Simulation based Nearest Neighbor entropy estimation for MCMC evaluation

Didier Chauveau

MAPMO - UMR 7349 - Université d'Orléans

Joint work with P. Vandekerkhove

24ème journée CASCIMODOT — 17 Juin 2016

Outline

Outline: Next up...

Notations for Bayesian model & MCMC setup

- Bayesian model:
  - the d-dimensional parameter $\theta \in \Theta \subseteq \mathbb{R}^d$
  - a prior distribution $\pi(\theta)$
  - the data $\boldsymbol{X} \sim \ell(\boldsymbol{x}|\theta) =$ the likelihood of the statistical model
  - the posterior $f(\theta|\boldsymbol{x}) = \dfrac{\ell(\boldsymbol{x}|\theta)\pi(\theta)}{\int \ell(\boldsymbol{x}|\theta)\pi(\theta)\, d\theta}$ non closed-form in general

Omit the dependence to $\boldsymbol{x}$ and denote the posterior simply $f(\theta)$ usually known only up to its normalizing constant: $f(\theta) \propto \ell(\boldsymbol{x}|\theta)\pi(\theta)$

Notations for Bayesian model & MCMC setup

- Bayesian model:
  - the d-dimensional parameter $\theta \in \Theta \subseteq \mathbb{R}^d$
  - a prior distribution $\pi(\theta)$
  - the data $\boldsymbol{X} \sim \ell(\boldsymbol{x}|\theta) =$ the likelihood of the statistical model
  - the posterior $f(\theta|\boldsymbol{x}) = \dfrac{\ell(\boldsymbol{x}|\theta)\pi(\theta)}{\int \ell(\boldsymbol{x}|\theta)\pi(\theta)\, d\theta}$ non closed-form in general

Omit the dependence to $\boldsymbol{x}$ and denote the posterior simply $f(\theta)$ usually known only up to its normalizing constant: $f(\theta) \propto \ell(\boldsymbol{x}|\theta)\pi(\theta)$

- A MCMC sampler generates a Markov Chain (MC) $\theta^1, \ldots, \theta^t, \ldots$ s.t.

$$\theta^t \sim f = \text{the sampler “target density” when } t \to \infty$$

Statistical inference from the simulations (LGN, TLC for MC’s)

A “famous” MCMC algorithm example

The Hastings-Metropolis Algorithm (HM)

Tool: a proposal density $q(y|x)$ easy to simulate
Iteration $\theta^t \to \theta^{t+1}$:

1. `simulate` $y \sim q(\cdot|\theta^t)$
2. `let` $\alpha(\theta^t, y) = \min\left\{1, \dfrac{f(y)q(\theta^t|y)}{f(\theta^t)q(y|\theta^t)}\right\}$
3. `set` $\theta^{t+1} = \begin{cases} y & \texttt{with probability } \alpha(\theta^t, y) \\ \theta^t & \texttt{with probability } 1-\alpha(\theta^t, y) \end{cases}$

Under mild conditions, $\left(\theta^t\right)$ is a Harris positive, ergodic Markov chain with stationnary distribution π

(i) Random Walk HM (RWHM), typically $y \sim \mathcal{N}_d(\theta^t, \Sigma) \equiv q(\cdot|\theta^t)$
(ii) Independence Sampler (IS), i.e. $q(y|x) \equiv q(y)$

Motivation: MCMC samplers evaluation/comparison

Current problems in MCMC application and research:

- Difficulties in assessing performance of specific MCMC samplers needed in actual applications
  (the old story of MCMC convergence assessment. . .)
- Many recent, including Adaptive MCMC (AMCMC) methods associated in practice to unknown rates of convergence
- Comparison of candidate (A)MCMC's against benchmarks, on synthetic or actual target densities

These questions are crucial even in small-to-moderate dimensions!

Partial review of current literature

Tools and methods for MCMC comparison:

- R package `SamplerCompare` Thompson (2010) (usual criterions, limited to 1 sampler tuning parameter, no HPC)
- Liu (2012) requires f-specific (tedious!) analytical work (small sets, regenerative cycles)...

Needs for comparisons in (A)MCMC developments and applications: does adaptation really helps?

- Altaleb & C (2002): Adaptive HM for logit models, $d \leq 5$
- Crain Rosenthal & Wang (2009)
- Vrugt et al. (2009): AMCMC against benchmarks, $d = 10$
- Bai et al. (2010): compares two AMCMC , $d \leq 5$

Motivation: A simulation-based software tool

Goal: To propose a methodological "black-box" tool

- Based on simulation only and not on a theoretical study which is typically MCMC and/or target-specific
- For MCMC users: to easily select a good sampler among possible candidates
- For researchers: to better understand which (A)MCMC methods perform best in which circumstances


package `EntropyMCMC`

Methodology based on simulation of "parallel" chains, using High Performance Computing (HPC)

Criterion: Kullback divergence to the target pdf

- p^t = the marginal density of the (A)MCMC at "time" (iteration) t
- Define the entropy of a probability density p over $\mathbb{R}^d$,

$$\mathcal{H}(p) := \int p \log p = \mathbb{E}_p(\log p)$$

A practical convergence/evaluation criterion:
the evolution in time (t) of the Kullback divergence

$$t \mapsto \mathcal{K}(p^t, f) := \int p^t \log\left(\frac{p^t}{f}\right) = \mathcal{H}(p^t) - \int (\log f) p^t$$

NB: Several results for Kullback divergence as a measure of MC's and MCMC's convergence Csiszár 1967, Miclo 1997, Harremoës and Holst 2007, C & Vandekerkhove 2007 & 2012, Douc et al. 2007...

Outline: Next up...

Estimation of $\mathcal{K}(p^t, f) = \mathcal{H}(p^t) - \int p^t \log f$

We have to estimate $\mathcal{H}(p^t)$ for an unknown p^t

- Many results on estimation of the entropy $\mathcal{H}(p)$ for uni- and multivariate density p from iid observations
- Historically mostly used only for $d = 1$ or 2
- Recent motivations for entropy estimation in molecular science, neuroscience, fluid mechanics for d "large"
  (Singh 2001, 2002; Harner et al. 2003; Stowell & Plumbley 2009...)

Estimation of $\mathcal{K}(p^t, f) = \mathcal{H}(p^t) - \int p^t \log f$

We have to estimate $\mathcal{H}(p^t)$ for an unknown p^t

- Many results on estimation of the entropy $\mathcal{H}(p)$ for uni- and multivariate density p from iid observations
- Historically mostly used only for $d = 1$ or 2
- Recent motivations for entropy estimation in molecular science, neuroscience, fluid mechanics for d "large"
  (Singh 2001, 2002; Harner et al. 2003; Stowell & Plumbley 2009...)

Remember the goal here: Criterion uniquely based on simulation from the MCMC sampler, no theoretical investigation

Estimation strategy:

- Build a nonparametric estimate of $\mathcal{H}(p^t)$ from N observations iid $\sim p^t$ at "time" t
- This requires N copies of iid chains, from which $\int p^t \log f$ can also be estimated

A "parallel chains" framework: N iid copies of the MC

Set $\theta_1^0, \ldots, \theta_N^0$ iid$\sim p^0 =$ some diffuse initial distribution, and simulate:

$$\begin{array}{rcl} \text{chain \# 1} & : & \theta_1^0 \to \theta_1^1 \to \cdots \to \theta_1^t \sim p^t \to \cdots \\ \vdots & & \vdots \\ \text{chain \# } i & : & \theta_i^0 \to \theta_i^1 \to \cdots \to \theta_i^t \sim p^t \to \cdots \\ \vdots & & \vdots \\ \text{chain \# } N & : & \theta_N^0 \to \theta_N^1 \to \cdots \to \theta_N^t \sim p^t \to \cdots \end{array}$$

- At "time" (or slice) t, the N chains locations

$$\boldsymbol{\theta}^t = \begin{bmatrix} \theta_1^t \\ \vdots \\ \theta_N^t \end{bmatrix} \quad N\text{-sample iid } \sim p^t$$

NB: In MCMC single *vs.* parallel chains has a long history!

Estimating $\int p^t \log f$ from parallel chains

For each "slice of time" (MCMC iteration) t, use the sample $\boldsymbol{\theta}^t = (\theta_1^t, \ldots, \theta_N^t)$ iid$\sim p^t$ for:

- Estimation of $\int p^t \log f$ directly from the SLLN

$$\widehat{p_N^t}(\log f) = \frac{1}{N} \sum_{i=1}^{N} \log f(\theta_i^t)$$

- But in Bayesian setup $f(\cdot) \propto \phi(\cdot)$, only ϕ is available,

$$\widehat{p_N^t}(\log \phi) = \frac{1}{N} \sum_{i=1}^{N} \log \phi(\theta_i^t) \to \int p^t \log \phi$$

  is accessible (see later)

Note: the shape f or ϕ is used in the estimate in both cases

Estimation of the entropy for a multivariate density

From a N-sample $\boldsymbol{X} = (X_1, \ldots, X_N)$ iid$\sim$ p over $\mathbb{R}^d$

Methods using plug-in of a nonparametric estimate (KDE) $\hat{p}_{\boldsymbol{X}}$ of p
Survey: Beirlant et al. 1997

- integral estimates: numerical integration $\int_{A_N} \hat{p}_{\boldsymbol{X}} \log \hat{p}_{\boldsymbol{X}}$
- resubstitution estimate: MC integration $\frac{1}{N} \sum_{i=1}^{N} \log \hat{p}_{\boldsymbol{X}}(X_i)$
- splitting data estimate: $\frac{1}{N/2} \sum_{i=1}^{[N/2]} \log \hat{p}_{\boldsymbol{Z}}(Y_i)$, $\boldsymbol{X} = \boldsymbol{Y} \cup \boldsymbol{Z}$
- Nearest Neighbor (NN) estimates: more on that later!

Most of these estimates require (smoothness) conditions not appropriate in MCMC context!

KDE and Hastings-Metropolis samplers

Based on Györfi & Van Der Meulen (1989) splitting estimate:

- split θ^t in two $[N/2]$-subsamples $\boldsymbol{Y}_N^t$ and $\boldsymbol{Z}_N^t$
- compute $\hat{p}^t_{\boldsymbol{Z}_N}(\theta) =$ a Kernel Density Estimate (KDE) of p^t,

$$\hat{p}^t_{\boldsymbol{Z}_N}(\theta) = \frac{1}{h_N{}^d N/2} \sum_{i=1}^{[N/2]} K\left(\frac{\theta - Z_i^t}{h_N}\right)$$

  for a kernel K, bandwidth h_N and usual assumptions
- Monte-Carlo integration using $\boldsymbol{Y}_N^t$, for $a_N > 0$ and $\lim_{N\to\infty} a_N = 0$

$$\hat{\mathcal{H}}_N^G(p^t) = \frac{1}{[N/2]} \sum_{i=1}^{[N/2]} \log \hat{p}^t_{\boldsymbol{Z}_N}(Y_i^t)\, \mathbb{I}_{\left\{\hat{p}^t_{\boldsymbol{Z}_N}(Y_i^t) \geq a_N\right\}}$$

Under some smoothness and tail conditions on f and the HM proposal, $\hat{\mathcal{H}}_N^G(p^t) \to \mathcal{H}(p^t)$ a.s. as $N \to \infty$ for $t \geq 1$ (C & Vandekerkhove 2012)

Problems: tuning of bandwidth h_N, treshold a_N, curse of dimension...

Nearest Neighbor (NN) functional estimates

Define the (Euclidean) distance from the ith point to its NN in $\boldsymbol{\theta}^t$,

$$\rho_i = \min\left\{d(\theta_i^t, \theta_j^t), j \in \{1, 2, \ldots, N\}, j \neq i\right\}$$

The NN estimate of $\mathcal{H}(p^t)$ is (Kozachenko & Leonenko 1987)

$$\hat{\mathcal{H}}_N(p^t) = -\left(\frac{d}{N}\sum_{i=1}^{N}\log(\rho_i) + \log(N-1) + \log(C_1(d)) + C_E\right)$$

C_E, $C_1(d)$ known.

Nearest Neighbor (NN) functional estimates

Define the (Euclidean) distance from the ith point to its NN in θ^t,

$$\rho_i = \min \left\{ d(\theta_i^t, \theta_j^t), j \in \{1, 2, \ldots, N\}, j \neq i \right\}$$

The NN estimate of $\mathcal{H}(p^t)$ is (Kozachenko & Leonenko 1987)

$$\hat{\mathcal{H}}_N(p^t) = -\left(\frac{d}{N} \sum_{i=1}^{N} \log(\rho_i) + \log(N-1) + \log(C_1(d)) + C_E \right)$$

C_E, $C_1(d)$ known.

- Generalization to k-Nearest Neighbor (e.g., Leonenko et al., 2005)
- Asymptotic unbiasedness and (weak) consistency for any d, under mild conditions s.a. $\int p |\log p|^{2+\varepsilon} < \infty$
- **No tuning parameters!**, compares successfully with KDE-based entropy estimate in our experiments

Outline: Next up...

Outline: Next up...

Content of the `EntropyMCMC` package:

Title: **MCMC evaluation through entropy estimation**

- Predefined target distributions and standard MCMC samplers
- Easy definition of user target & samplers
- Functions for running simulations, estimating entropy and Kullback's, methods for visualizing results, comparing samplers
- NN and KDE codes written in C
- MCMC parallel simulation can be done within the package, or imported from external file
- Two possible strategies:
  1. parallel simulation up to time n, *then* Kullback estimation
  2. simultaneous simulation+estimation for each t, forgetting the past
- HPC implementations using `parallel`, `snow`, snow MPI cluster (`Rmpi`), or singlecore version

Definition of a MCMC sampler

A list of named elements:

- `q_pdf`: the proposal density $q(y|x)$
- `q_proposal`: the function that simulates a proposal $\sim q(\cdot|x)$
- `step`: the function for simulation of 1 step $\theta^t \to \theta^{t+1}$

NB: the target f is not in the definition

Example for a Random Walk Hastings-Metropolis:

```
> RWHM$step
function (theta,                              # current position
          target_f,                           # target density
          q_pdf=gaussian_pdf,                 # default proposal
          q_proposal=gaussian_proposal,
          f_param, q_param)                   # f & q parameters
{...}
```

MCMC samplers

Some common MCMC's currently implemented:

- `RWHM`: Random Walk Hasting-Metropolis
  (with default gaussian proposal)
- `HMIS_norm`: Independence Sampler HM with gaussian proposal
  (from which any HMIS can be defined)
- `AMHaario`: Adaptive-Metropolis from Haario (2001)

MCMC samplers

Some common MCMC's currently implemented:

- `RWHM`: Random Walk Hasting-Metropolis
  (with default gaussian proposal)
- `HMIS_norm`: Independence Sampler HM with gaussian proposal
  (from which any HMIS can be defined)
- `AMHaario`: Adaptive-Metropolis from Haario (2001)

$+$ `IID_norm`: a "fake" MCMC $=$ Gaussian iid sampler for which
$p^t \equiv f = \mathcal{N}_d(\mu, \Sigma)$ for all $t \geq 1$

N chains for n iterations $\Leftrightarrow$ n iid replications of N-samples iid$\sim f$
$\rightarrow$ estimation of the (known) entropy $\mathcal{H}(\mathcal{N}_d(\mu, \Sigma))$ with replications
$\rightarrow$ study of MSE, bias ...

MCMC target and proposal

- Target definition  template

```
target_f <- function(theta, parameter){...}
```

where

  - `theta` is a $(n \times d)$ matrix of n d-dim points (vectorized evaluation preferable)
  - `parameter` is a list holding all the (hyper-) parameters, including the data in Bayesian framework where $f(\theta) \propto \ell(\boldsymbol{x}|\theta)\pi(\theta)$

- Proposal density $q(y|x)$ definition

```
proposal_pdf <- function(next, current, parameter){...}
```

I: Simulation first and then Kullback estimation

Step 1: Simulate N iid chains from some θ^0

```
MCMCcopies.mc(mcmc_algo,      # MCMC definition (list)
              n=100, nmc=N,   # for n iter. and N chains
              Ptheta0,        # N d-dim starting values
              target, f_param,# f and its parameters
              q_param,        # proposal parameters
              nbcores=detectCores()) # multicore version here
```

$\rightarrow$ object of class `"plMCMC"` holding the MCMC defs & the N simulated chains in an array ($n \times d \times N$) . . . that can be huge!

I: Simulation first and then Kullback estimation

Step 1: Simulate N iid chains from some θ^0

```
MCMCcopies.mc(mcmc_algo,      # MCMC definition (list)
          n=100, nmc=N,       # for n iter. and N chains
          Ptheta0,            # N d-dim starting values
          target, f_param,    # f and its parameters
          q_param,            # proposal parameters
          nbcores=detectCores()) # multicore version here
```

$\rightarrow$ object of class `"plMCMC"` holding the MCMC defs & the N simulated chains in an array ($n \times d \times N$) . . . that can be huge!

Step 2: Compute estimates $\widehat{\mathcal{H}}_N(p^t)$ and $\widehat{\mathcal{K}}_N(p^t, f)$ from this object

```
EntropyMCMC.mc(plmcmc1,                 # result from above
         method="Nearest.Neighbor", # or "KDE" or...
         nbcores=detectCores())     # parallel version
```

$\rightarrow$  object of class `"KbMCMC"` with associated methods (`plot`,. . .)

II: Simulation and Kullback estimation simultaneously

Advantages: Storing only $\boldsymbol{\theta}^t$ at current t

- parallel simulation of the N next steps: $\boldsymbol{\theta}^t \to \boldsymbol{\theta}^{t+1}$
- (parallel) estimation of $\widehat{\mathcal{H}}_N(p^{t+1})$ and $\widehat{\mathcal{K}}_N(p^{t+1}, f)$
- $\boldsymbol{\theta}^t \leftarrow \boldsymbol{\theta}^{t+1}$ forgetting the past!

Note: for AMCMC, needed sufficient statistics from the past need to be stored as well (e.g., empirical covariance matrix,...)

II: Simulation and Kullback estimation simultaneously

Advantages: Storing only θ^t at current t

- parallel simulation of the N next steps: $\theta^t \to \theta^{t+1}$
- (parallel) estimation of $\widehat{\mathcal{H}}_N(p^{t+1})$ and $\widehat{\mathcal{K}}_N(p^{t+1}, f)$
- $\theta^t \leftarrow \theta^{t+1}$ forgetting the past!

Note: for AMCMC, needed sufficient statistics from the past need to be stored as well (e.g., empirical covariance matrix,...)

```
EntropyParallel.cl(mcmc_algo,        # MCMC definition (list)
     n=100, nmc=N, Ptheta0,          # like before
     target, f_param, q_param,       # f and q parameters
     method="Nearest.Neighbor",      # KDE available as well
     cltype="PAR_SOCK",nbnodes=4)    # cluster type
```

HPC implementations: socket cluster with `parallel`, socket cluster with `snow`, snow MPI cluster with `Rmpi`, or singlecore version

$\to$  object of class `"KbMCMC"` with associated methods (`plot`,...)

Outline: Next up...

Example: A synthetic multidimensional mixture model

- Target f: A 3 components mixture of d-dimensional Gaussian,

$$f(\boldsymbol{\theta}) = \sum_{j=1}^{3} \lambda_j \mathcal{N}_d(\mu_j, \Sigma_j)(\boldsymbol{\theta}),$$

with parameters

$$\begin{aligned} \text{weights} &: \lambda_1 = \lambda_2 = \lambda_3 = 1/3 \\ \text{means} &: \mu_1 = 0\mathbf{1}_d, \quad \mu_2 = 4\mathbf{1}_d, \quad \mu_3 = -4\mathbf{1}_d \\ \text{covariances} &: \Sigma_j = j\mathbb{I}_d, \quad j = 1, 2, 3 \end{aligned}$$

Example: A synthetic multidimensional mixture model

■ Target f: A 3 components mixture of d-dimensional Gaussian,

$$f(\boldsymbol{\theta}) = \sum_{j=1}^{3} \lambda_j \mathcal{N}_d(\mu_j, \Sigma_j)(\boldsymbol{\theta}),$$

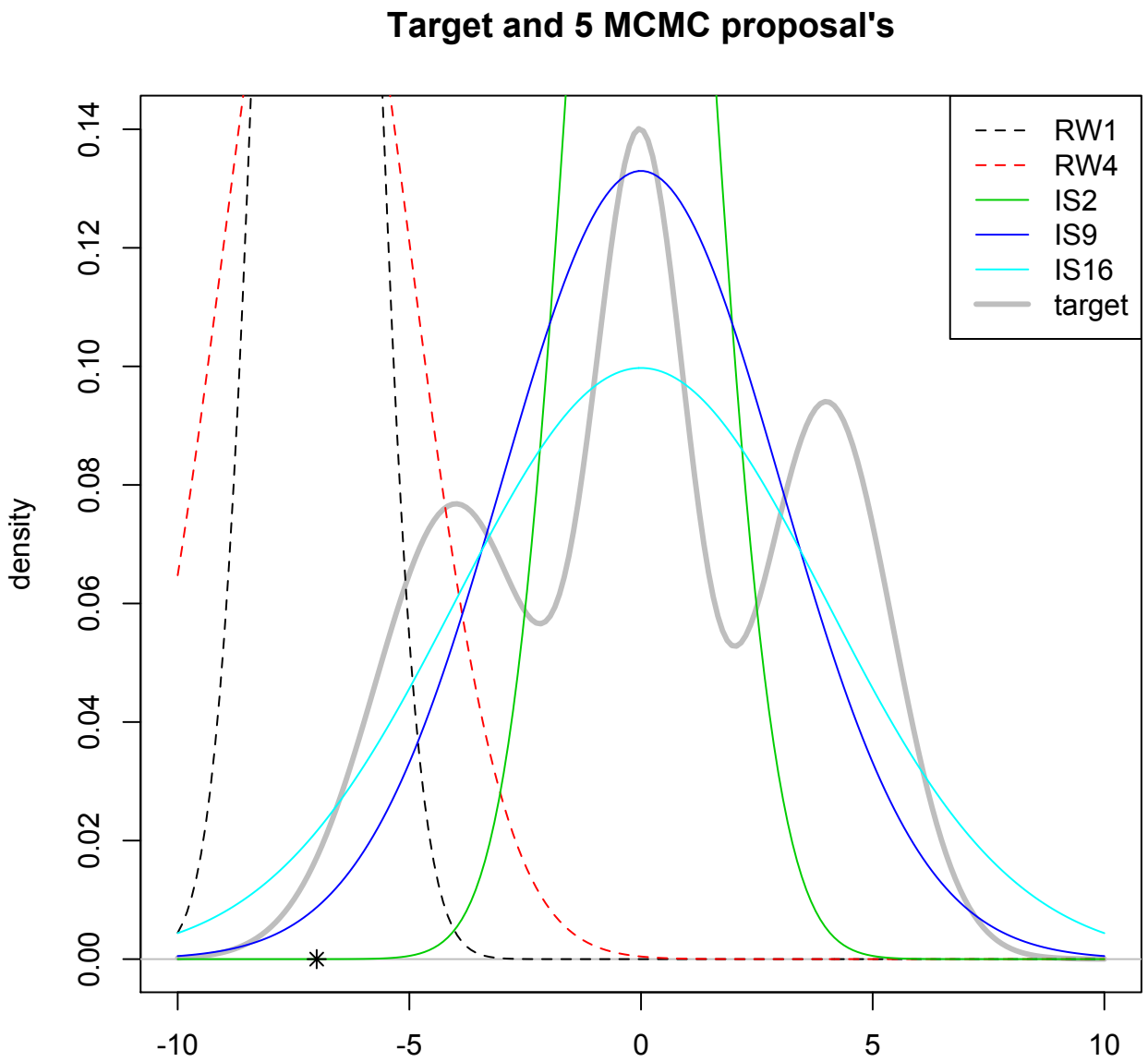
with parameters

$$\begin{aligned} \text{weights} &: \lambda_1 = \lambda_2 = \lambda_3 = 1/3 \\ \text{means} &: \mu_1 = 0\mathbf{1}_d, \quad \mu_2 = 4\mathbf{1}_d, \quad \mu_3 = -4\mathbf{1}_d \\ \text{covariances} &: \Sigma_j = j\mathbb{I}_d, \quad j = 1, 2, 3 \end{aligned}$$

■ 5 MCMC samplers:

- 2 Random-Walk HM with Gaussian proposal variances: $1\mathbb{I}_d$ (RW1) and $4\mathbb{I}_d$ (RW4)
- 3 HM Independence Samplers "ISσ^2" with proposal $\mathcal{N}_d(\mathbf{0}, \sigma^2\mathbb{I}_d)$: IS2, IS9 and IS16

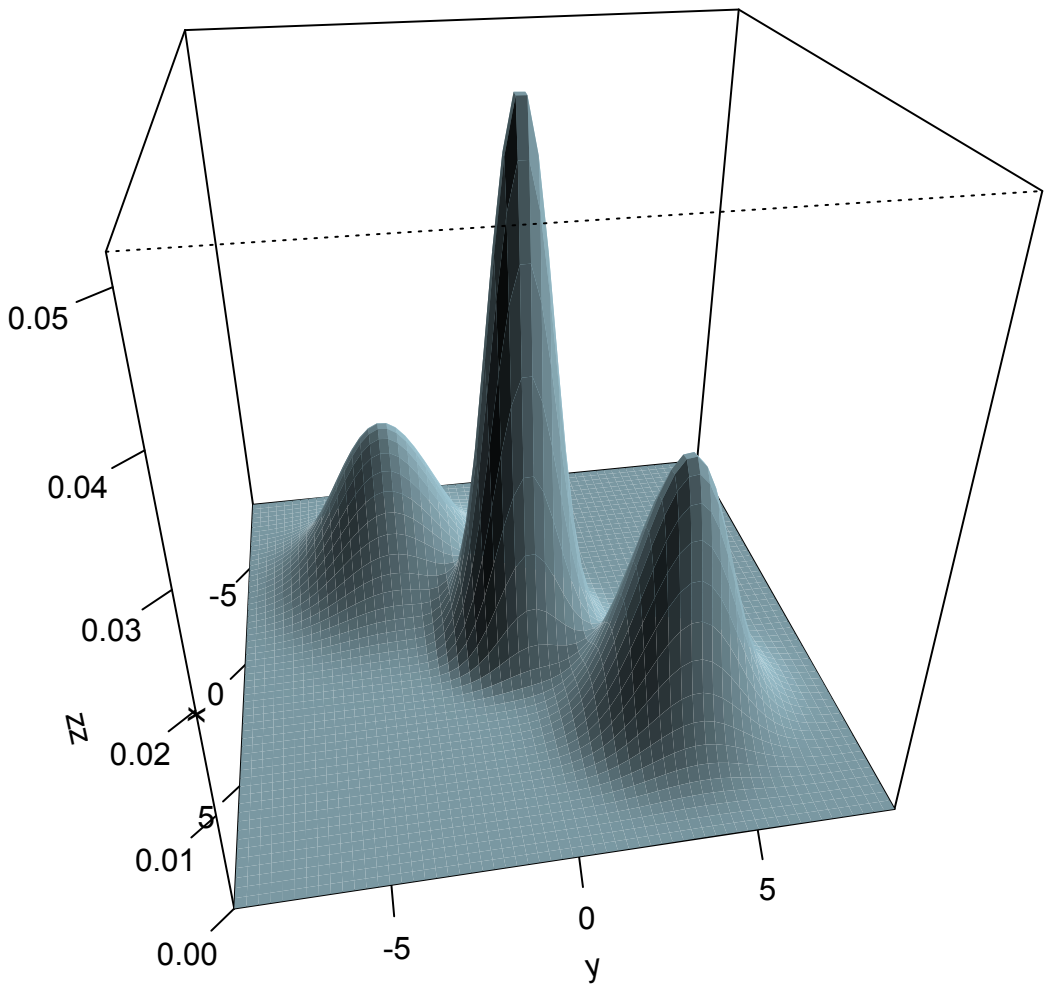
Target f and proposals $q(\cdot|x)$ visualization



Sample targets
$d = 1$: overlapping

IS2 cannot converge!
distance$(\mu_1, \mu_2) = 4$

Target f and proposals $q(\cdot|x)$ visualization



Sample targets

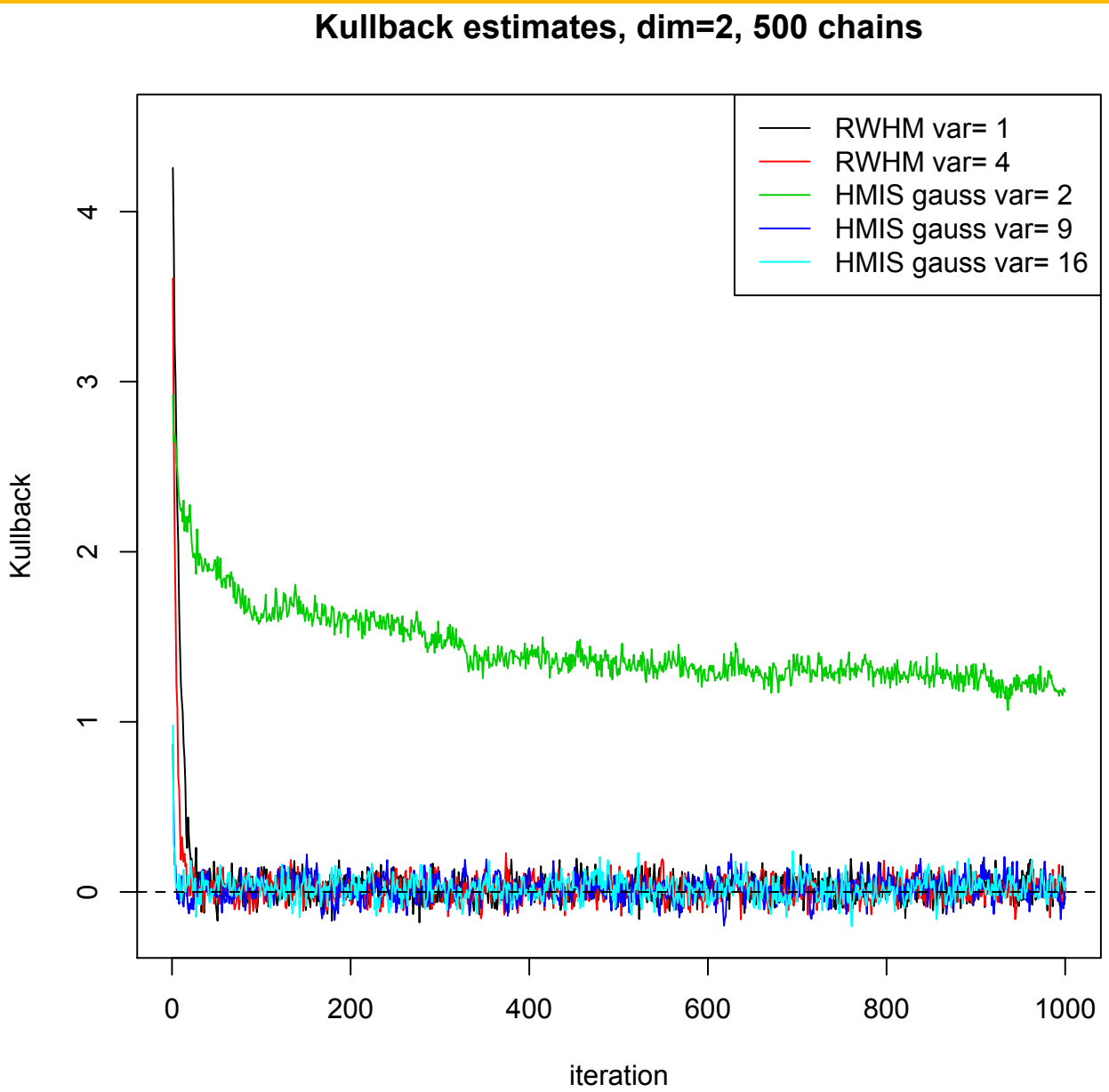
$d = 2$: less overlapping
distance$(\mu_1, \mu_2) = 5.67$

and so on...

Simulation & entropy estimation typical coding

```
library(snow); library(Rmpi) # for MPI cluster
library(EntropyMCMC)
nbnodes <- mpi.universe.size() # max available

n=10000; nmc=500 # iterations & nb of chains N
Ptheta0 <- DrawInit(nmc,d,initpdf="rnorm",mean=0,sd=5)

varq=1 # proposal parameters definition for RW1
q_param=list(mean=rep(0,d), v=varq*diag(d))

# Simulation/entropy+Kullback estimation/vizualization
RW1 <- EntropyParallel.cl(RWHM, n, nmc, Ptheta0,
                          target, target_param, q_param,
                          cltype="SNOW_RMPI", nbnodes)

plot(RW1)    # S3 method for plotting "KbMCMC" object
```

$d = 2$ mixture model: $\mathcal{K}(p^t, f)$ comparisons



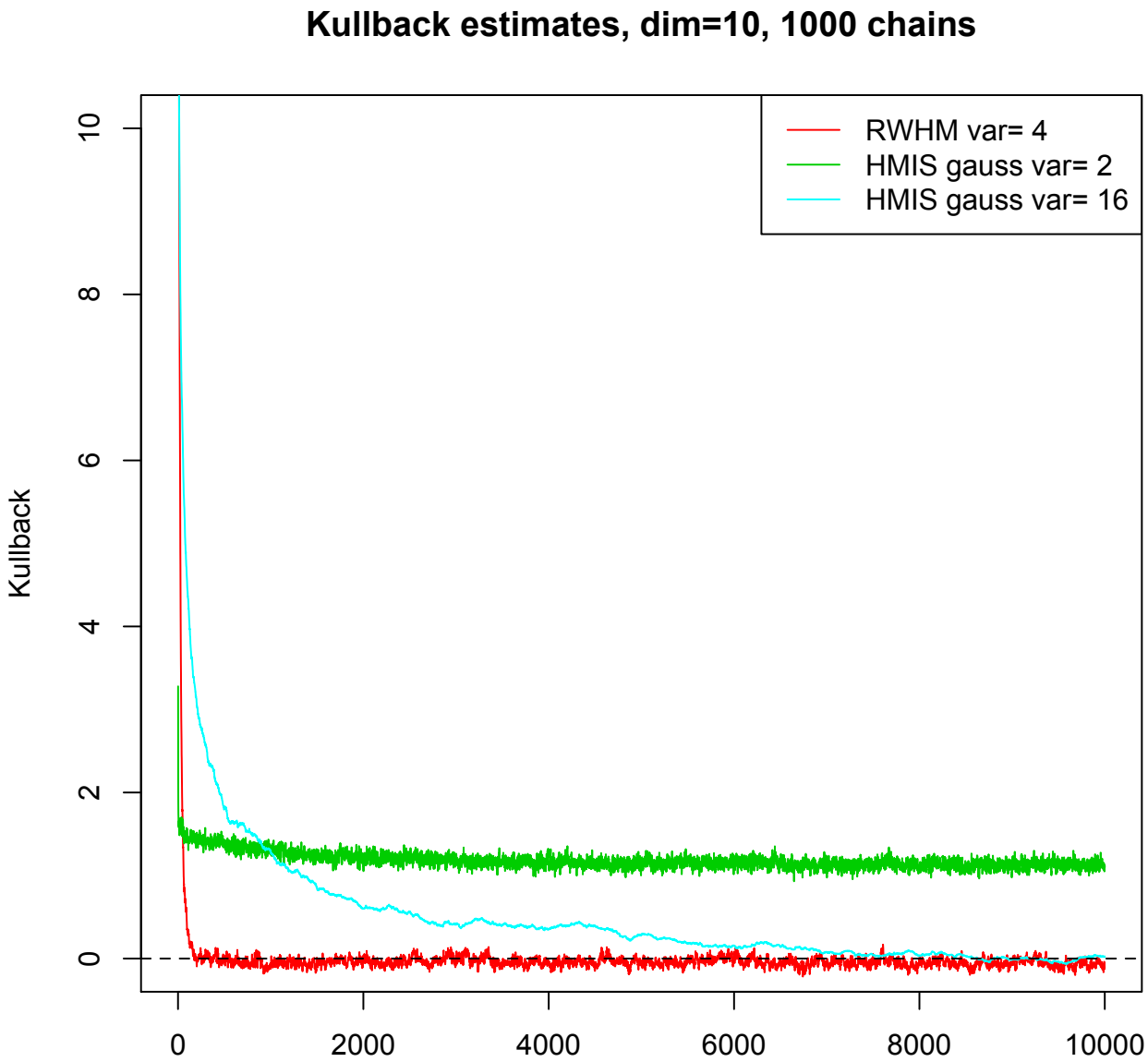
$N = 500$ iid chains

- Stabilizations ≈ 0 give convergence time
- decays give MCMC's performances
- IS2 is not converging

All the others similar

[12 cores time: 24mn]

$d = 10$: $\mathcal{K}(p^t, f)$ comparisons



Notice that now:

IS16 slower than before

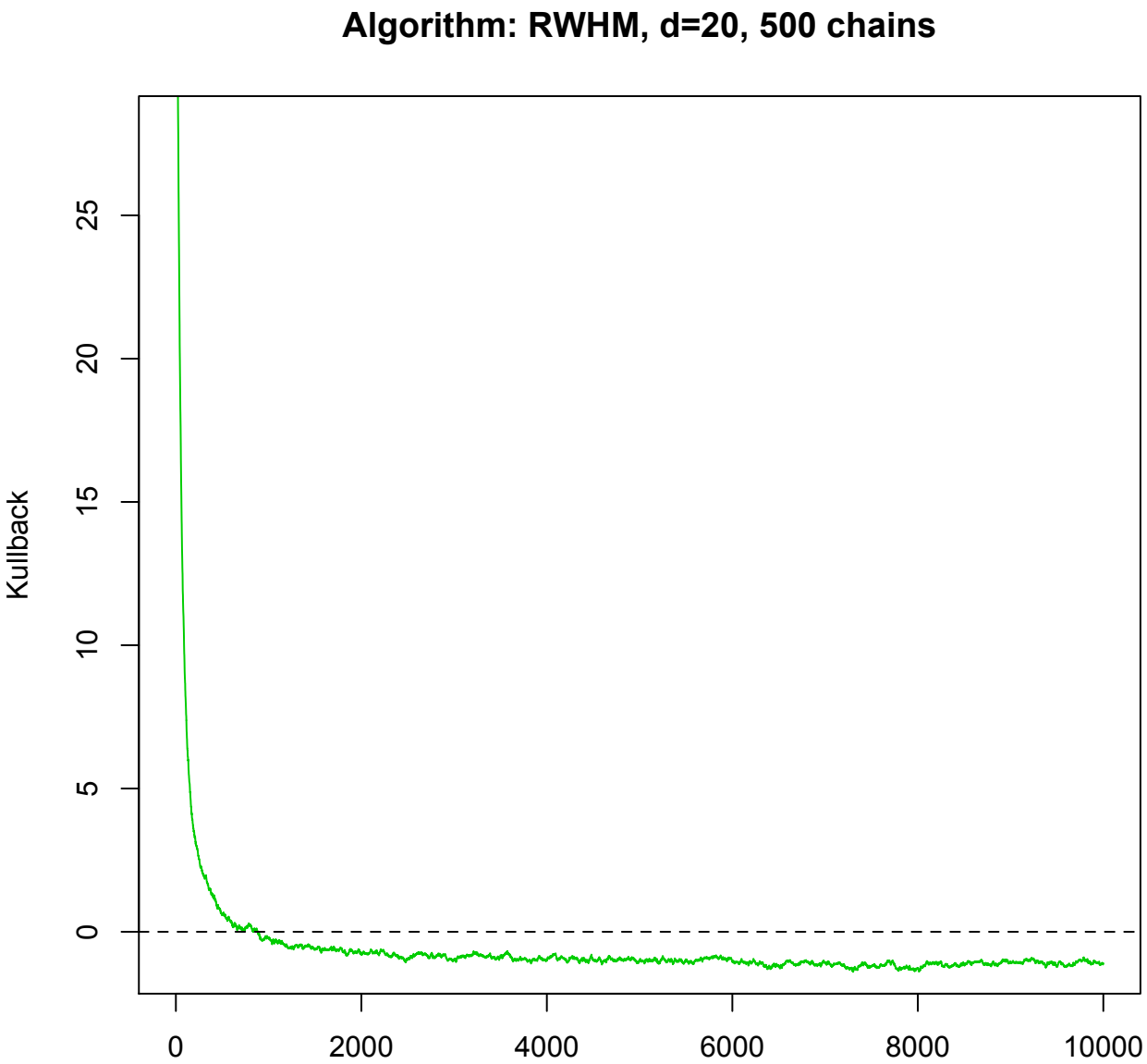
IS2 still non convergent

The behaviors change with dimension!

"only" $N = 1000$ chains provide a diagnostic

[12 cores time: 54mn]

$d = 20$: the curse of dimension!




```
plot(RW4,col="green")
```

Why does $\mathcal{K}(p^t, f)$ stabilizes at a negative value?

Bias in $\mathcal{H}(p^t)$ estimation! when d get's large

[12 cores time: 27mn]

Curse of dimension and bias in $\mathcal{H}(p)$ np estimates

Effect of dimension on bias, already noticed in recent literature

Facts and numerical evidence

- The variance decreases as $\mathcal{O}(N^{-1})$
- The bias decreases as $\mathcal{O}(N^{-1/(d+1)})$, a "glacially slow" rate
- both for Kernel density and NN-based estimates

Stowell and Plumbley (2009); Sricharan *et al*, arxiv (2013),...

Confirmed in our case using the `IID_norm` sampler for $\mathcal{H}(\mathcal{N}_d(\mathbf{0}, \Sigma))$

Curse of dimension and bias in $\mathcal{H}(p)$ np estimates

Effect of dimension on bias, already noticed in recent literature

Facts and numerical evidence

- The variance decreases as $\mathcal{O}(N^{-1})$
- The bias decreases as $\mathcal{O}(N^{-1/(d+1)})$, a "glacially slow" rate
- both for Kernel density and NN-based estimates

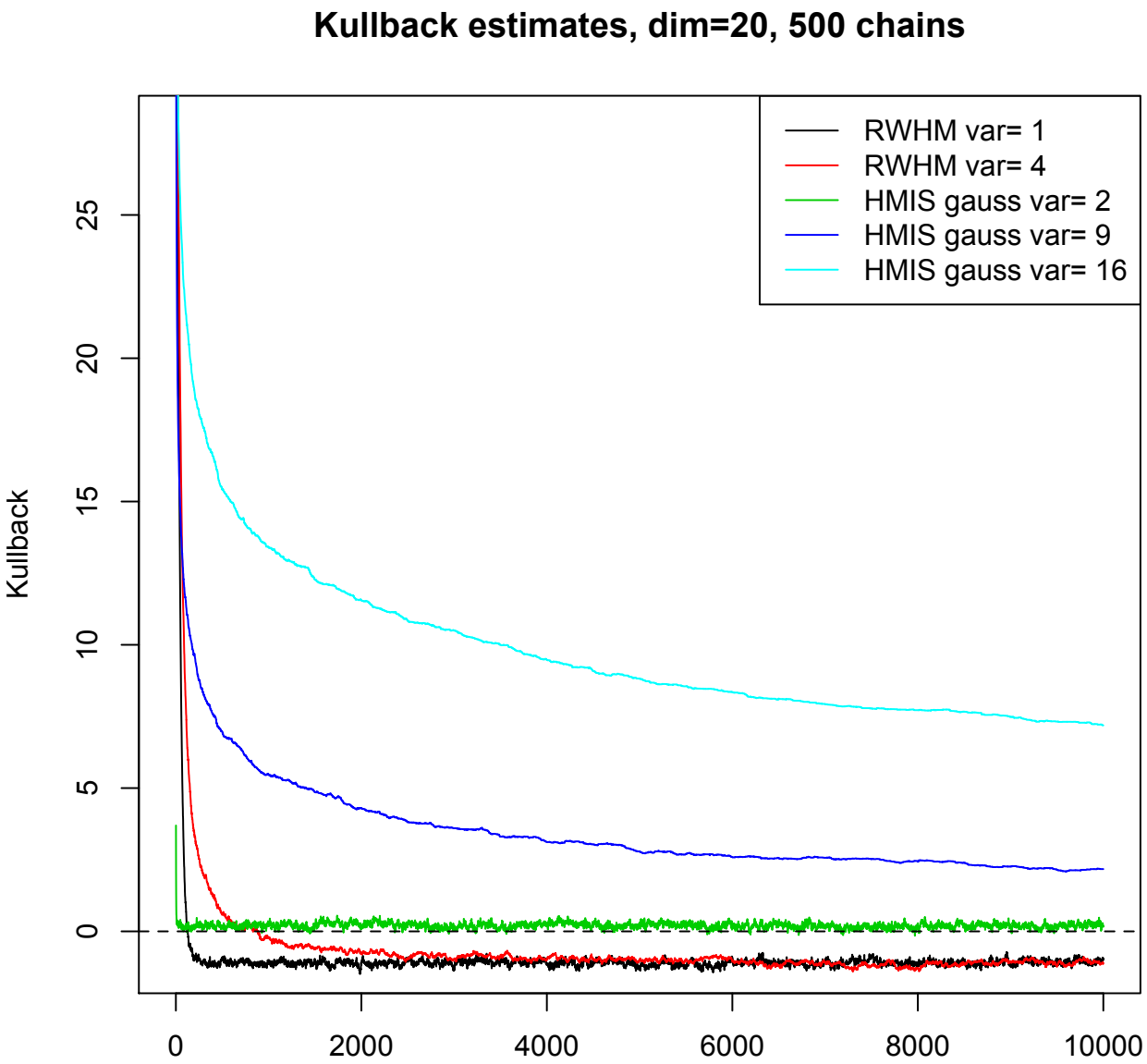
Stowell and Plumbley (2009); Sricharan *et al*, arxiv (2013),...

Confirmed in our case using the `IID_norm` sampler for $\mathcal{H}(\mathcal{N}_d(\mathbf{0}, \Sigma))$

But we are only interested in decay and stabilization

No need to simulate thousands of MCMC copies (not feasible anyway!)

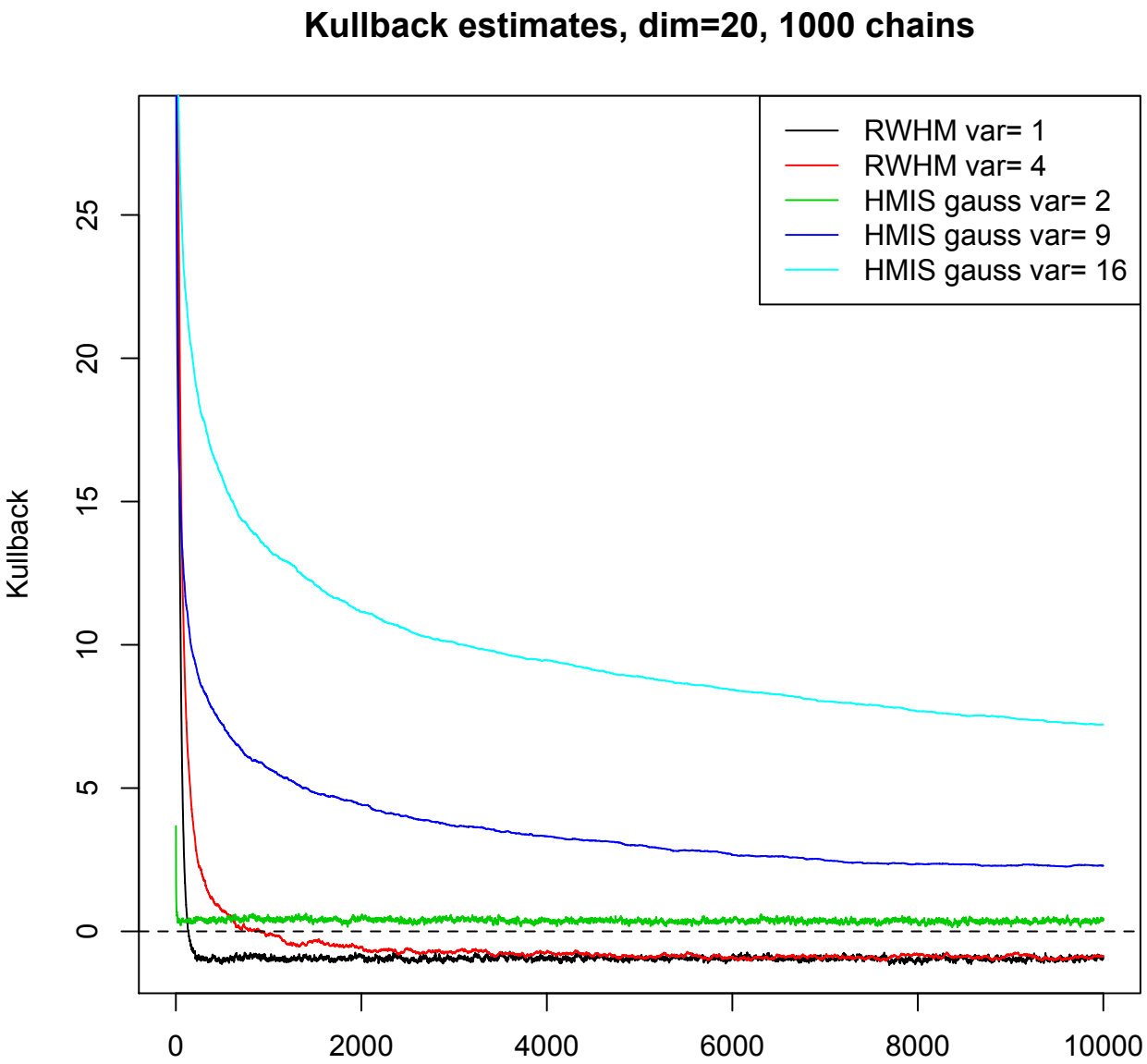
$d = 20$ mixture model: bias and decision criterion



$N = 500$ iid chains:

- IS2 cannot converge!
- Don't look for stabilization ≈ 0

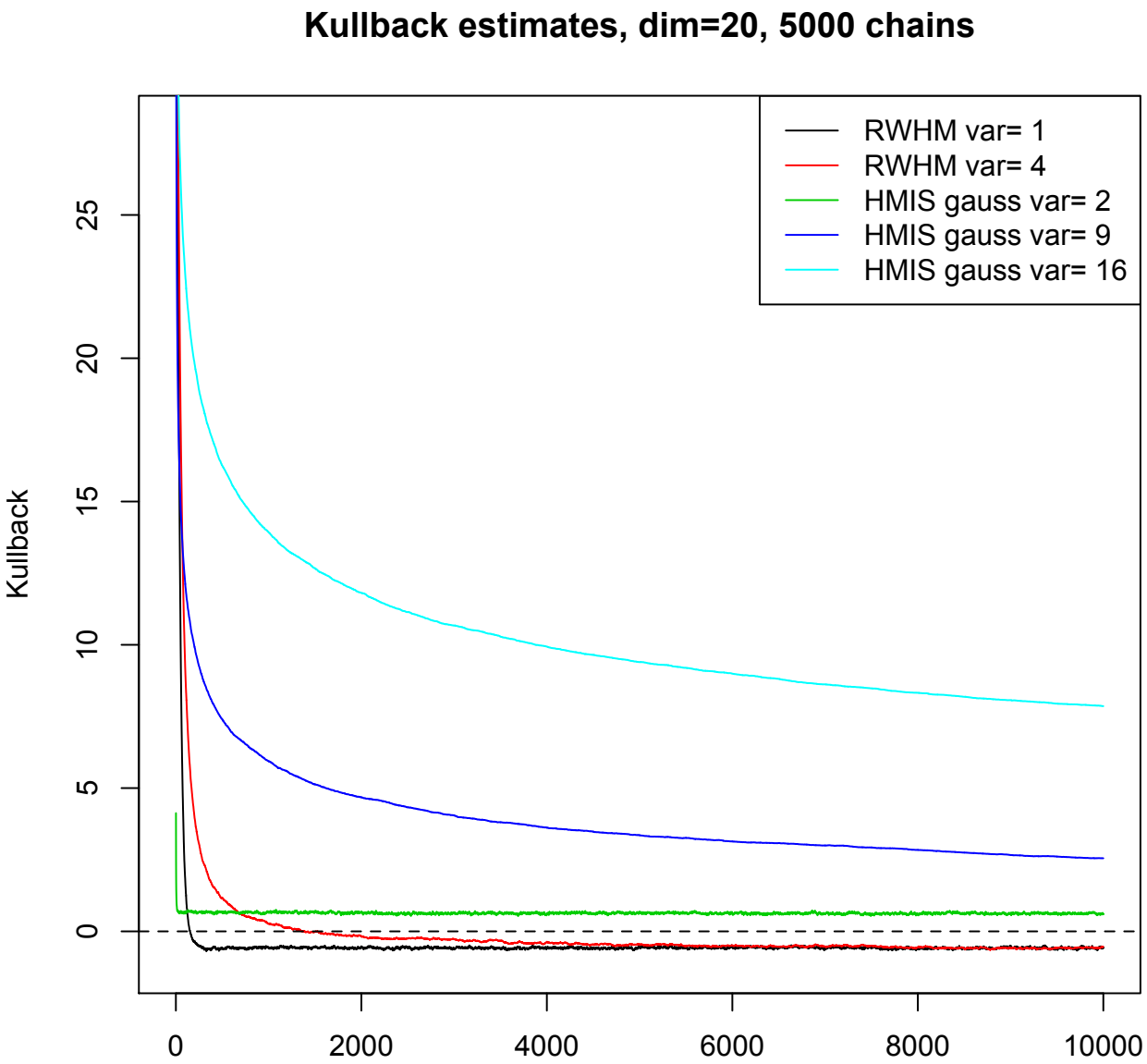
$d = 20$ mixture model: bias and decision criterion



- IS2 cannot converge!
- Don't look for stabilization ≈ 0

$N = 1000$ iid chains

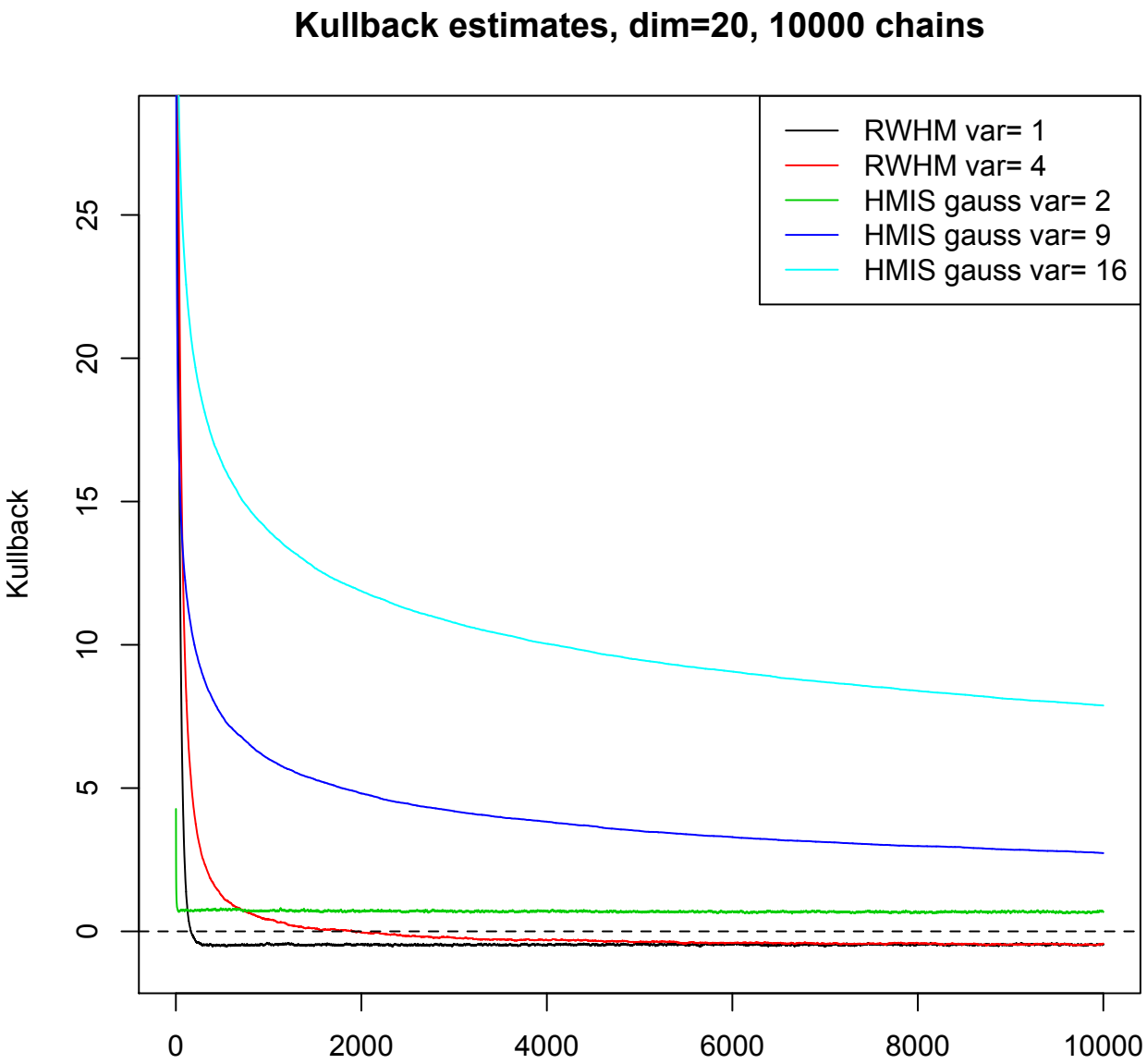
$d = 20$ mixture model: bias and decision criterion



- IS2 cannot converge!
- Don't look for stabilization ≈ 0

$N = 5000$ iid chains

$d = 20$ mixture model: bias and decision criterion



- IS2 cannot converge!
- Don't look for stabilization ≈ 0

$N = 10,000$ iid chains!

- Similar decision than for $N = 500$

both RW's serve as **benchmark convergent MCMC**'s here

Methodological approach for large dimension

In practical situations:

- Trying to achieve the asymptotic unbiasedness is hopeless
- N can be made large enough in practice so that:
  - the variance becomes neglictible, i.e. graphical comparison OK
  - but some $\text{bias}_N(p^t, d)$ still remains
- $f(\theta) = C(f)\,\phi(\theta)$ where $C(f)$ is the unknown normalizing constant
- A converging MCMC strategy stabilizes $\approx \log C(f) + \text{bias}_N(f, d)$
- A non-converging MCMC ($p^t \to g \neq f$) can stabilize anywhere!

Diagnostic using a benchmark MCMC known to converge

Identifying the proper stabilization value including bias, it allows:

- performance comparison against other MCMC strategies
- (non-) convergence assessment of other MCMC strategies

Outline: Next up...

Some typical CPU times

All the examples have been ran on:

- a 12 cores workstation (2 Intel Xeon CPU X5650 @ 2.67GHz)
- The Regional cluster CCSC using OpenMPI

Centre de Calcul Scientifique en région Centre `http://cascimodot.fdpoisson.fr/?q=ccsc`

Total (user + system) times for the d-dim mixture example and $n = 10,000$ iterations, 12 cores single workstation:

	$N = 500$	$N = 1000$
$d = 2$	24 mn	
$d = 10$		54 mn
$d = 20$	27 mn	
$d = 30$	32 mn	

NB: more than 6 times faster than using a singlecore version

Conclusions and perspectives

Pro's:

- HPC implementation available for multicore computers, cloud computing (snow), and actual clusters (Rmpi)
- simulations (or part of it) from the best sampler are recyclable after comparisons, or can be re-used in the fly for statistical inference
- Practical, easy-to-understand graphical criterion

Con's:

- heavy computational load for high dimension (what is *high*?)
- needs a benchmark MCMC when d gets large (say $d \geq 10$)

Ongoing work and perspectives:

- Dimension reduction through PCA and crossed-entropy estimation
- Extension to k-NN entropy estimation (Singh et al. 2003) Pbs: choice of k, bias reduction unclear...

The end

THANKS FOR YOUR ATTENTION !

For Further Reading...

Y. Bai, R. V. Craiu, and A. F. Di Narzo. Divide and conquer: A mixture-based approach to regional adaptation for mcmc. *J. Comp. Graph. Stat.*, pages 1–17, 2010.

J. Beirlant, E. J. Dudewicz, L. Györfi, and E. C. van der Meulen. Nonparametric entropy estimation, an overview. *Int. J. Math. Stat. Sci.*, 6:17–39, 1997.

Chauveau, D. and Vandekerkhove, P. A Monte Carlo estimation of the entropy for Markov chains. *Method. & Comput. in Appl. Probab.* 9 (1): 133–149, 2007.

Chauveau, D. and Vandekerkhove, P. Smoothness of Metropolis-Hastings algorithm and application to entropy estimation. *ESAIM: Probability and Statistics* 17 (1), 2013.

Chauveau D. and Vandekerkhove, P. (2014), Simulation Based Nearest Neighbor Entropy Estimation for (Adaptive) MCMC Evaluation, In *JSM Proceedings*, Statistical Computing Section. Alexandria, VA: American Statistical Association. 2816–2827.

Fort, G, Moulines, E, Priouret, P. and Vandekerkhove, P. A Central Limit Theorem for Adaptive and Interacting Markov Chains *Bernouilli* 2012 (to appear).

L. Györfi and E. C. Van Der Meulen. An entropy estimate based on a kernel density estimation. *Colloquia Mathematica societatis János Bolyai 57, Limit Theorems in Probability and Statistics Pécs*, pages 229–240, 1989.

L. Kozachenko and N. N. Leonenko. Sample estimate of entropy of a random vector. *Problems of Information Transmission*, 23:95–101, 1987.

H. Singh, N. Misra, V. Hnizdo, A. Fedorowicz, and E. Demchuk. Nearest neighbor estimate of entropy. *American Journal of Mathematical and Management Sciences*, 23(3):301–321, 2003.

J. A. Vrugt, C. Braak, C. Diks, B. A. Robinson, J. M. Hyman, and D. Higdon. Accelerating markov chain monte carlo simulation by differential evolution with self-adaptive randomized subspace sampling. *International Journal of Nonlinear*