \cdot b^{-\ell}. \tag{1}$$

Fijo $\varepsilon \leq 1/b^\ell, w \in A^\ell$ y sea k suficientemente grande tal que

$$|\text{Bad'}(A^\ell, k, w, \varepsilon)| < \varepsilon b^{\ell k} \tag{2}$$

Tengo que (con Bad' = Bad'$(A^\ell, k, w, \varepsilon)$)

$$\begin{aligned}
\liminf_{n\to\infty} \frac{\|x[1\ldots n\ell k]\|_w}{nk} &= \sum_{v\in A^{\ell k}} \overbrace{\frac{\|x[1\ldots n\ell k]\|_v}{n}\frac{\|v\|_w}{k}}^{C} && \text{(Lema 3)}\\
&= \liminf_{n\to\infty} \sum_{v\in A^{\ell k}\setminus \text{Bad'}} C + \sum_{v\in \text{Bad'}} C && \text{(Total = bad + good)}\\
&\geq \liminf_{n\to\infty} \sum_{v\in A^{\ell k}\setminus \text{Bad'}} C\\
&= \liminf_{n\to\infty} \sum_{v\in A^{\ell k}\setminus \text{Bad'}} \frac{\|x[1\ldots n\ell k]\|_v}{n}\frac{\|v\|_w}{k}\\
&\geq (1-\varepsilon)b^{-\ell} \liminf_{n\to\infty} \sum_{v\in A^{\ell k}\setminus \text{Bad'}} \frac{\|x[1\ldots n\ell k]\|_v}{n} && \text{(Lema 4)}\\
&= (1-\varepsilon)b^{-\ell} \liminf_{n\to\infty} \left(1 - \sum_{v\in \text{Bad'}} \frac{\|x[1\ldots n\ell k]\|_v}{n}\right) && \text{(freq bad + freq good = 1)}\\
&\geq (1-\varepsilon)b^{-\ell} \left(1 - \sum_{v\in \text{Bad'}} \limsup_{n\to\infty} \frac{\|x[1\ldots n\ell k]\|_v}{n}\right)\\
&> (1-\varepsilon)b^{-\ell} \left(1 - \sum_{v\in \text{Bad'}} C\cdot b^{-\ell k}\right) && (1)\\
&\geq (1-\varepsilon)b^{-\ell} \left(1 - \varepsilon b^\ell C b^{-\ell k}\right) && (2)\\
&= (1-\varepsilon)b^{-\ell}(1 - C\varepsilon)
\end{aligned}$$

Y como esto vale para todo $\varepsilon \leq 1/b^\ell$ positivo, puedo tomar uno suficientemente chico como para que $(1-\varepsilon)$ y $(1-C\varepsilon)$ se acerquen a 1, y de esa forma

$$\liminf_{n\to\infty} \frac{\|x[1\dots n\ell k]\|_w}{nk} \geq b^{-\ell}.$$

Como esto vale para toda $w \in A^\ell$, por el Lema 1

$$\lim_{n\to\infty} \frac{\|x[1\dots n\ell k]\|_w}{nk} = b^{-\ell}.$$

Tomando $n' = nk$, tengo

$$\lim_{n'\to\infty} \frac{\|x[1\dots n'\ell]\|_w}{n'} = b^{-\ell}.$$

$\therefore$ x es normal. □

Ejercicio 5 - Turing

Adaptar el algoritmo de Turing para que genere un número normal en base 2 y 3, y que lo exprese en base 3 en vez de base 2. Demostrar su correctitud.

Obs. Un número real x es normal en base 2 y 3 si es simplemente normal en todas las bases 2^i y 3^i para $i \in \mathbb{N}$.

Def (Discrepancia simple)**.** Sea $x \in [0,1]$ un real en el intervalo unitario y x_b su expansión en base b. Definimos su **discrepancia simple** para algún N tamaño de segmento inicial de la siguiente forma,

$$\Delta_N(x_b) = \max_{d\in\{0,\dots,b-1\}} \left| \frac{|x_b[1\dots N]|_d}{N} - \frac{1}{b} \right|.$$

x será simplemente normal en base b si,

$$\lim_{N\to\infty} \Delta_N(x_b) = 0.$$

Def (Pasos del algoritmo)**.** Usamos n como el paso del algoritmo y definimos las sig funciones,

> $\boldsymbol{N_n} = 2^{n_0+2n}$, el número de dígitos vistos en el paso n, donde $n_0 = 11$ (n_0 *solo está ahí para simplificar las cuentas*)
>
> $\boldsymbol{b_n} = \lfloor \log N_n \rfloor$ es la base más grande considerada en el paso n
>
> $\boldsymbol{\varepsilon_n} = 1/b_n$ es la diferencia entre la frecuencia esperada de dígitos y la frecuencia actual en el paso n. El *nivel de tolerancia*, dependiente de la cantidad de bases que se están viendo.

$b_n \geq 2$ y es no decreciente y no acotada, N_n es no decreciente y no acotada, y ε_n es no creciente y va a 0.

Def (Conjuntos de candidatos)**.** Definimos los conjuntos de números reales

$$E_0 = (0,1)$$
$$E_n = \bigcap_{b \in B} \{x \in (0,1) : \Delta_{N_n}(x_b) < \varepsilon_n\}$$

Donde B es el conjunto de bases consideradas,

$$B = \{b \mid b \in \{2, \ldots, b_n\} \text{ y } \exists\, i \in \mathbb{N} : b = 3^i \text{ o } b = 2^i\}$$

Para cada n, el conjunto E_n consiste de todos los números reales cuyas expansiones en las bases $2, 3, 4, 9, \ldots, b_n$ exhiben **buenas frecuencias** de digitos hasta ε_n, en los primeros N_n digitos (en el segmento inicial de tamaño N_n)

El algoritmo en sí es el siguiente, con output $y_1 y_2 y_3 \ldots$

Paso inicial, $n = 0$. $I_0 = (0,1),\ E_0 = (0,1)$

Paso recursivo, $n > 1$

En el paso anterior computamos I_{n-1}. Sean I_n^0, I_n^1 y I_n^2 el primer, segundo y tercer tercio de I_{n-1} respectivamente.

- Si $\mu\left(I_n^0 \cap \bigcap_{j=0}^n E_j\right) > 1/N_n$ entonces $I_n = I_n^0$ y $y_n = 0$.
- Si $\mu\left(I_n^1 \cap \bigcap_{j=0}^n E_j\right) > 1/N_n$ entonces $I_n = I_n^1$ y $y_n = 1$.
- Sino, $I_n = I_n^2$ y $y_n = 2$.

μA es la medida de Lebesgue de A.

Demostración de correctitud de Turing adaptado

La estrategia para demostrar la correctitud del algoritmo consiste en mostrar que,

- $\bigcap_{n \geq 0} E_n$ tiene medida positiva y consiste de números normales en base 2 y 3.
- El algoritmo genera $x = 0.y_1 y_2 y_3 \ldots \in \bigcap_{n \geq 1} I_n$
- El algoritmo preserva el invariante $\mu\left(I_n \cap \bigcap_{j=1}^n E_j\right) > 0$ (hay números)
- En el infinito,

$$\bigcap_{n \geq 1} I_n = \bigcap_{n \geq 0} \left(I_n \cap \bigcap_{j=1}^{n} E_j \right)$$

Esto es así pues $(I_n)_{n \geq 0}$ es anotado y se cumple el invariante $\mu(I_n \cap \bigcap_{j=0}^n E_j)$

Juntando todo, el algoritmo genera x, y

$$x \in \bigcap_{n\geq 1} I_n = \bigcap_{n\geq 0} \left(I_n \cap \bigcap_{j=1}^{n} E_j \right) \subset \bigcap_{n\geq 0} E_n$$

Luego, como todos los elementos de la intersección de candidatos son normales en base 2 y 3, el punto $x = 0.y_1y_2y_3\ldots$ que pertenece a ese conjunto es normal en base 2 y 3.

Veamos las demostraciones pendientes de los puntos,

Lema. El conjunto $\bigcap_{n\geq 0} E_n$ tiene medida positiva y consiste solo números normales en base 2 y 3.

Proof. Veamos que tiene medida positiva y que son normales en base 2 y 3.

- Como el conjunto E_n del algoritmo de Turing original

$$E_n = \bigcap_{b\in\{2,\ldots,b_n\}} \{x \in (0,1) : \Delta_{N_n}(x_b) < \varepsilon_n\}$$

  tiene medida positiva (Prop 1), y este considera menos bases para la intersección (solo múltiplos de 2 y 3 en vez de todas), va a tener más elementos, y por lo tanto

$$E'_n \supset E_n \Rightarrow \mu E'_n > \mu E_n > 0 \Rightarrow \mu E'_n > 0.$$

- Quiero ver que consiste de números normales en base 2 y 3.

  Sea $x \in \bigcap_{n\geq 0} E_n$. Luego, para todo n, $x \in E_n$, entonces para $b \in B$

$$\Delta_{N_n}(x_b) \leq \varepsilon_n.$$

  Sea b una base y M una posición. Tomo n tal que,

$$N_n \leq M < N_{n+1}.$$

  Como

$$\Delta_N(x_b) = \max_{d\in\{0,\ldots,b-1\}} \left| \frac{|x_b[1\ldots N]|_d}{N} - \frac{1}{b} \right|,$$

  Tengo que para todo d

$$\begin{aligned} \Delta_N(x_b) < \varepsilon_n &\Leftrightarrow \frac{|x_b[1\ldots N]|_d}{N} - \frac{1}{b} < \varepsilon_n \\ &\Leftrightarrow |x_b[1\ldots N]|_d < \left(\varepsilon_n + \frac{1}{b}\right) N \end{aligned}$$

Para cada b mas chico que b_n tenemos que para todo dígito $d \in \{0, \ldots, b-1\}$

$$\begin{aligned}\frac{|x_b[1 \ldots M]|_d}{M} &\leq \frac{|x_b[1 \ldots N_{n+1}]|_d}{N_n} \\ &\leq \frac{N_{n+1}}{N_n}\left(\varepsilon_{n+1} + \frac{1}{b}\right) \\ &= \frac{2^{n_0+2n+2}}{2^{n_0+2n}}\left(\varepsilon_{n+1} + \frac{1}{b}\right) \\ &= 4\left(\varepsilon_{n+1} + \frac{1}{b}\right)\end{aligned}$$

Luego para cada base $b \in B$,

$$\limsup_{n\to\infty} \frac{|x_b[1 \ldots N]|_b}{N} < 4b^{-1}$$

Y para cada base $b \in B$ y cada digito en base b,

$$\limsup_{n\to\infty} \frac{|x_b[1 \ldots N]|_b}{N} < 4b^{-\ell}$$

Por lo tanto, por Piatetski-Shapiro x es simplemente normal en toda base $b \in B$, por lo que es normal en bases 2 y 3.

□

Lema (Invariante)**.** La siguiente condición es invariante en cada paso n del algoritmo,

$$\mu\left(I_n \cap \bigcap_{j=1}^{n} E_j\right) > 0.$$

Proof. Lo demostramos por inducción sobre n. Recordando que $N_n = 2^{n_0+2n}$

- CB ($n = 0$)

  Tenemos que

  $$\mu\left(I_0 \cap E_0\right) = \mu\left((0,1)\right) = 1 > \frac{1}{N_0^2} = \frac{1}{2^{2n_0}}$$

- PI ($n \Rightarrow n+1$)

  Tenemos como HI que vale para n,

$$\mu\left(I_n \cap \bigcap_{j=0}^{n} E_j\right) > \frac{1}{N_n}.$$

Veamos que vale para $n+1$. Por la Prop. 1 sabemos que $\mu E_n > 1 - 1/N_n^2$, luego

$$\begin{aligned}
\mu\left(I_n \cap \bigcap_{j=0}^{n+1} E_j\right) &= \mu\left((I_n \cap \bigcap_{j=0}^{n} E_j) \cap E_{n+1}\right) \\
&\geq \mu\left(I_n \cap \bigcap_{j=0}^{n} E_j\right) - \mu\left(E_{n+1}^c\right) && \text{(Prop 2)} \\
&> \frac{1}{N_n} - \frac{1}{N_{n+1}^2} && \text{(HI y Prop. 1)} \\
&> \frac{2}{N_{n+1}}
\end{aligned}$$

Y finalmente, el algoritmo elije I_{n+1} entre I_n^0, I_n^1 e I_n^2 asegurando que

$$\mu\left(I_{n+1} \cap \bigcap_{j=0}^{n+1} E_j\right) > \frac{1}{N_{n+1}}.$$

□

Ejercicio 6 - BHS

Adaptar el algoritmo BHS para generar un número normal en base 2 y 3 (en vez de uno absolutamente normal) y expresarlo en su expansión fraccionaria en base 3 (en vez de base 2).

Def (b-ario)**.** Decimos que un intervalo es b-ario (o b-ádico) de orden n si tiene la forma

$$\left(\frac{a}{b^n}, \frac{a+1}{b^n}\right)$$

para algun entero a tal que $0 \leq a < b^n$.

Si σ_b y τ_b son intervalos b-arios y $\tau_b \subseteq \sigma_b$ decimos que el *orden relativo* de τ_b con respecto a σ_b es el orden de τ_b menos el orden de σ_b.

Lema 5 (Lemma 3.1 BHS 2013)**.** Sean u y v bloques y ε un real positivo,

1. Si $\Delta(u) < \varepsilon$ y $\Delta(v) < \varepsilon$ entonces $\Delta(uv) < \varepsilon$.

   Si u es buena y v también, entonces su concatenación lo es. Esto me dice que si cuento dígito, puedo considerar la suma de las proporciones y va a andar todo bien.

2. Si $\Delta(u) < \varepsilon, v = a_1 \dots a_{|v|}$ y $|v|/|u| < \varepsilon$ entonces $\Delta(vu) < 2\varepsilon$ y para cada ℓ tal que $1 \leq \ell \leq |v|, \Delta(ua_1a_2 \dots a_l) < 2\varepsilon$.

   $|v|/|u| < \varepsilon \Leftrightarrow |v| < \varepsilon|u|$, y como ε es un número chiquito entre 0 y 1, se interpreta como que u es muy chica en comparación a v.

   Concatenar una palabrita mucho mas chica que una buena al lado de una buena a lo sumo empeora el doble la discrepancia

   Para el algoritmo, nos dice que lo que ya viste no hace falta volverlo a mirar, podés mirar cosas nuevas. Si tengo un buen segmento inicial y lo que elijo para agregar es cortito, a lo sumo llevo mi discrepancia al doble.

Lema 6 (Lemma 3.4 BHS2013)**.** Para cualquier intervalo I y cualquier base b, hay un subintervalo b-ario J tal que $\mu J \geq \mu I/(2b)$.

Podemos encontrar J grande y b-ádico que se parezca mucho en medida

Def (t-sequences)**.** Una t-sequence $\vec{\sigma}$ es una secuencia de intervalos $(\sigma_2, \dots, \sigma_t)$ tales que

- Para cada base $b = 2, \dots, t$, σ_b es b-ario.
- Para cada base $b = 3, \dots, t$, $\sigma_b \subset \sigma_{b-1}$ y $\mu\sigma_b \geq \mu\sigma_{b-1}/(2b)$.

Son intervalos anidados, con la medida lo más grande posible.

La definición implica que $\mu\sigma_t \geq (\mu\sigma_2)/(2^t t!)$ *(es un peor caso, no siempre ocurre y a veces calza justo)*

Def (Refinamiento)**.** Dos definiciones,

- Una t-sequence $\vec{\tau} = (\tau_2, \dots, \tau_t)$ *refina* una t'-secuencia $\vec{\sigma} = (\sigma_2, \dots, \sigma_{t'})$ si $t' \leq t$ y $\tau_b \subset \sigma_b$ para cada $b = 2, \dots, t'$.
- Un refinamiento tiene *discrepancia* menor a ε si para cada $b = 2, \dots, t'$ hay palabras u, v tales que $\sigma_b = I_u, \tau_b = I_{uv}$ y $\Delta(v) < \varepsilon$.

El τ que elegiste tiene una discrepancia $< \varepsilon$.

Lema 7. Sean $t \geq 2$ un entero, $t' = t \vee t + 1$, $\varepsilon < 1/t$ un real positivo. Luego, toda t-sequence $\vec{\sigma} = (\sigma_2, \dots, \sigma_t)$ admite un refinamiento $\vec{\tau} = (\tau_2, \dots, tau_{t'})$ con discrepancia menor a ε.

El orden relativo de τ_2 puede ser cualquier entero k tal que

$$k \geq max \left\{ \frac{6}{\varepsilon}, \frac{24(\log_2 t)(\log(t!))}{\varepsilon^2} \right\}.$$

Def (Pasos del algoritmo)**.** El algoritmo considera las siguientes funciones para un paso n dado,

$\boldsymbol{t_n} = \max(2, \lfloor \sqrt[4]{\log n} \rfloor)$ es la máxima base a considerar en el paso n,

$\boldsymbol{\varepsilon_n} = 1/t_n$ es la discrepancia máxima tolerada en el paso n, y

$\boldsymbol{N_n} = \lfloor \log_3 n \rfloor + n_{\text{start}}$ es el número de dígitos en base 3 agregados en el paso n,

Donde n_{start} es el minimo entero que valida la condición del Lema 7. Por lo tanto, requerimos que para todo $n > 0$,

$$\lfloor \log_3 n \rfloor + n_{\text{start}} \geq max \left\{ \frac{6}{\varepsilon}, \frac{24(\log_2 t)(\log(t!))}{\varepsilon^2} \right\}.$$

Algoritmo

Tiene de output $x = y_1 y_2 y_3 \dots$ que son los símbolos en la expansión en base 3 de un número normal en base 2 y 3.

Paso inicial ($\boldsymbol{n = 1}$)

$\vec{\sigma}_1 = (\sigma_3)$, con $\sigma_3 = (0, 1)$. Observo que $(0, 1)$ es triádico, pues

$$\sigma_3 = \left(\frac{0}{3^0}, \frac{1}{3^0} \right).$$

Paso recursivo ($\boldsymbol{n > 1}$)

En el paso anterior computamos $\vec{\sigma}_{n-1} = (\sigma_3, \sigma_2, \dots, \sigma_{t_n - 1})$.

Tomamos $\sigma_n = (\tau_3, \tau_2, \dots, \tau_{t_n})$ la t_n-sequence más a la izquierda que refina $\vec{\sigma}_{n-1}$ con discrepancia menor que ε_n tal que si $\sigma_3 = I_u$ entonces $\tau_3 = I_{uv}$ con $|v| = N_n$.

Ponemos $y_{M_n+1} \dots y_{M_n+N_n} = v$ donde $M_n = \sum_{j=1}^{n} N_j$.

El algoritmo construye $\vec{\sigma}_0, \vec{\sigma}_1, \vec{\sigma}_2, \dots$ tal que $\vec{\sigma}_0 = (0, 1)$ y para cada $n \geq 1$, $\vec{\sigma}_n$ es una t_n-sequencia que refina $\vec{\sigma}_{n-1}$ con discrepancia ε_n y tal que el orden de $\sigma_{n,2}$ es N_n mas el orden de $\sigma_{n-1,2}$.

Genera un número real x en base 3 que pertenece a la intersección de todos los intervalos, por lo que es simplemente normal en toda base $b \in B$, y es normal en base 2 y 3.