Squares vs. rectangles: which ones are heavier?

Jérôme Plût

2013-12-03

Weighing squares and rectangles

Which one of these is heavier?

- Comparing areas is difficult!
- We compare the **Hamming weight** of their areas instead.
- We pick random squares and rectangles of size 2^N.
- We compare squares and rectangles to lines (random numbers of size 2^{2N}, with expected Hamming weight N).

Divide and conquer

On closer look, squares and rectangles have both a **big end** (top half) and a **small end** (bottom half):

With boring numbers instead:

$$\underbrace{011011}_{\text{big end}}\underbrace{000001}_{\text{small end}}$$

This work is politically correct and inclusive. In particular, we respect all kinds of mathematics, and shall do analysis both in the real numbers (big end) and the 2-adic numbers (small end).

On the small end

2-adic squares

A 2-adic number is a square iff it is of the form

$$(\ldots\ldots)001\underbrace{00\ldots00}_{\text{even}}.$$

with geometric distribution.

- The expected weight of the lower half of a square is $-3/2$ bits.
- The expected weight of the lower half of a rectangle is $-1/2$ bits.
- **The lower half of squares is lighter by 1 bit.**

On the big end

Let $x \in [0, 2^N[$.

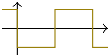
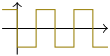
- The N bits on the big end of x^2 are the N first bits of $(x/2^N)^2$.
- We know the density f of $(x/2^N)^2$ in the interval $[0, 1[$:

$$f(t)dt = \frac{dt}{2\sqrt{t}}.$$

- We compute the average Hamming weight of a random number with density f.

\def\more{more}Divide \more, conquer \more

- The Hamming weight S_n of $t \in [0,1]$ is the sum of the Hamming weight W_i of individual bits.
- The functions W_i are periodic with period 2^{-i}:

W_1 : , W_2 : 

- The expected value of W_i is

$$\overline{W}_i = \int_0^1 W_i(t) f(t) dt = \langle W_i, f \rangle_{L^2}.$$

- We can compute this scalar product using Fourier series decomposition!

How to get rid of @ll your annoying c0nstants 1n one easy step

- We compute the Fourier coefficients of W_i and f on $[0,1]$.
- Since we are lazy, we compute only W_1 and then $W_i(x) = W_1(2^{i-1}x)$.
- Since we are lazy, we compute only the sine coefficients of f:

$$b_m(f) \;=\; 2\int_0^1 \sin(2\pi mt)\frac{dt}{2\sqrt{t}} \;=\; \frac{2}{\sqrt{2\pi m}}\int_0^{\sqrt{2\pi m}} \sin(t^2)\,dt.$$

How to get rid of @ll your annoying c0nstants 1n one easy step

- We compute the Fourier coefficients of W_i and f on $[0, 1]$.
- Since we are lazy, we compute only W_1 and then $W_i(x) = W_1(2^{i-1}x)$.
- Since we are lazy, we compute only the sine coefficients of f:

$$b_m(f) \;=\; 2\int_0^1 \sin(2\pi m t)\frac{dt}{2\sqrt{t}} \;=\; \frac{\cancel{2}}{\sqrt{\cancel{2}\pi m}} \int_0^{\sqrt{2\pi m}} \sin(t^2)\,dt.$$

How to get rid of @ll your annoying c0nstants 1n one easy step

- We compute the Fourier coefficients of W_i and f on $[0,1]$.
- Since we are lazy, we compute only W_1 and then $W_i(x) = W_1(2^{i-1}x)$.
- Since we are lazy, we compute only the sine coefficients of f:

$$b_m(f) \; = \; 2\int_0^1 \sin(2\pi mt)\frac{dt}{2\sqrt{t}} \; = \; \frac{\cancel{2}}{\sqrt{\cancel{2\pi} m}} \int_0^{\sqrt{2\pi m}} \cancel{\sin}(t^2)\,dt.$$

How to get rid of @ll your annoying c0nstants 1n one easy step

- We compute the Fourier coefficients of W_i and f on $[0, 1]$.
- Since we are lazy, we compute only W_1 and then $W_i(x) = W_1(2^{i-1}x)$.
- Since we are lazy, we compute only the sine coefficients of f:

$$b_m(f) \;=\; 2\int_0^1 \sin(2\pi m t)\frac{dt}{2\sqrt{t}} \;=\; \frac{\cancel{2}}{\sqrt{\cancel{2}\cancel{\pi} m}} \int_0^{\sqrt{2\pi m}} \sin(t^2)\,dt.$$

How to get rid of @ll your annoying c0nstants 1n one easy step

- We compute the Fourier coefficients of W_i and f on $[0, 1]$.
- Since we are lazy, we compute only W_1 and then $W_i(x) = W_1(2^{i-1}x)$.
- Since we are lazy, we compute only the sine coefficients of f:

$$b_m(f) \;=\; 2\int_0^1 \sin(2\pi m t)\frac{dt}{2\sqrt{t}} \;=\; \frac{2}{\sqrt{2\pi m}}\int_0^{\sqrt{2\pi m}} \sin(t^2)\,dt.$$

- We get:

$$b_m(f) \sim \frac{1}{\sqrt{m}}.$$

Boringness is the cardinalest sing

- Sum the bits to compute the approximation for the Hamming weight of the first n bits on the big end:

$$S_n^{\mathrm{sqr}} = \underbrace{-1.587239\text{4631649104}\text{531239363}\ldots}_{\text{(gluttony)}} + \underbrace{\frac{\sqrt{2}+3}{2\pi}\,\zeta\left(\tfrac{3}{2}\right)}_{\text{(pride)}} 2^{-n/2} + \underbrace{\ldots}_{\text{(sloth)}}$$

Boringness is the cardinalest sing

- Sum the bits to compute the approximation for the Hamming weight of the first n bits on the big end:

$$S_n^{\mathrm{sqr}} = \underbrace{-1.587239{\scriptstyle 4631649104}{\scriptscriptstyle 531239363\ldots}}_{\text{(gluttony)}} + \underbrace{\frac{\sqrt{2}+3}{2\pi}\zeta\left(\tfrac{3}{2}\right)}_{\text{(pride)}} 2^{-n/2} + \underbrace{\ldots}_{\text{(sloth)}}$$

- We may perform the same computations for rectangles...

$$S_n^{\mathrm{mul}} = -1.7289433\underbrace{\ldots}_{\text{(envy)}} + \frac{\log 2}{2} \cdot n \cdot 2^{-n} + O(2^{-n}).$$

- **The high half of squares is heavier by about 0.15 bit.**

I forgot to put an introduction, so here it is

- [Amiel, Feix, Tunstall, Whelan, Marnane 2008] observed that squares tended to be about 1 bit lighter than rectangles.
- We wanted to determine the speed of convergence for increasing values of n.
- We find that the average difference between the Hamming weight of a square and a product, as $n \to \infty$, is 0.8492962 bits.
- So the actual speed of convergence to 1 bit is **extremely slow**.

ಧನ್ಯವಾದ!