Computation in Big Spaces

John Skilling (john@skilling.co.uk) AI in Astronomy 2019

The 20th century got a lot wrong.

1. It argued about Bayes.
2. It viewed functions through 19th century lens.
3. It viewed inference through Laplace transform.
4. It thought dimensionality was hard.

Result: Principles of inference were obscured.
Easy stuff was judged impossible.

It's 2019: we can do better!

Bayes

Fact: the only consistent calculus for uncertainty is ordinary probability.

Sum rule: proportions (probabilities) add up.
Product rule: proportions (probabilities) multiply. ✓✓

$$\text{Prob}(\theta)\ \text{Prob}(\text{Data} \mid \theta) \ = \ \text{Prob}(\text{Data})\ \text{Prob}(\theta \mid \text{Data})$$

Prior × Likelihood ⟶ Evidence × Posterior

↑ Question ↓ Explanatory? ↓ Answer

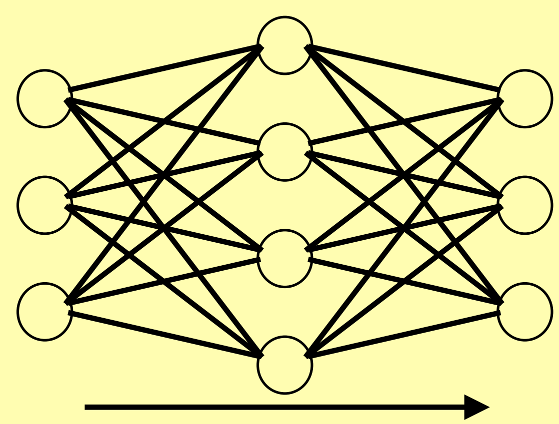
THE framework of inference

Prior: *What's your world model?* (human crafted, deep-learned, . . .)

OLD: World $= \sum \int \begin{pmatrix} \text{modified Hankel functions} \\ \text{of 3rd type} \ldots\ldots \text{etc.} \end{pmatrix}$ (parameters θ)

NEW: Parameters θ

Architecture



World ✓✓

Computational Inference

$$\text{Prob}(\theta)\,\text{Prob}(\text{Data} \mid \theta) \;=\; \text{Prob}(\text{Data})\,\text{Prob}(\theta \mid \text{Data})$$

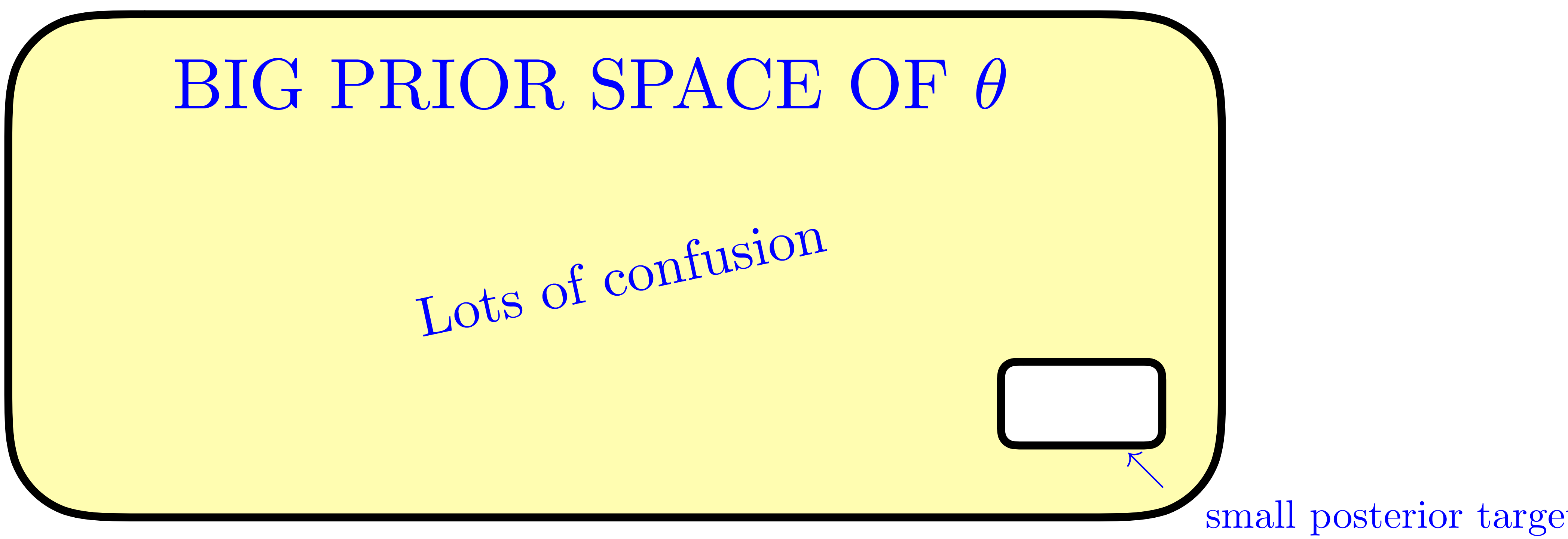
$$d\pi(\theta) \times L(\theta) \quad \longrightarrow \quad Z \;\times\; dP(\theta)$$

$d\pi(\theta)$ ↑ Question; Z ↓ Explanatory?; $dP(\theta)$ ↓ Answer

THE framework of inference

(1) Evidence: $Z = \int L(\theta)\, d\pi(\theta)$ ⟵ Looks hard but isn't.

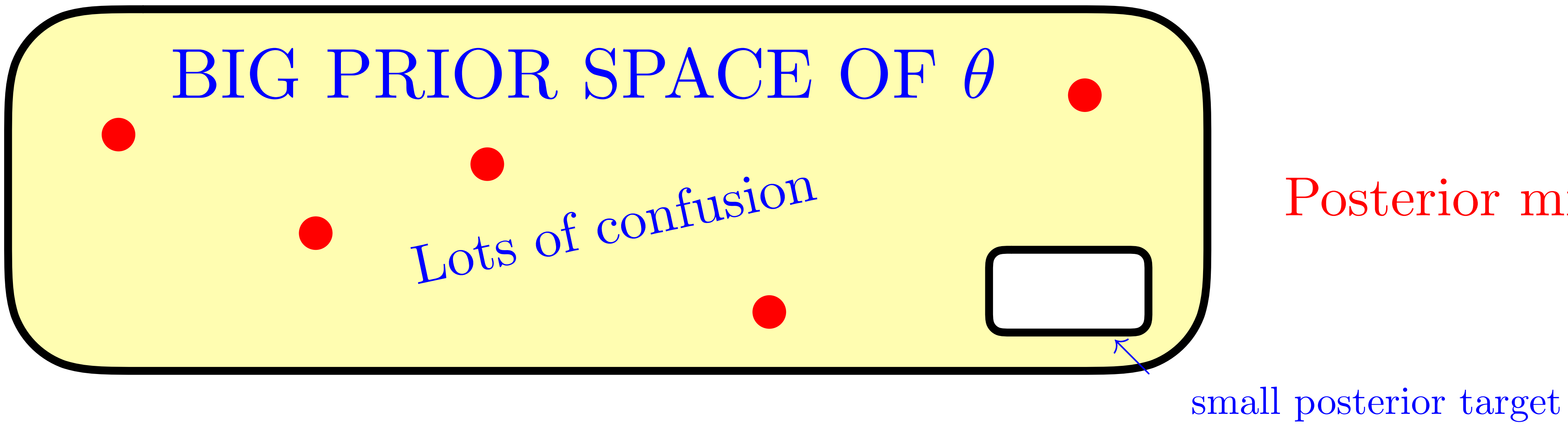
(2) Posterior: $dP(\theta) = \dfrac{L(\theta)\, d\pi(\theta)}{Z}$ ⟵ Secondary.



Can't look everywhere. Must sample (Monte Carlo).

Hopeless methods for $Z = \int L(\theta)\, d\pi(\theta)$

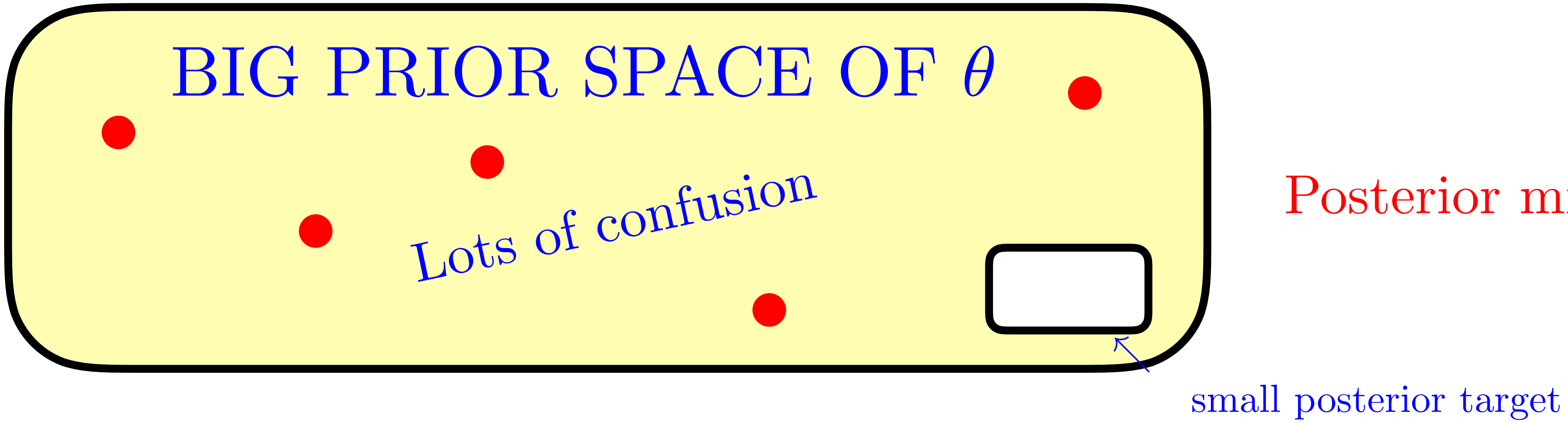
Direct: $Z = \langle L \rangle_{\text{prior}}$



Posterior missed

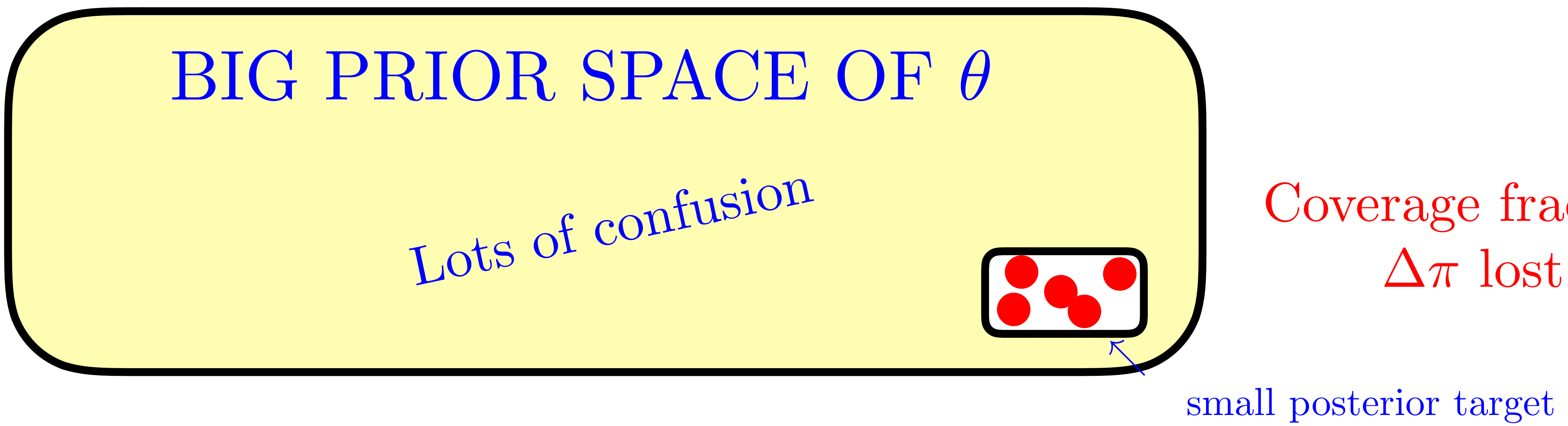
Hopeless methods for $Z = \int L(\theta)\, d\pi(\theta)$

Direct: $Z = \langle L \rangle_{\text{prior}}$



Posterior missed

Harmonic mean: $\frac{1}{Z} = \left\langle \frac{1}{L} \right\rangle_{\text{posterior}}$



Coverage fraction
$\Delta\pi$ lost

Must connect prior-to-posterior.

Must connect prior-to-posterior. ✓✓

Another bad idea (annealing)

Laplace transform - terrible idea.
Destroys detail. Can't do phase changes.
(Nothing interesting.)

$$Z(\beta) = \int L(\theta)^\beta \, d\pi(\theta)$$

$\beta = 0$ prior $\longrightarrow$ $\beta = 1$ posterior

The 20th century got a lot wrong.

1. It argued about Bayes.
2. It viewed functions through 19th century lens.
3. It viewed inference through Laplace transform.
4. It thought dimensionality was hard.

Back to first principles — SMASH DIMENSIONALITY!

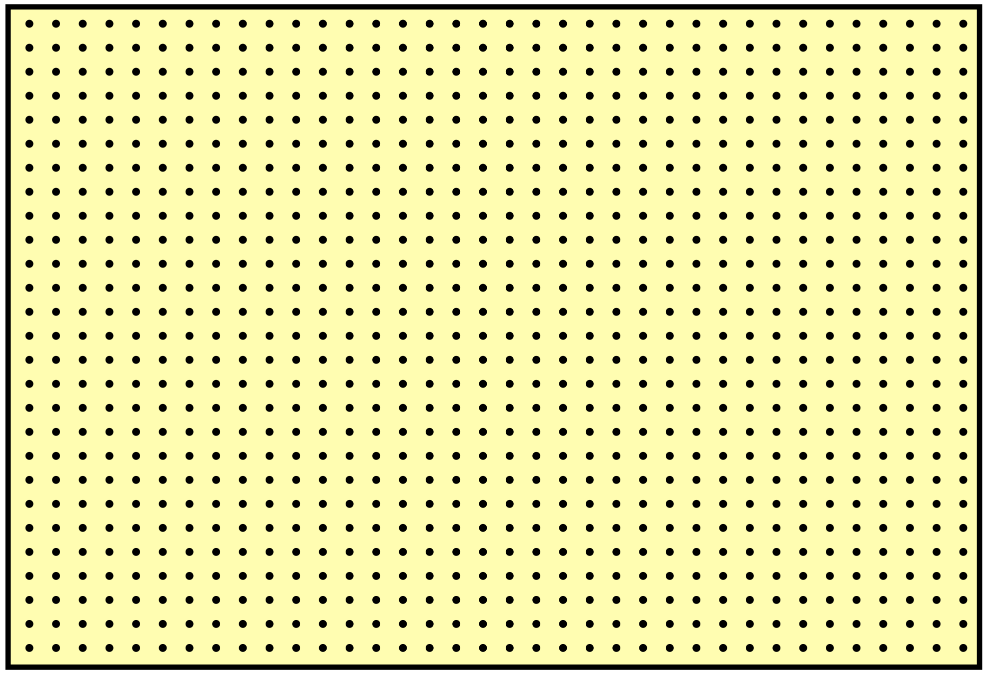
✓✓

Compression

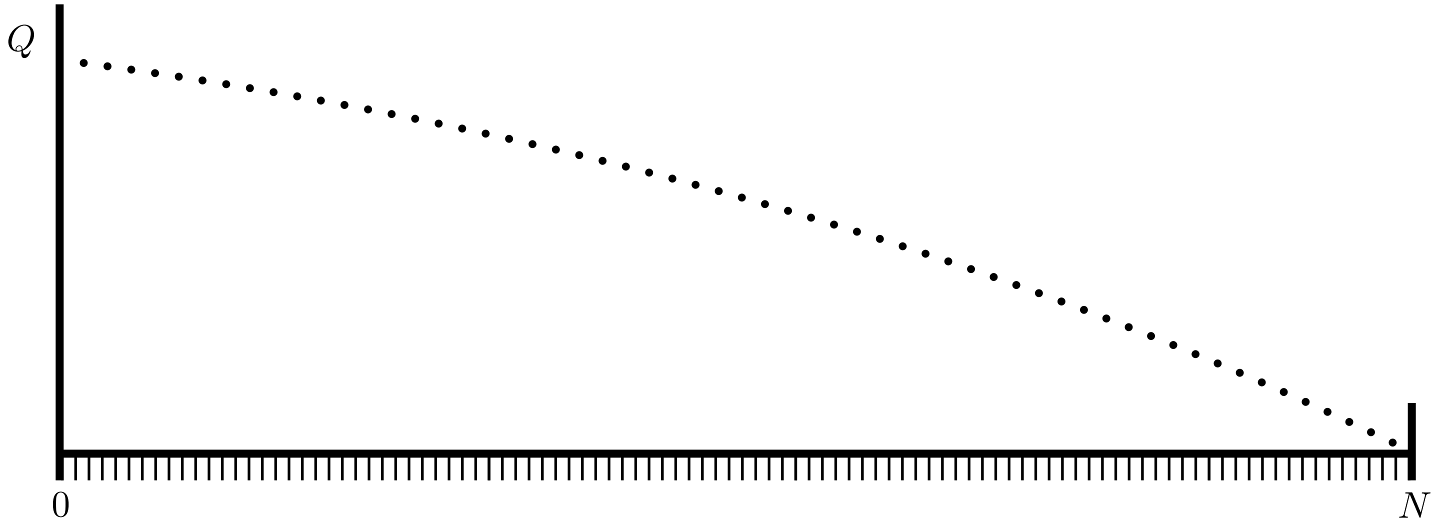
Q: How does a mathematician find a needle in a haystack?

A: Keep halving the haystack and discarding the "wrong" half.

Performance: factor of 2 per step.



N possible points



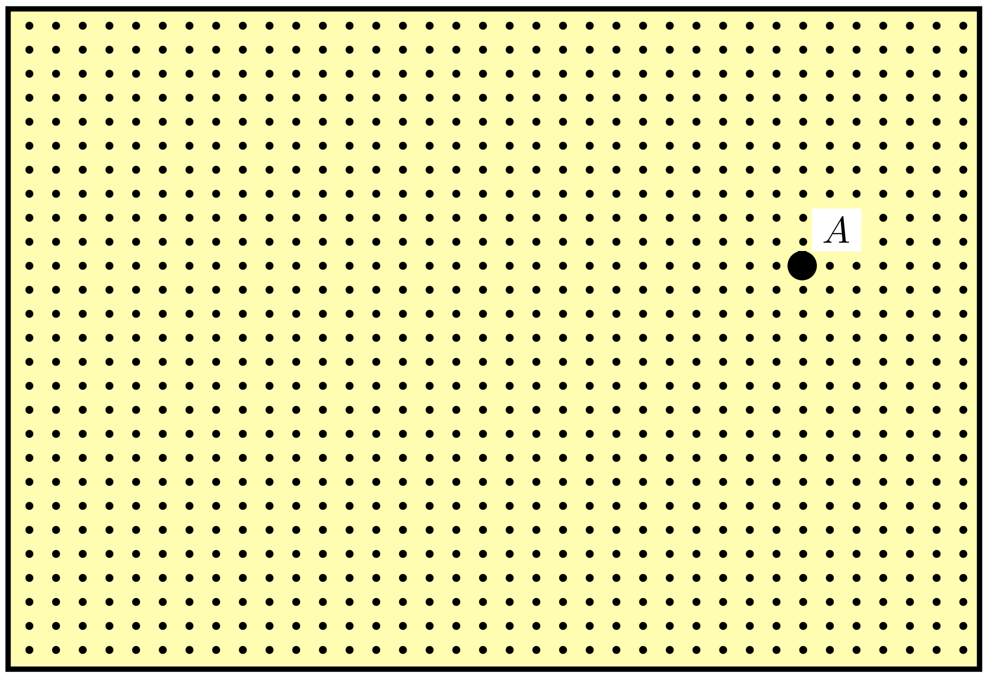
N points ranked by quality

Compression

Q: How does a programmer find a small target?

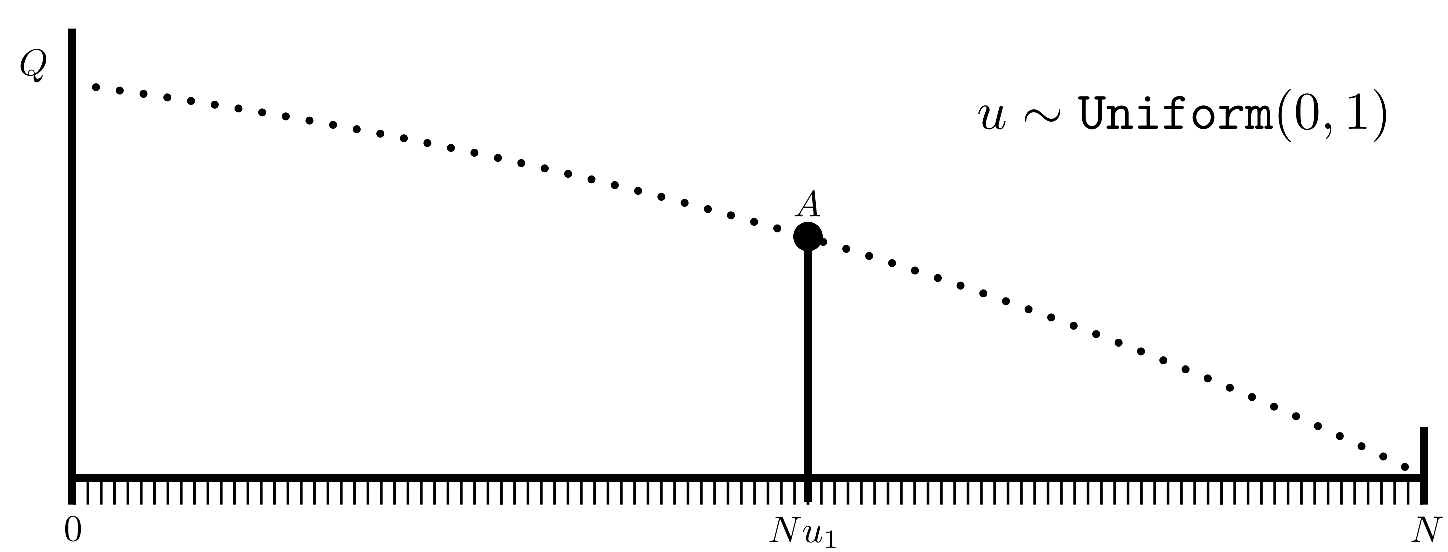
A: Keep taking a random point and discarding everywhere worse.

Performance: statistically a factor of e per step.



N possible points

A random $\qquad \log(u_1) = -1 \pm 1$



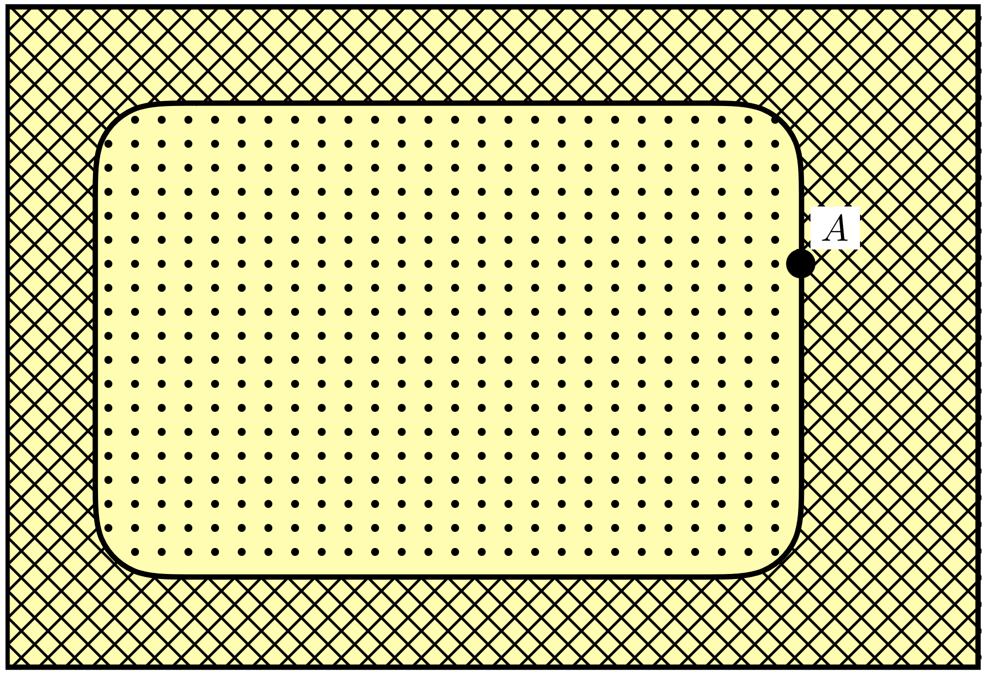
N points ranked by quality

Compression

Q: How does a programmer find a small target?

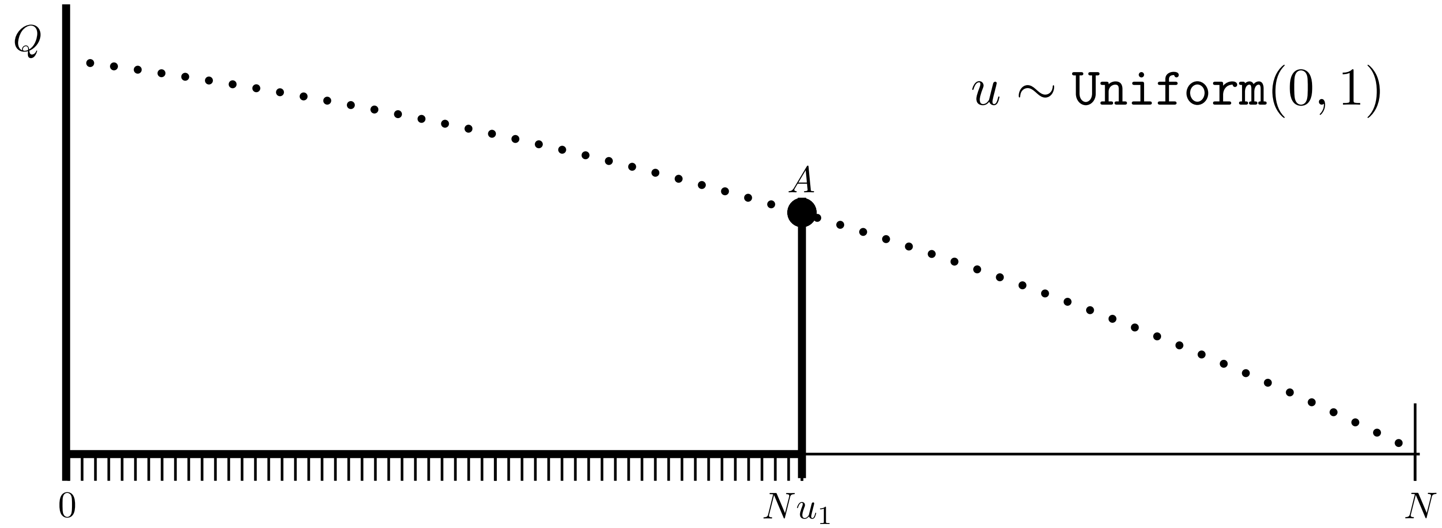
A: Keep taking a random point and discarding everywhere worse.

Performance: statistically a factor of e per step.



N possible points

A random $\qquad \log(u_1) = -1 \pm 1$



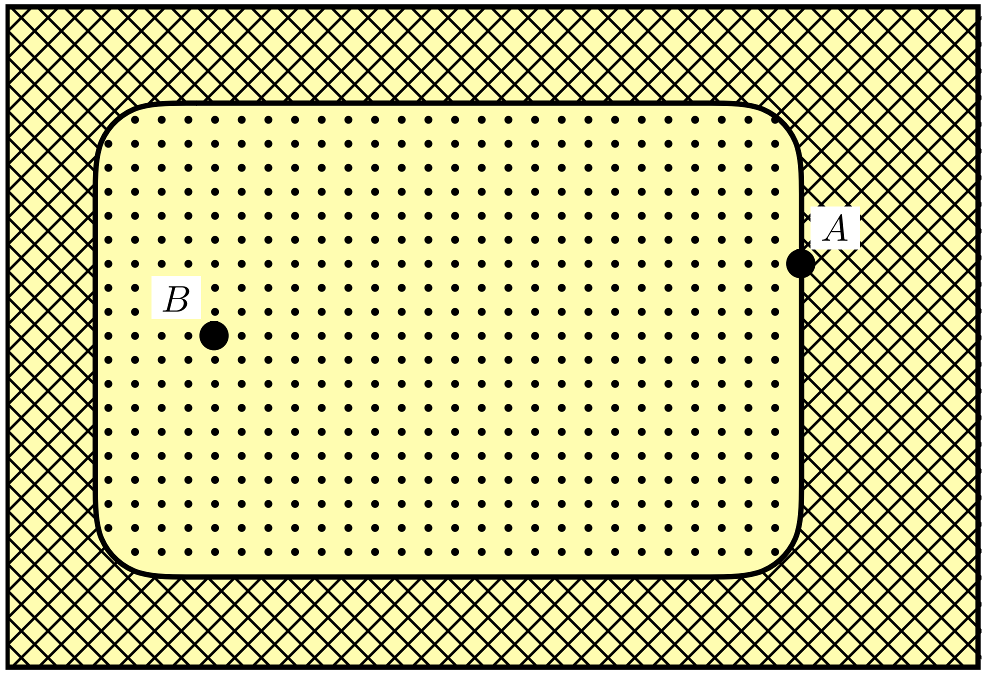
N points ranked by quality

Compression

Q: How does a programmer find a small target?

A: Keep taking a random point and discarding everywhere worse.

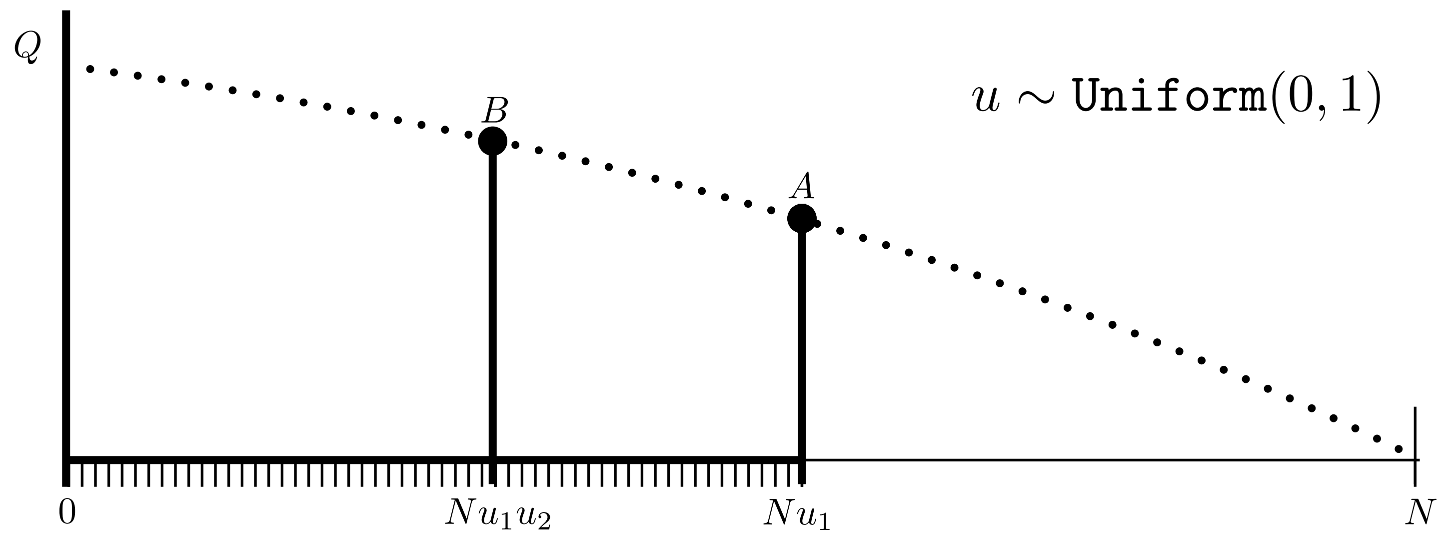
Performance: statistically a factor of e per step.



N possible points

A random $\quad \log(u_1) = -1 \pm 1$

B random in $Q \geq Q(B)$ $\quad \log(u_1 u_2) = -2 \pm \sqrt{2}$



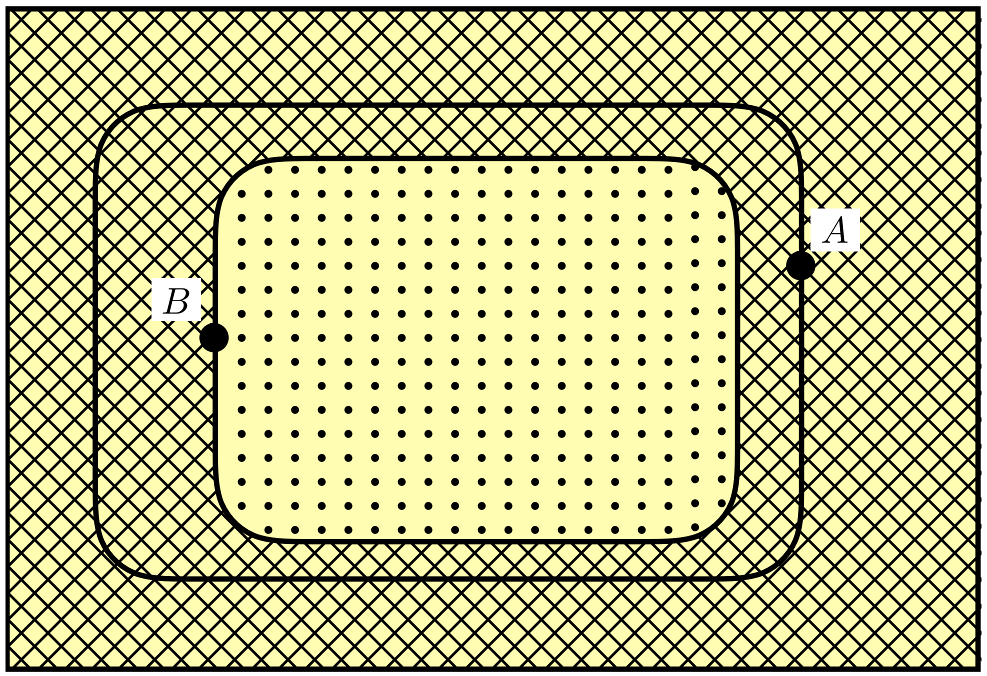
N points ranked by quality

Compression

Q: How does a programmer find a small target?

A: Keep taking a random point and discarding everywhere worse.

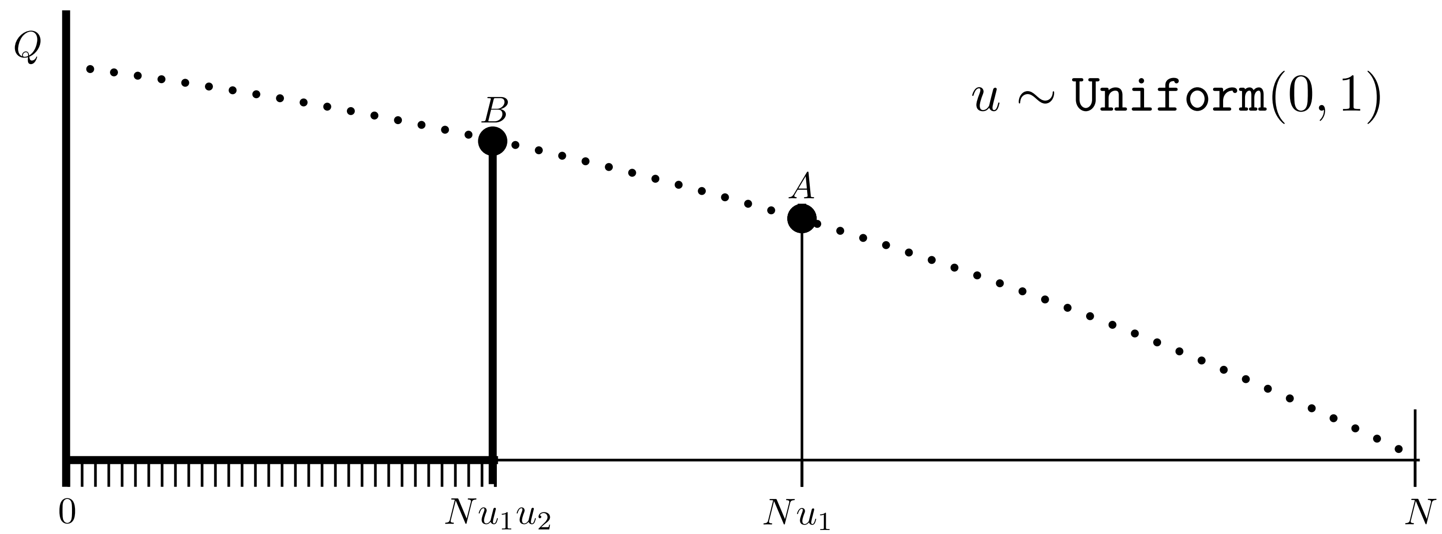
Performance: statistically a factor of e per step.



N possible points

A random $\qquad \log(u_1) = -1 \pm 1$

B random in $Q \geq Q(B)$ $\qquad \log(u_1 u_2) = -2 \pm \sqrt{2}$



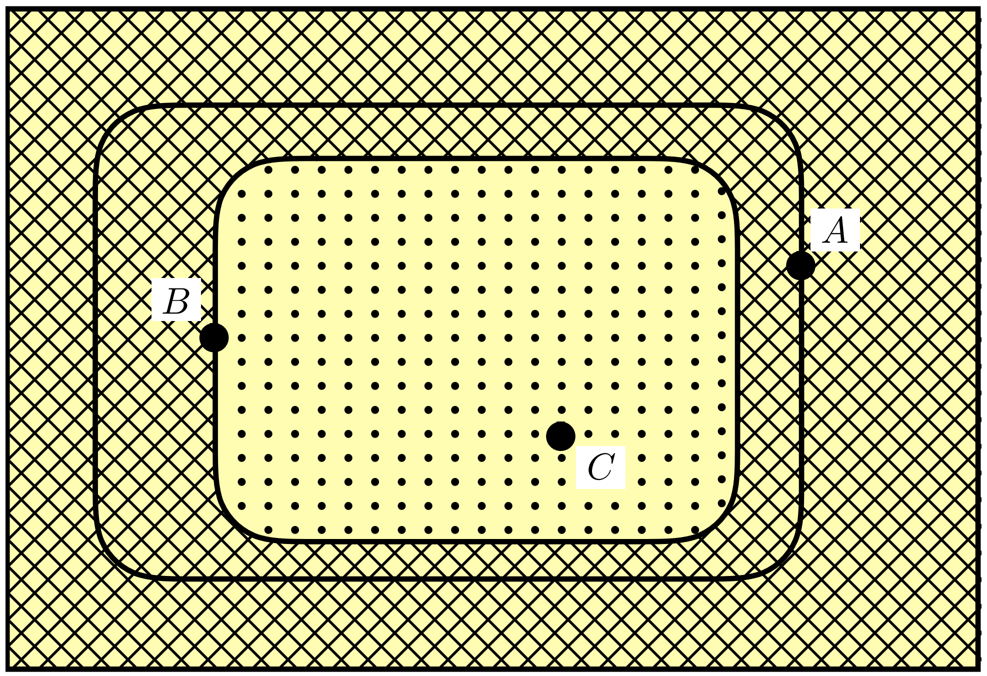
N points ranked by quality

Compression

Q: How does a programmer find a small target?

A: Keep taking a random point and discarding everywhere worse.

Performance: statistically a factor of e per step.

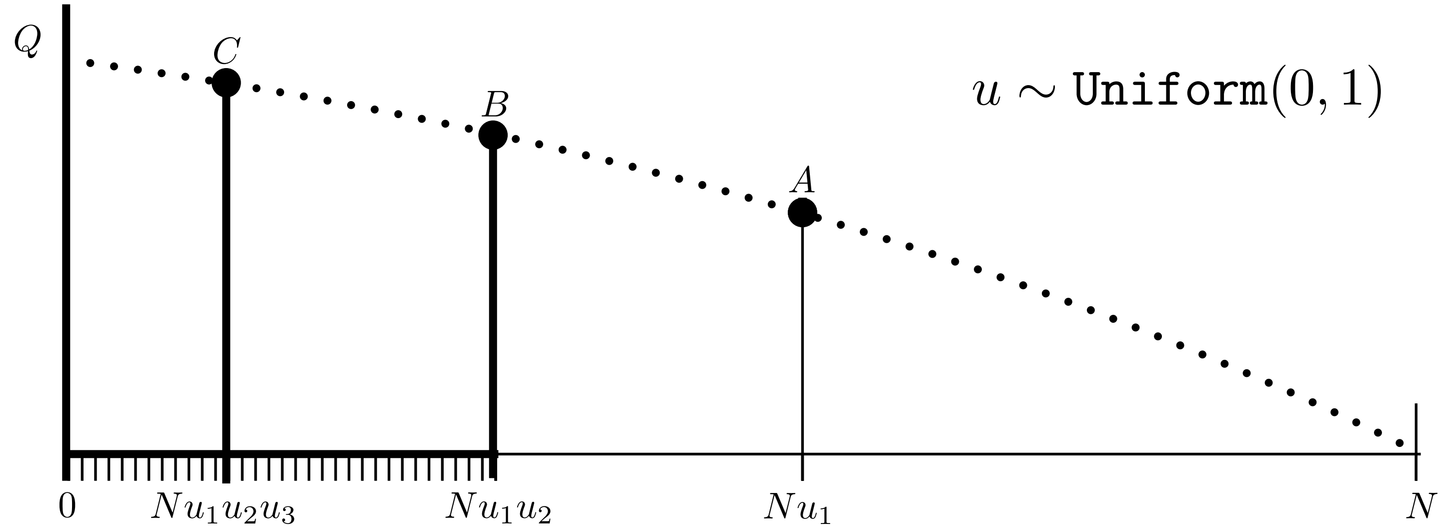


N possible points

A random $\quad \log(u_1) = -1 \pm 1$

B random in $Q \geq Q(B)$ $\quad \log(u_1 u_2) = -2 \pm \sqrt{2}$

C random in $Q \geq Q(C)$ $\quad \log(u_1 u_2 u_3) = -3 \pm \sqrt{3}$



N points ranked by quality

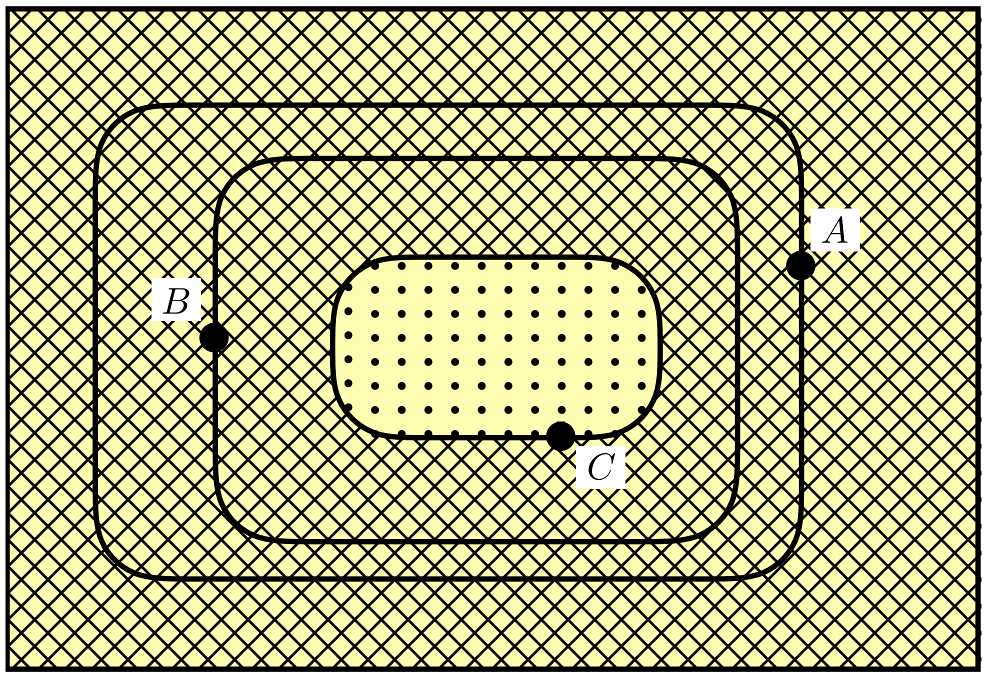
Compression

Q: How does a programmer find a small target?

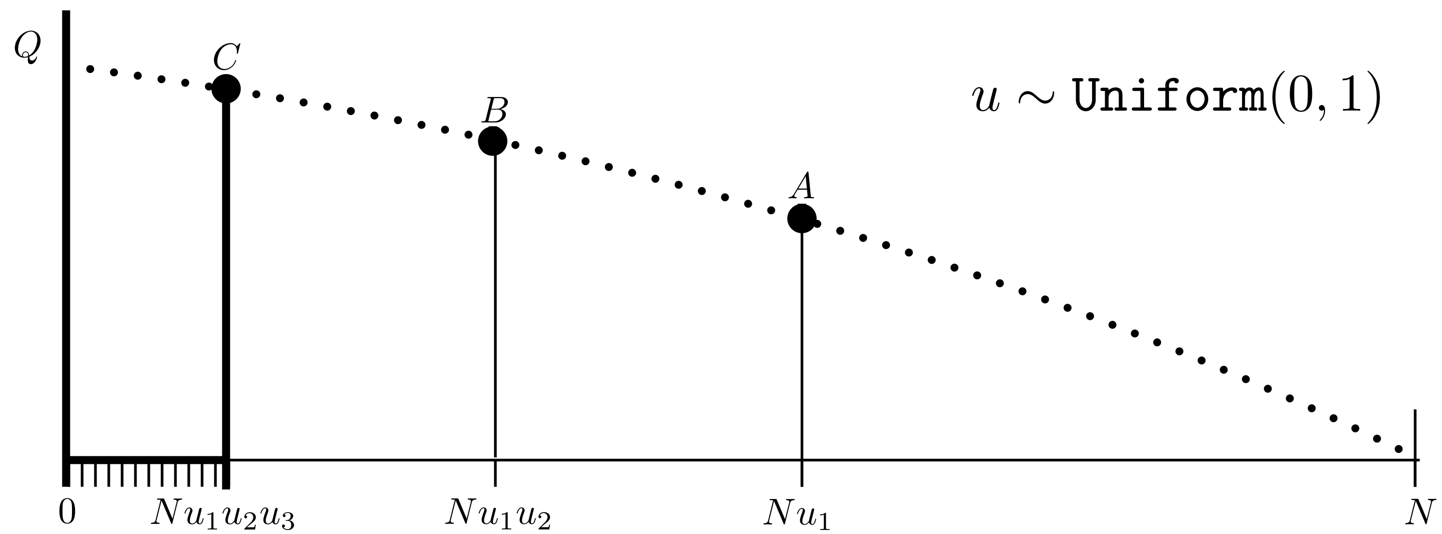
A: Keep taking a random point and discarding everywhere worse.

Performance: statistically a factor of e per step.

A random	$\log(u_1) = -1 \pm 1$
B random in $Q \geq Q(B)$	$\log(u_1 u_2) = -2 \pm \sqrt{2}$
C random in $Q \geq Q(C)$	$\log(u_1 u_2 u_3) = -3 \pm \sqrt{3}$



N possible points



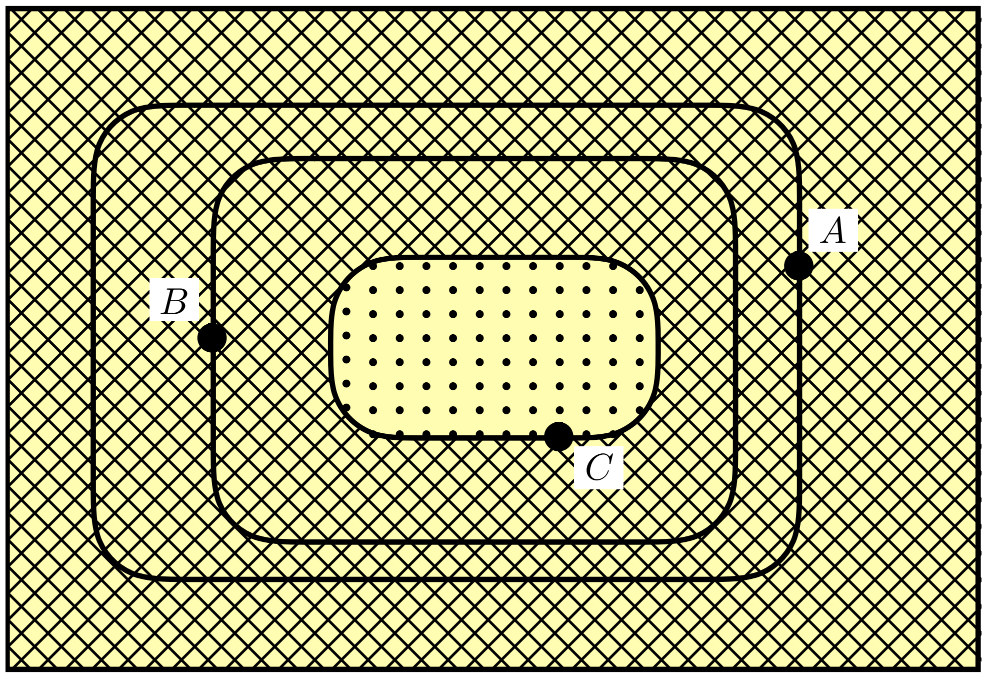
N points ranked by quality

Compression

Q: How does a programmer find a small target?

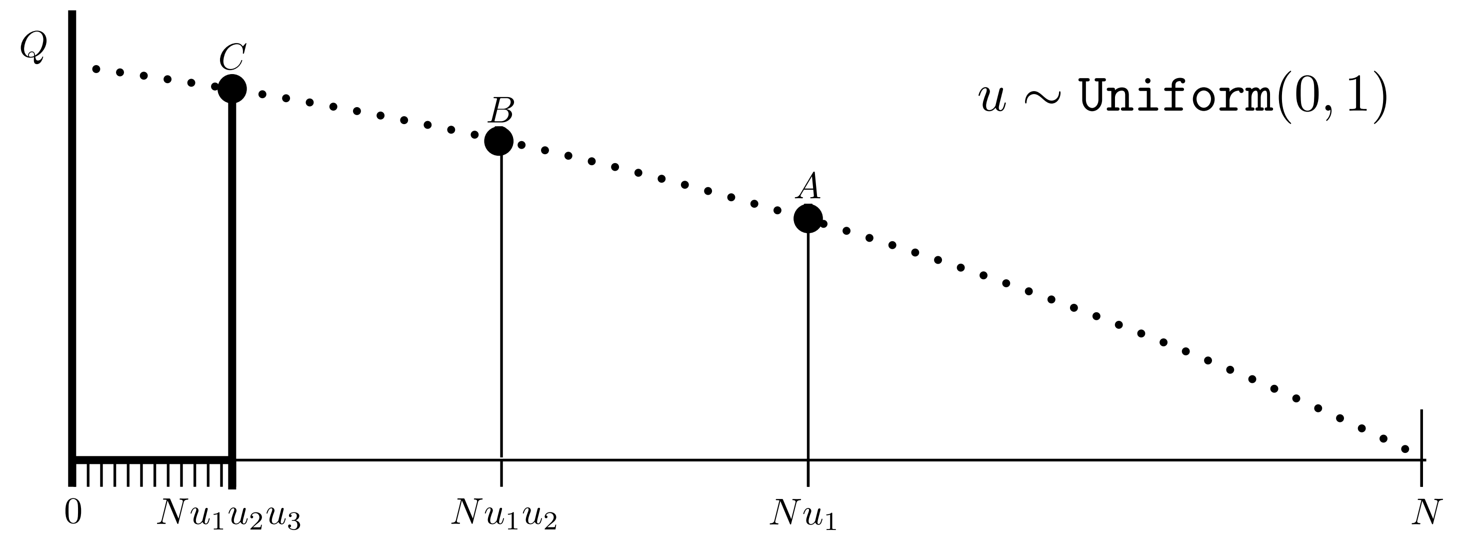
A: Keep taking a random point and discarding everywhere worse.

Performance: statistically a factor of e per step.



N possible points

A random	$\log(u_1) = -1 \pm 1$
B random in $Q \geq Q(B)$	$\log(u_1 u_2) = -2 \pm \sqrt{2}$
C random in $Q \geq Q(C)$	$\log(u_1 u_2 u_3) = -3 \pm \sqrt{3}$



N points ranked by quality

After k steps, accumulated compression $e^{k \pm \sqrt{k}}$ has enclosed quality $Q \geq Q_k$.

$$\log\left(\frac{\#\ \text{targets}}{\#\ \text{possibles}}\right) = -k \pm \sqrt{k}$$

Compression estimated statistically in log(ratio) steps *without exploring everywhere.*

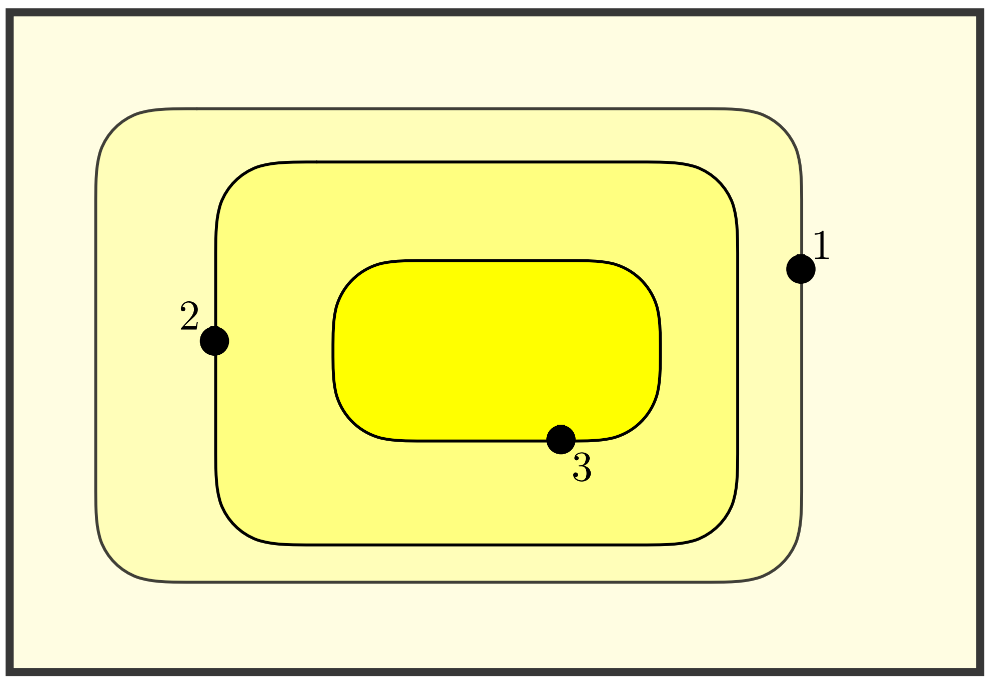
This is the *nested sampling* algorithm.

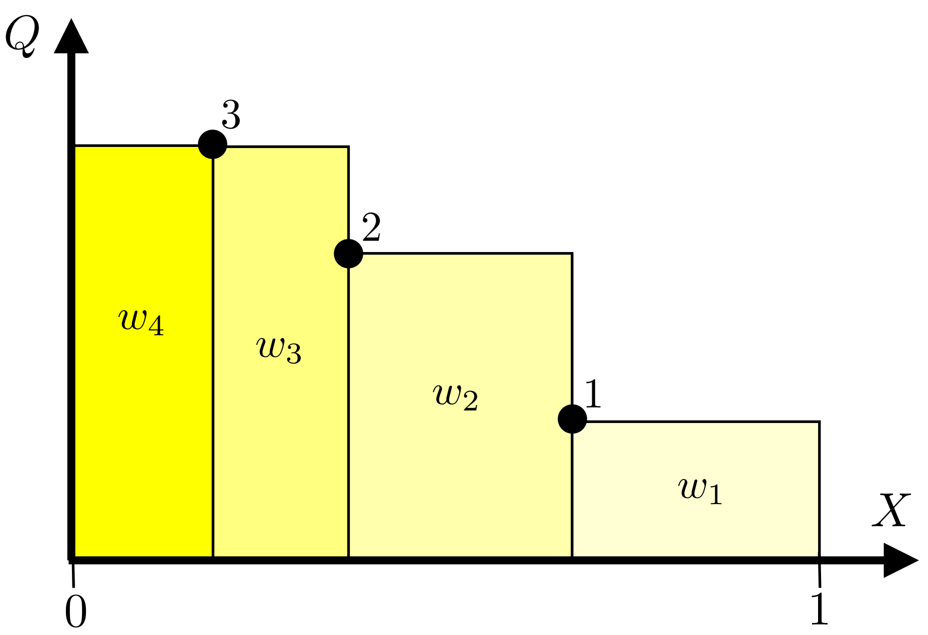
Quantification

A nested sampling run yields a sequence $k = 1, 2, 3, \ldots$ of

$$\begin{cases} \mathbf{x}_k = \text{location} \\ Q(\mathbf{x}_k) = \text{quality} \\ X_k = \text{fraction of possibilities with quality} \geq Q\text{, as statistical estimate} \end{cases}$$

Thus it gives the relationship $Q(X)$ and a sample location $\mathbf{x}$ in each shell of quality.





Shell k contributes $\quad w_k = Q_k\, \Delta X_k \quad$ to $\quad \int Q(\mathbf{x}) d\mathbf{x} = \int Q(X) dX \approx \sum Q_k\, \Delta X_k.$

So we can integrate in any geometrical dimension — we just need the background measure.

Quantification — Bayes

Quality = Likelihood

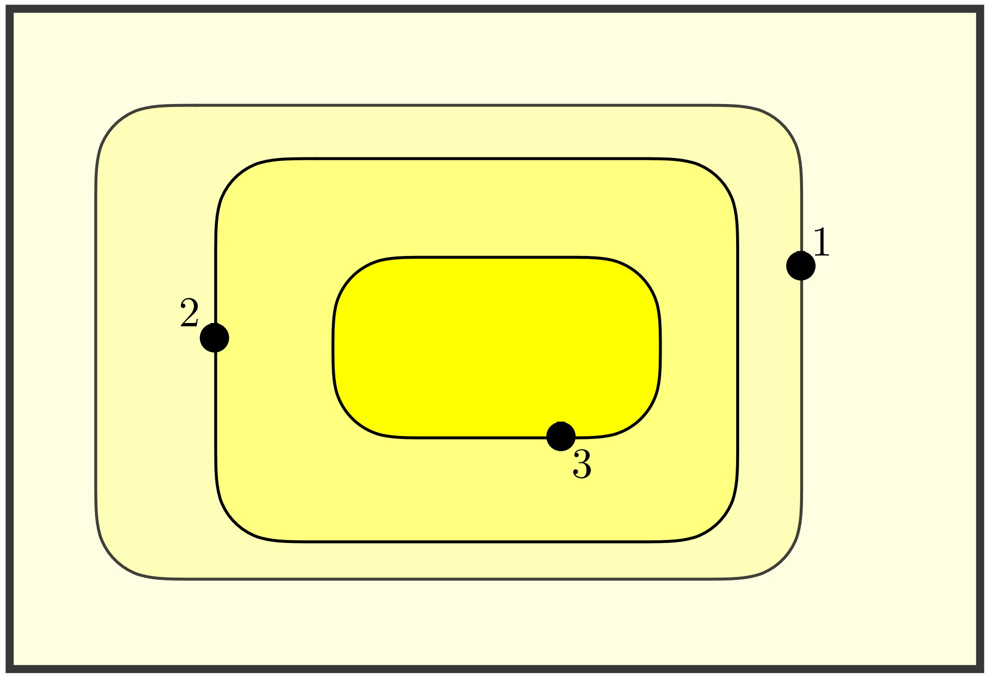
$$\underbrace{\mathrm{Prob}(\mathbf{x})}_{\text{Prior } dX} \times \underbrace{\mathrm{Prob}(\mathrm{data} \mid \mathbf{x})}_{\text{Likelihood } L} = \mathrm{Prob}(\mathbf{x}, \mathrm{data}) = \underbrace{\mathrm{Prob}(\mathrm{data})}_{\text{Evidence } Z} \times \underbrace{\mathrm{Prob}(\mathbf{x} \mid \mathrm{data})}_{\text{Posterior } dP}$$

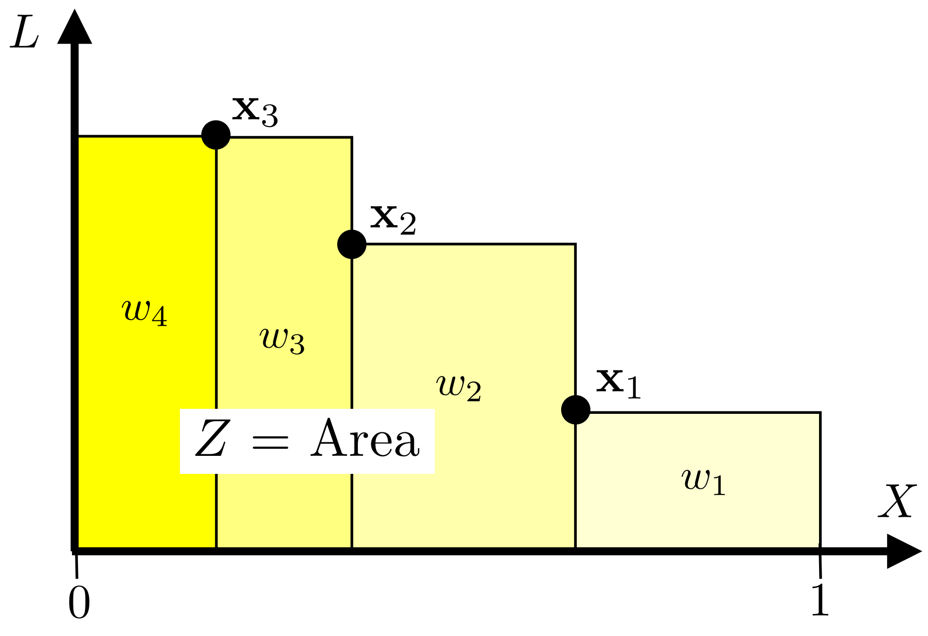
Inputs $\Longrightarrow$

- Prior is $\pi(\mathbf{x})$, equivalent to uniform on $0 < X < 1$
- Likelihood is $L(\mathbf{x})$ or $L(X)$

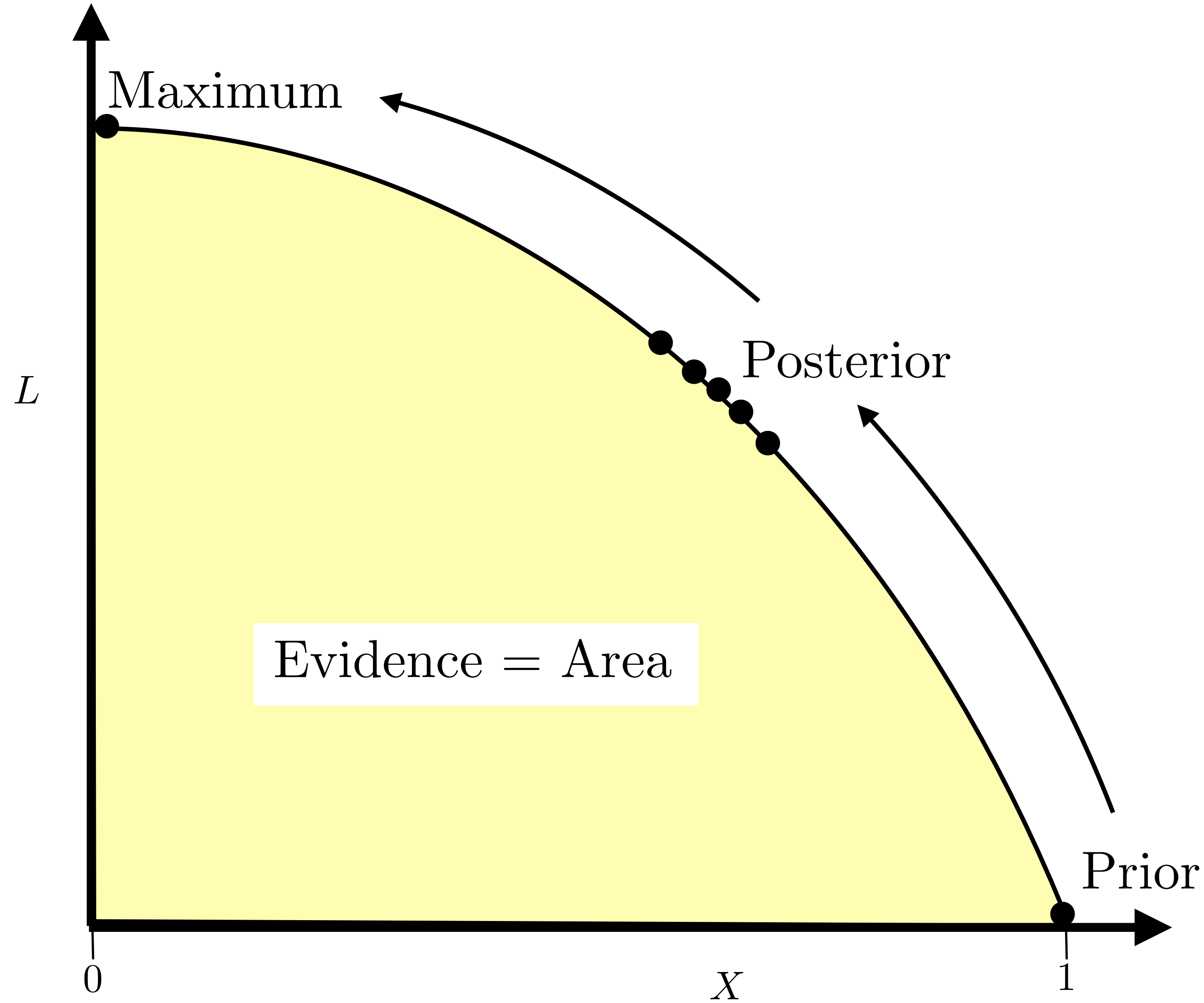
- Evidence is $Z = \int L(\mathbf{x})\pi(\mathbf{x})d\mathbf{x} = \int_0^1 L(X)dX$ or $Z \approx \sum w_k$ (1)
- Posterior is weighted samples $\mathbf{x}_1, \mathbf{x}_2, \ldots,$ with $\mathrm{Prob}(k) = w_k/Z$ (2)

$\Longrightarrow$ Outputs





Quantification — Bayes by nested sampling.



Compression yields posterior samples on the fly.

Unified computation of evidence and posterior.

Take-home message:

Bayes is required for consistent inference and
it's not as hard as you may have thought.

Not all that glitters is gold:

Maximum . . .	*non-Bayesian*
Posterior	*semi-Bayesian*
Posterior + Evidence	*Bayesian*

John Skilling, AI for Astronomy 2019

Garching-bei-München, EU