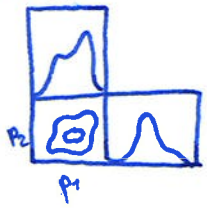
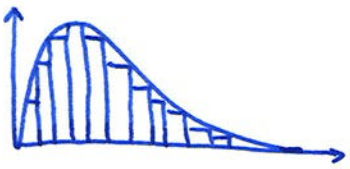
Sampling — 8th April 2019

Want:



or 

or $x \sim G(0, \sigma^2)$ then $x_1/x_2 \sim ?$

> Holy Grail of Sampling: generate samples from a distribution such that
>
> $$n(\vec{\theta}) \propto \mathcal{P}(\vec{\theta})$$

1) Why do we sample? (⇝ Integrals)
2) How do we sample? (⇝ Algorithms)

1) Why:

Imagine data $\vec{d}$, Parameters $\vec{\theta}$, and you want to constrain $\vec{\theta}$ by $\vec{d}$ ⇝ you want to map out the posterior $\mathcal{P}(\vec{\theta}|\vec{d})$

$$\mathcal{P}(\vec{\theta}|\vec{d}) = \frac{\mathcal{L}(\vec{d}|\vec{\theta})\,\Pi(\vec{\theta})}{\mathcal{E}(\vec{d})}$$

($\mathcal{L}$: likelihood, Π: prior, $\mathcal{P}$: posterior, $\mathcal{E}$: evidence)

with $\mathcal{E}(\vec{d}) = \int d^n\theta\, \mathcal{L}(\vec{d}|\vec{\theta})\,\Pi(\vec{\theta})$

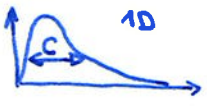
[needed for model selection; it normalizes the posterior]

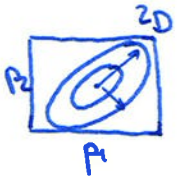
Want: samples which satisfy that their number density $n(\vec{\theta})$ is proportional to the posterior probability: $n(\vec{\theta}) \propto \mathcal{P}(\vec{d}|\vec{\theta})\Pi(\vec{\theta})$

⇝ Such samples make integration trivial:

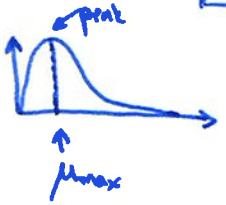
$$\vec{\mu} = \int \vec{\theta}\, \mathcal{P}(\vec{\theta}|\vec{d})\, d^n\theta$$

$$C_{\vec{\theta}} = \int (\vec{\theta}-\vec{\mu})(\vec{\theta}-\vec{\mu})^T\, \mathcal{P}(\vec{\theta}|\vec{d})\, d^n\theta$$





$$\vec{\mu}_{max} = \text{argmax}\left[\mathcal{P}(\vec{\theta}|\vec{d})\right]$$

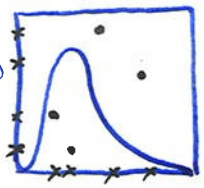


+ the full distribution!

Because then we can do all we want, as the distribution lies at the heart of all calculations.

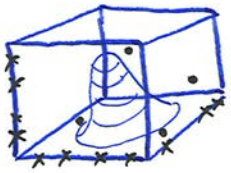
Fighting the curse of dimensionality

2D: (for 1d-distribution)



⇝ 2 hits, out of 4

3D (for 2D-distribution)



⇝ Zero hits out of 4

⇝ $D = 100$ is still a „small" statistics dimension!

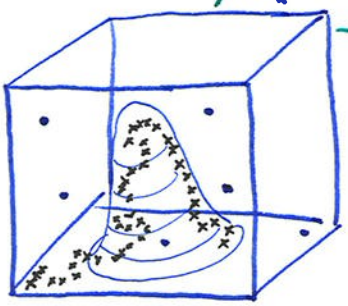
⇝ How do we sample in high dimensions?

⇒ Problem diagnosis: r_1, r_2, r_3 drew random numbers independently.

⇒ enforce dependence by linking them together: independent trials ⇝ Chain

⇒ dependency always bad, hence minimize dependencies in the chain:

> Demand: if $x_1, x_2, x_3, \ldots, x_n$ samples stored in a chain, then x_i shall only depend on x_{i-1}. (Minimal possible dependence)
>
> ⇝ „Monte Carlo Markov Chain"



- independent trials (rejection sampling)
- × MCMC

⇝ works close to always

⇝ the best universal samplers all fall into the category Monte Carlo Markov Chain, but there are different algorithms, all of which build up MCMC chains.

What happens when we sample?

(under the hood)

→ This can be seen quite well in the Gnu Scientific Library (GSL)

```
#include <gsl/gsl_rng.h>          (rng: random number generator)
#include <gsl/gsl_randist.h>      (randist: distribution)

int main()
{
    gsl_rng * r;                  // pseudo random number generator (series of random numbers [0,1])
    int seed = 404;               // pseudo! we can controll the seed
    gsl_rng_set (r, seed);

    double x = gsl_ran_poisson (r, μ);   // to sample from a distribution is more complex than creating random numbers
                                         // accepts input parameters, e.g. μ = mean = 4.0
                                         // does „something" to the random numbers of the generator
                                         // outputs poisson-distributed random numbers
    ⋮
}
```

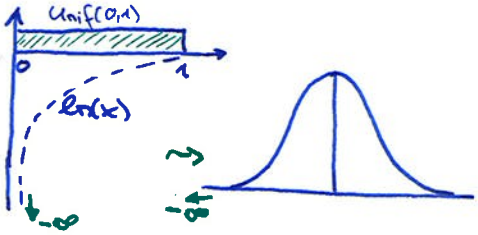
Example: generating Gaussian random numbers (Box-Muller transform)

→ Works if transformation is known

- if $x_1, x_2 \sim \text{Unif}[0,1]$ then $g_1, g_2 \sim$ standard normal, where

$$g_1 = \sqrt{-2\ln(x_1)}\cos(2\pi x_2)$$

$$g_2 = \sqrt{-2\ln(x_1)}\sin(2\pi x_2)$$



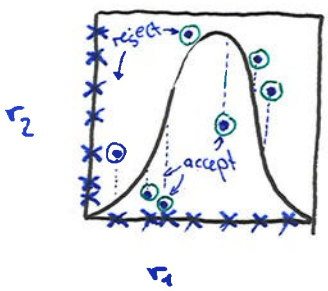
→ pairwise draw from random number generator

→ pairwise production of gaussian normal variates $\sim \mathcal{N}(0,1)$ } → „efficiency 1"

→ if $x \sim \mathcal{N}(0,1)$, then $y = (\sigma \cdot x + \mu) \sim G(\mu, \sigma^2)$

Example: generating Gaussian random numbers via rejection sampling

→ Works if envelope or at least maximum can be guessed



$y = \text{Gauss}(r_1)$, r_2

if $r_2 < y$: accept

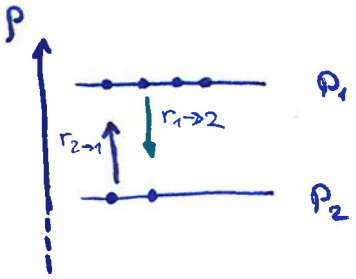
if $r_2 > y$: reject

} → efficiency depends on sampling envelope (here: limits for r_2)

Metropolis Algorithm for MCMC

Prerequisite: „Detailed Balance":

Minimal case: • 2 discrete probability levels:



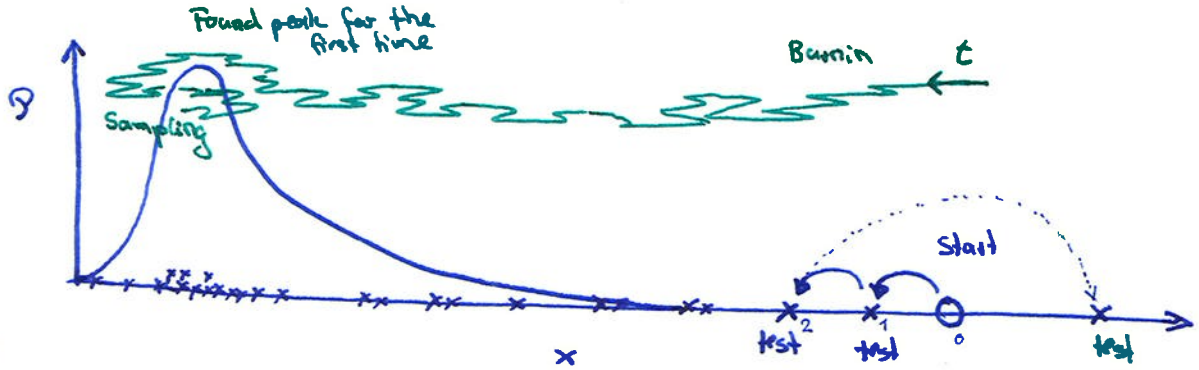
- want (always) $n \propto p$
- want (asymptotically): equilibrium, meaning the chain shall equilibrate to the target distribution

→ Equilibrium reached if

$$r_{1\to2}\, p_1 = r_{2\to1}\, p_2$$

„detailed balance"

Metropolis algorithm:



- Pick x_{start} ; $x_0 = x_{start}$
- compute $p(x_{start})$

for i in range $(1, N_{chain})$:
- draw $\Delta \sim D$
- $x_i + \Delta$: test point
- Compute $p(x_i + \Delta)$
- if $p(x_i+\Delta) > p(x_i)$: accept : $x_{i+1} = x_i + \Delta$
- if $p(x_i+\Delta) < p(x_i)$:

  $\alpha \sim \text{Unif}[0,1]$

  if $\alpha \leq \frac{p(x_i+\Delta)}{p(x_i)}$: accept nonetheless : $x_{i+1} = x_i + \Delta$

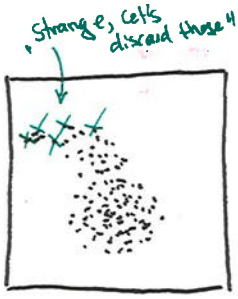
  if $\alpha > \frac{p(x_i+\Delta)}{p(x_i)}$: reject : $x_{i+1} = x_i$

(Pseudo code in script)

→ to satisfy detailed balance points in the chain must be repeated or have weights

(in) Admissible manipulations of MCMC chains

1)



↝ All samples of a thinned (or uncorrelated) chain are valid samples: do not pick or discard at will.

2)

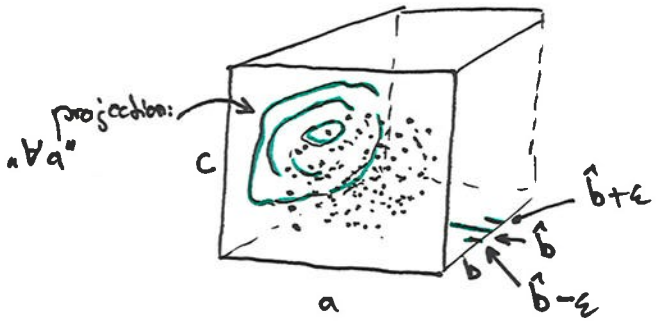
chain.txt

0.1	-4.3	19.4	121.4
0.09	-4.2	18.3	139.4
0.095	-4.35	17.8	129.5
⋮			

parameter1 ↑ parameter2 ↑ parameter3 ↑ $-2\log(\mathcal{P}(\vec{\theta}|\vec{x}))$

Marginalization = Integration = ignoring values in uninteresting columns

Conditionalization = selection: only keep samples which fall into some $\pm\varepsilon$ interval:



3) Monte Carlo integration:

$$\text{if } x_i \sim \mathcal{P}(\vec{x}|M) \quad \text{then} \quad \int dx\, \mathcal{P}(x|M)\, g(x) \approx \frac{1}{N}\sum_{n=1}^{N} g(x_n)$$

Mind the if-then! If your samples have a poor quality, then your integrals come out incorrectly!
↳ „convergence" if your samples come from a chain

Examples:

- Let $g(x)=1$, then $\frac{1}{N}\sum_{n=1}^{N} g(x_n) = 1$ (counting measure, not a normalization).
- $g(x)=x$, then $\frac{1}{N}\sum x_n = \langle x\rangle$, the mean.
- $g(\vec{x}) = (\vec{x}-\langle\vec{x}\rangle)(\vec{x}-\langle\vec{x}\rangle)^T$, then $\frac{1}{N}\sum_{n=1}^{N}(\vec{x}_n-\langle\vec{x}\rangle)(\vec{x}_n-\langle\vec{x}\rangle)^T$ (sample estimated covariance matrix)

4) Your chain will contain a sample with the highest likelihood, in comparison to the others. This is an estimate of the true peak. It will change for each seed or starting point.