Contraction of coupled Hamiltonian Monte Carlo

JH, PEJ

August 2017

Setting

This script accompanies the article "Unbiased Hamiltonian Monte Carlo with couplings", by Jeremy Heng and Pierre E. Jacob.

This script illustrates how a pair of Hamiltonian Monte Carlo chains can contract if given the same velocity at each step.

We begin by loading the package, registering multiple cores, setting the random number generator, etc.

```
library(debiasedhmc)
library(gridExtra)
rm(list = ls())
set.seed(18)
setmytheme()
```

Target distribution

We then define a Normal target distribution of dimension 250, and define the initial distribution of the chains to be the target.

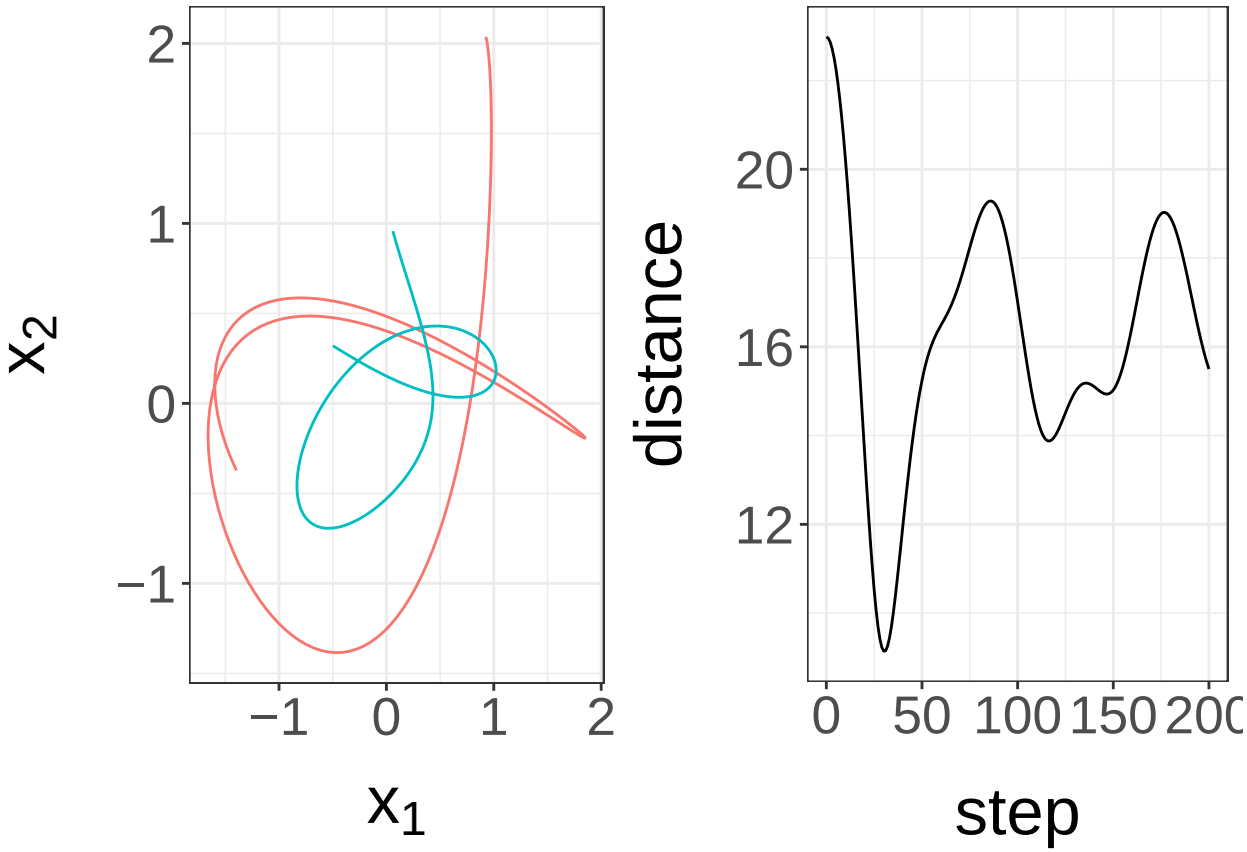
```
# define target of dimension p
dimension <- 250
mean_target <- rep(0, dimension)
Sigma_target <- diag(1, dimension, dimension)
for (i in 1:dimension) {
    for (j in 1:dimension) {
        Sigma_target[i, j] <- exp(-abs(i - j))
    }
}
target <- get_mvnormal(dimension, mean_target, Sigma_target)
# initial distribution of chains
rinit <- function() fast_rmvnorm(1, mean_target, Sigma_target)
```

Long Hamiltonian trajectories

Let's generate pairs of long Hamiltonian trajectories with a common initial velocity. We then plot the first two first components of the trajectories, and the Euclidean distance between the trajectories.

```
# leap-frog step size
stepsize <- 0.05
# trajectory length
trajectorylength <- 10
# number of leap-frog steps
nsteps <- floor(trajectorylength/stepsize)
# load HMC kenrel
hmc_kernel <- get_hmc_kernel(target$logtarget, target$gradlogtarget, stepsize,
```

```
    nsteps, dimension)
# initial positions
x1 <- rinit()
x2 <- rinit()
# draw coupled trajectories
result <- hmc_kernel$coupled_kernel(x1, x2)
# trajectory of particle 1
xtraj1 <- result$xtraj1
# trajectory of particle 2
xtraj2 <- result$xtraj2
# transformation before plotting
xtraj1.df <- data.frame(xtraj1)
xtraj2.df <- data.frame(xtraj2)
# plot of the 2 first marginals
g <- ggplot(xtraj1.df, aes(x = X1, y = X2, colour = "trajectory 1")) + geom_path() +
    theme(legend.position = "none")
g <- g + geom_path(data = xtraj2.df, aes(colour = "trajectory 2"))
g <- g + xlab(expression(x[1])) + ylab(expression(x[2]))
distances <- sapply(1:(nsteps + 1), FUN = function(index) sqrt(sum((xtraj1[index,
    ] - xtraj2[index, ])^2)))
gdistance <- qplot(x = 0:nsteps, y = distances, geom = "line") + xlab("step") +
    ylab("distance")
grid.arrange(g, gdistance, nrow = 1)
```



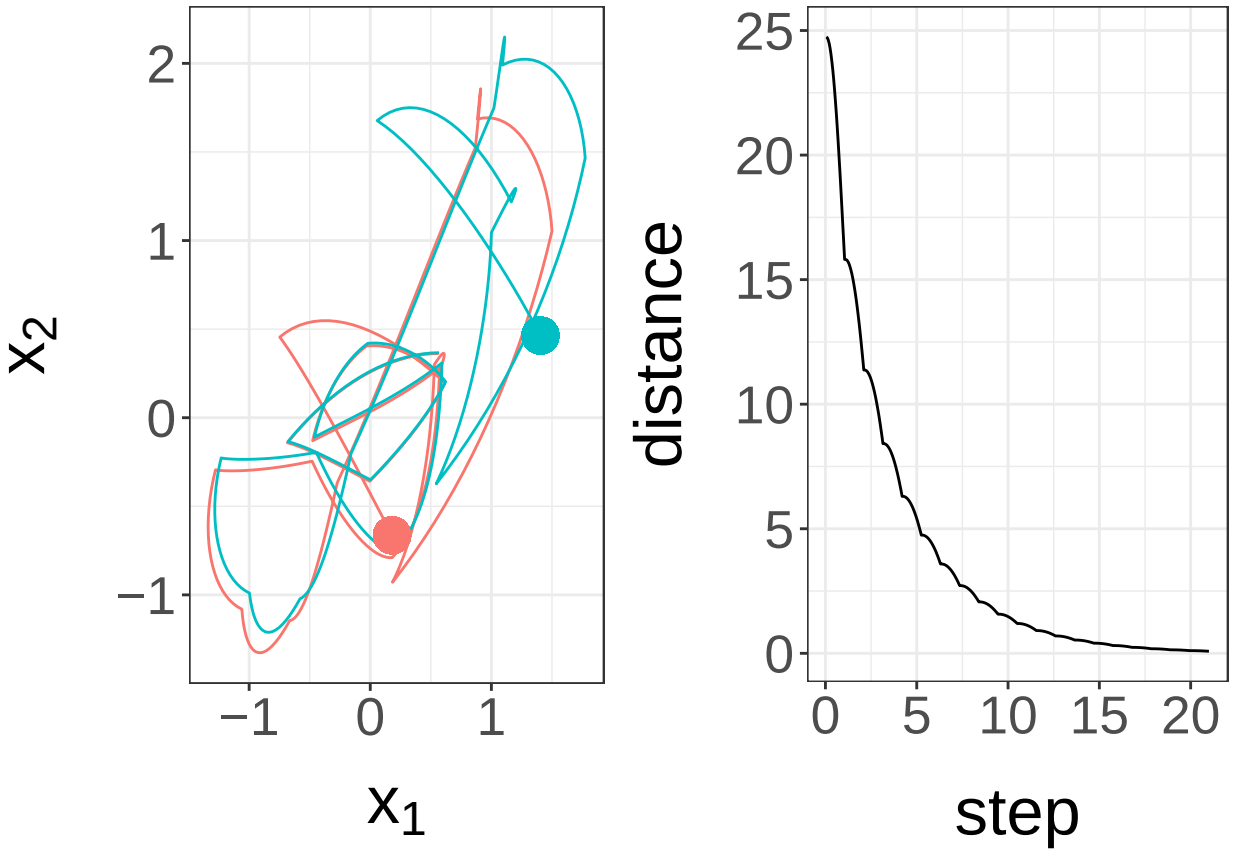
The trajectories seem to be doing their own thing, and the distance between them fluctuates... but note that the distance initially decreases.

Iterating Hamiltonian trajectories

Now we look at what happens when we iterate coupled HMC steps, each of them using a common initial velocity.

```
# number of trajectories
ntrajectories <- 20
# trajectory length
trajectorylength <- 1
# number of leap-frog steps
nsteps <- floor(trajectorylength/stepsize)
# load HMC kernels
hmc_kernel <- get_hmc_kernel(target$logtarget, target$gradlogtarget, stepsize,
    nsteps, dimension)
# initial positions
x1 <- rinit()
x2 <- rinit()
xtraj1 <- matrix(nrow = ntrajectories * (nsteps + 1), ncol = dimension)
xtraj2 <- matrix(nrow = ntrajectories * (nsteps + 1), ncol = dimension)
# perform many short coupled trajectories
for (itraj in 1:ntrajectories) {
    result <- hmc_kernel$coupled_kernel(x1, x2)
    x1 <- result$chain_state1
    x2 <- result$chain_state2
    xtraj1[((itraj - 1) * (nsteps + 1) + 1):(itraj * (nsteps + 1)), ] <- result$xtraj1
    xtraj2[((itraj - 1) * (nsteps + 1) + 1):(itraj * (nsteps + 1)), ] <- result$xtraj2
}
# transform for plotting
xtraj1.df <- data.frame(xtraj1)
xtraj2.df <- data.frame(xtraj2)
xtraj1.df$iteration <- rep(1:ntrajectories, each = (nsteps + 1))
xtraj1.df$step <- rep(0:nsteps, times = ntrajectories)
xtraj2.df$iteration <- rep(1:ntrajectories, each = (nsteps + 1))
xtraj2.df$step <- rep(0:nsteps, times = ntrajectories)

g <- ggplot(xtraj1.df, aes(x = X1, y = X2, colour = "trajectory 1")) + geom_path() +
    theme(legend.position = "none")
g <- g + geom_path(data = xtraj2.df, aes(colour = "trajectory 2"))
g <- g + xlab(expression(x[1])) + ylab(expression(x[2]))
g <- g + geom_point(aes(x = xtraj1[1, 1], y = xtraj1[1, 2], colour = "trajectory 1"),
    size = 6)
g <- g + geom_point(aes(x = xtraj2[1, 1], y = xtraj2[1, 2], colour = "trajectory 2"),
    size = 6)
distances <- sapply(1:nrow(xtraj1), FUN = function(index) sqrt(sum((xtraj1[index,
    ] - xtraj2[index, ])^2)))
gdistance <- qplot(x = 1:nrow(xtraj1)/nsteps, y = distances, geom = "line") +
    xlab("step") + ylab("distance")
grid.arrange(g, gdistance, nrow = 1)
```



Now we see the trajectories getting close to one another. The initial positions are indicated by big dots. By follow the pair of trajectories one can see the contraction operating. It would be perhaps better illustrated by animations. The distance between the chains converges to zero at a geometric rate.