Tutorial: distance calculations

EB, PJ, MG, CR

March 21, 2017

Setting

This script compares different distances between data sets, in a multivariate, i.i.d. setting.

The model specifies $Y \sim \mathcal{N}(\mu, \Sigma)$, where Σ is a fixed covariance matrix. The prior on μ is $\mathcal{N}(0, 1)$ on each component.

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

```r
# load package
library(winference)
# register parallel cores
registerDoParallel(cores = detectCores())
# remove all
rm(list = ls())
# apply preferences for ggplotting
require(gridExtra)
theme_set(theme_bw())
# set RNG seed
set.seed(11)
```

Data and model

We define a multivariate Gaussian model and generate some data.

```r
# number of observations
dimension <- 5
target <- get_multivariate_normal(dimension)
target$parameters$S <- diag(1, dimension, dimension)
for (i in 1:dimension) {
    for (j in 1:dimension) {
        target$parameters$S[i, j] <- 0.5^(abs(i - j))
    }
}
nobservations <- 100
target$simulate <- function(theta) target$robservation(nobservations, theta,
    target$parameters, target$generate_randomness(nobservations))
# number of observations
true_theta <- rnorm(dimension)
obs <- target$simulate(true_theta)
# the observations are in a (ydim,nobservations) matrix
dim(obs)
```

```
## [1]   5 100
```

Distances

We consider four distances between multivariate samples:

- the *exact Wasserstein* distance, using the transport package (Schuhmacher, D., Bhre, B., Gottschlich, C. and Heinemann, F. (2017) transport: Optimal Transport in Various Forms),
- the *Sinkhorn distance* (Cuturi, M. (2013) Sinkhorn distances: lightspeed computation of optimal transport. In Advances in Neural Information Processing Systems (NIPS), 2292–2300),
- the *Hilbert distance*, based on the Hilbert space filling curve, proposed in our article (Bernton, E., Jacob, P. E., Gerber, M. and Robert, C. P. (2017) Inference in generative models using the Wasserstein distance. arXiv preprint arXiv:1701.05146),
- and finally the *swapping distance*, proposed in (Puccetti, G. (2017) An algorithm to approximate the optimal expected inner product of two vectors with given marginals. Journal of Mathematical Analysis and Applications, 451, 132–145).

Let's compute these distances between two data sets, playing a bit with the tuning parameters of the Sinkhorn distance, that is, the regularizer ε and the number of Sinkhorn iterations.

```
# generate a fake data set
fake_obs <- target$simulate(rnorm(dimension))
exact_transport_distance(obs, fake_obs, p = 1, ground_p = 2)
```

```
## Initial solution based on shortlist is degenerate. Adding 99 basis vector(s)... done.
```

```
## [1] 4.149958
```

```
sinkhorn_distance(obs, fake_obs, p = 1, ground_p = 2, eps = 0.1, niterations = 100)$corrected
```

```
## [1] 4.563124
```

```
sinkhorn_distance(obs, fake_obs, p = 1, ground_p = 2, eps = 0.01, niterations = 100)$corrected
```

```
## [1] 4.185117
```

```
sinkhorn_distance(obs, fake_obs, p = 1, ground_p = 2, eps = 0.01, niterations = 1000)$corrected
```

```
## [1] 4.18509
```

```
swap_distance(obs, fake_obs, p = 1, ground_p = 2, tolerance = 1e-5)$distance
```

```
## [1] 4.159995
```

```
hilbert_distance(obs, fake_obs, p = 1, ground_p = 2)
```

```
## [1] 4.301406
```

We see that these distances take similar values: in fact the last three upper-bound the first.

Inference using the Hilbert distance

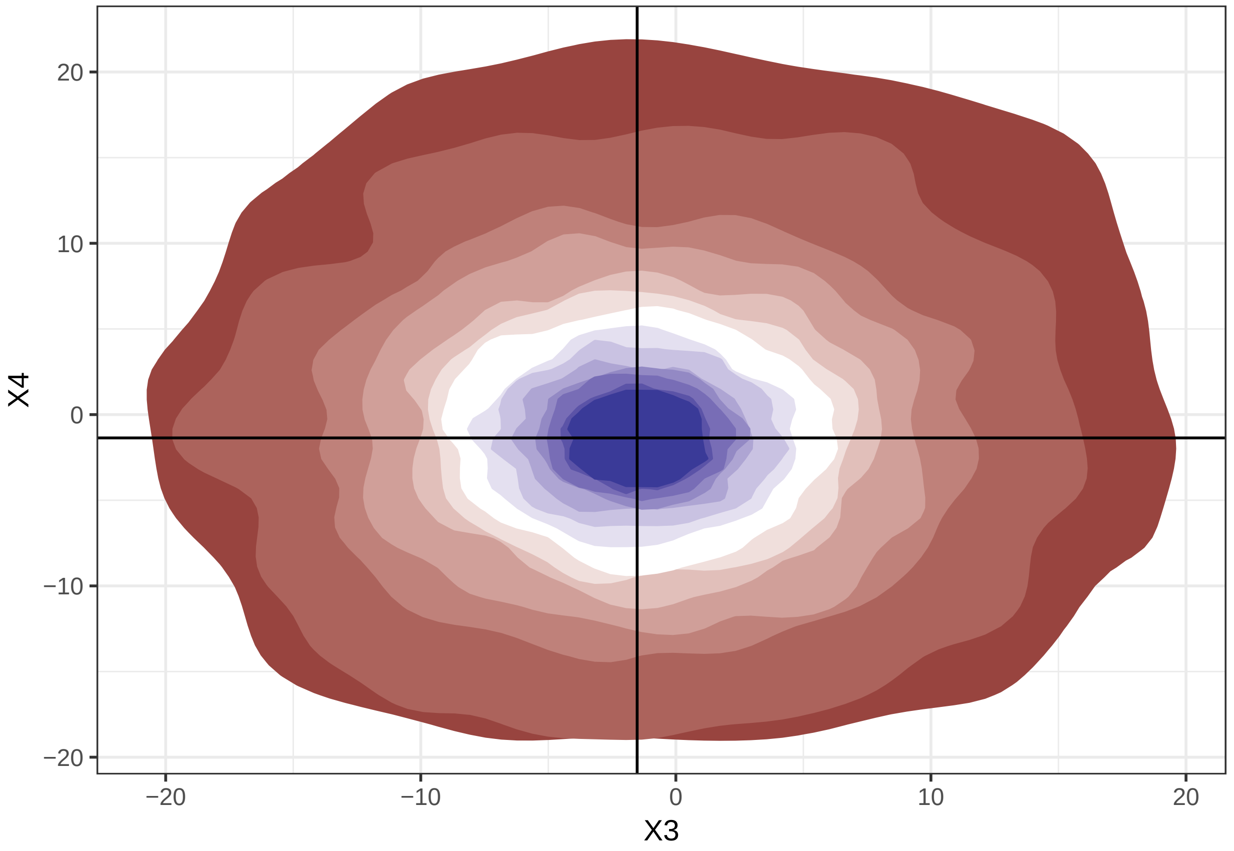
Now let's infer the parameters using the Hilbert distance. We specify the Hilbert distance to the *wsmc* function, which runs an SMC sampler with r-hit MCMC rejuvenation steps.

```
param_algo <- list(nthetas = 1024, nmoves = 1, proposal = mixture_rmixmod(),
    minimum_diversity = 0.5, R = 2, maxtrials = 1e+05)
compute_distance <- get_hilbert_to_y(obs)
wsmcresults_hilbert <- wsmc(compute_distance, target, param_algo, maxtime = 20)
```

```
## step 1 completed, in 0.781 seconds, 1024 distances calculated
## step 2... running until time >= 20 seconds
##   acceptance rates: 67.87109 %, threshold = 20.80382 , min. dist. = 4.934307
##   total # distances calculated: 4552 (for this step: 3528)
##   total time spent: 2.813 seconds (for this step: 2.032 seconds)
## step 3... running until time >= 20 seconds
##   acceptance rates: 60.35156 %, threshold = 16.01216 , min. dist. = 3.020232
##   total # distances calculated: 8215 (for this step: 3663)
##   total time spent: 4.627 seconds (for this step: 1.814 seconds)
## step 4... running until time >= 20 seconds
##   acceptance rates: 61.23047 %, threshold = 13.21339 , min. dist. = 3.025437
##   total # distances calculated: 11934 (for this step: 3719)
##   total time spent: 5.927 seconds (for this step: 1.3 seconds)
## step 5... running until time >= 20 seconds
##   acceptance rates: 58.88672 %, threshold = 11.12567 , min. dist. = 2.475331
##   total # distances calculated: 15596 (for this step: 3662)
##   total time spent: 7.418 seconds (for this step: 1.491 seconds)
## step 6... running until time >= 20 seconds
##   acceptance rates: 59.86328 %, threshold = 9.634652 , min. dist. = 2.450191
##   total # distances calculated: 19360 (for this step: 3764)
##   total time spent: 8.788 seconds (for this step: 1.37 seconds)
## step 7... running until time >= 20 seconds
##   acceptance rates: 55.56641 %, threshold = 8.209019 , min. dist. = 2.052517
##   total # distances calculated: 23079 (for this step: 3719)
##   total time spent: 10.083 seconds (for this step: 1.295 seconds)
## step 8... running until time >= 20 seconds
##   acceptance rates: 56.44531 %, threshold = 7.1128 , min. dist. = 2.570864
##   total # distances calculated: 26884 (for this step: 3805)
##   total time spent: 12.109 seconds (for this step: 2.026 seconds)
## step 9... running until time >= 20 seconds
##   acceptance rates: 53.125 %, threshold = 6.236227 , min. dist. = 2.238811
##   total # distances calculated: 30864 (for this step: 3980)
##   total time spent: 13.651 seconds (for this step: 1.542 seconds)
## step 10... running until time >= 20 seconds
##   acceptance rates: 54.98047 %, threshold = 5.570166 , min. dist. = 2.13767
##   total # distances calculated: 34745 (for this step: 3881)
##   total time spent: 15.062 seconds (for this step: 1.411 seconds)
## step 11... running until time >= 20 seconds
##   acceptance rates: 56.54297 %, threshold = 4.953446 , min. dist. = 1.732431
##   total # distances calculated: 38469 (for this step: 3724)
##   total time spent: 16.403 seconds (for this step: 1.341 seconds)
## step 12... running until time >= 20 seconds
##   acceptance rates: 51.85547 %, threshold = 4.339967 , min. dist. = 1.841749
##   total # distances calculated: 42331 (for this step: 3862)
##   total time spent: 17.828 seconds (for this step: 1.425 seconds)
## step 13... running until time >= 20 seconds
##   acceptance rates: 55.66406 %, threshold = 3.873351 , min. dist. = 1.818674
##   total # distances calculated: 46189 (for this step: 3858)
##   total time spent: 19.121 seconds (for this step: 1.293 seconds)
## step 14... running until time >= 20 seconds
##   acceptance rates: 54.6875 %, threshold = 3.473483 , min. dist. = 1.820002
##   total # distances calculated: 50089 (for this step: 3900)
##   total time spent: 20.553 seconds (for this step: 1.432 seconds)
```

The *wsmc* function outputs the above information to monitor the progress of the sampler, as the threshold decreases. We can plot the resulting approximation of the posterior distribution, for instance as follows, where the 2d-density plots obtained for the sequence of thresholds are overlaid. We focus on one of the bivariate marginals.

```
plot_bivariate_polygon(wsmcresults_hilbert, i1 = 3, i2 = 4) + geom_vline(xintercept = true_theta[3]) +
    geom_hline(yintercept = true_theta[4]) + xlab("X3") + ylab("X4")
```



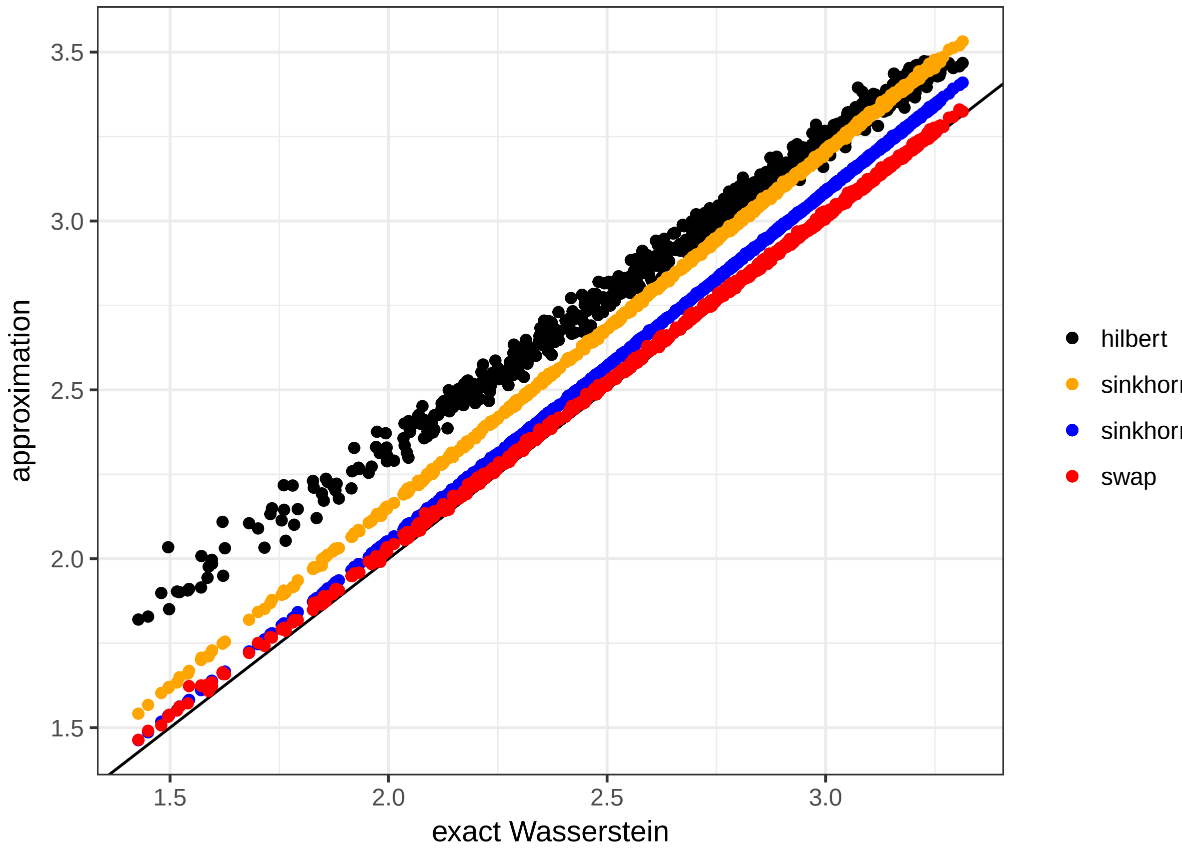
We see the WABC posterior distributions concentrating around the data-generating values, indicated by vertical and horizontal lines.

Distance comparison

Based on the resulting samples, we can compare the distances to the exact Wasserstein distances. To save calculations, we resort to the functions that compute the distances given a pre-computed cost matrix.

```
w1 <- rep(1/nobservations, nobservations)
w2 <- rep(1/nobservations, nobservations)
y_samples <- wsmcresults_hilbert$latest_y
d_comparison <- foreach(i = 1:length(y_samples), .combine = rbind) %dorng% {
    C <- cost_matrix_L2(obs, y_samples[[i]])
    hilbert <- hilbert_distance(obs, y_samples[[i]], p = 1, ground_p = 2)
    exact <- as.numeric(exact_transport_given_C(w1, w2, C, p = 1))
    sinkhorn1 <- sinkhorn_given_C(w1, w2, C, p = 1, eps = 0.05, niterations = 100)$corrected
    sinkhorn2 <- sinkhorn_given_C(w1, w2, C, p = 1, eps = 0.025, niterations = 1000)$corrected
    swap <- swap_distance(obs, y_samples[[i]], p = 1, ground_p = 2, tolerance = 1e-05)$distance
    data.frame(hilbert = hilbert, exact = exact, swap = swap, sinkhorn1 = sinkhorn1,
```

```r
        sinkhorn2 = sinkhorn2)
}
g <- qplot(x = d_comparison$exact, y = d_comparison$hilbert, geom = "blank")
g <- g + geom_point(aes(colour = "hilbert")) + geom_abline(slope = 1, intercept = 0)
g <- g + geom_point(aes(x = d_comparison$exact, y = d_comparison$sinkhorn1,
    colour = "sinkhorn 1"))
g <- g + geom_point(aes(x = d_comparison$exact, y = d_comparison$sinkhorn2,
    colour = "sinkhorn 2"))
g <- g + geom_point(aes(x = d_comparison$exact, y = d_comparison$swap, colour = "swap"))
g <- g + xlab("exact Wasserstein") + ylab("approximation") + scale_colour_manual(name = "",
    values = c("black", "orange", "blue", "red"))
g
```

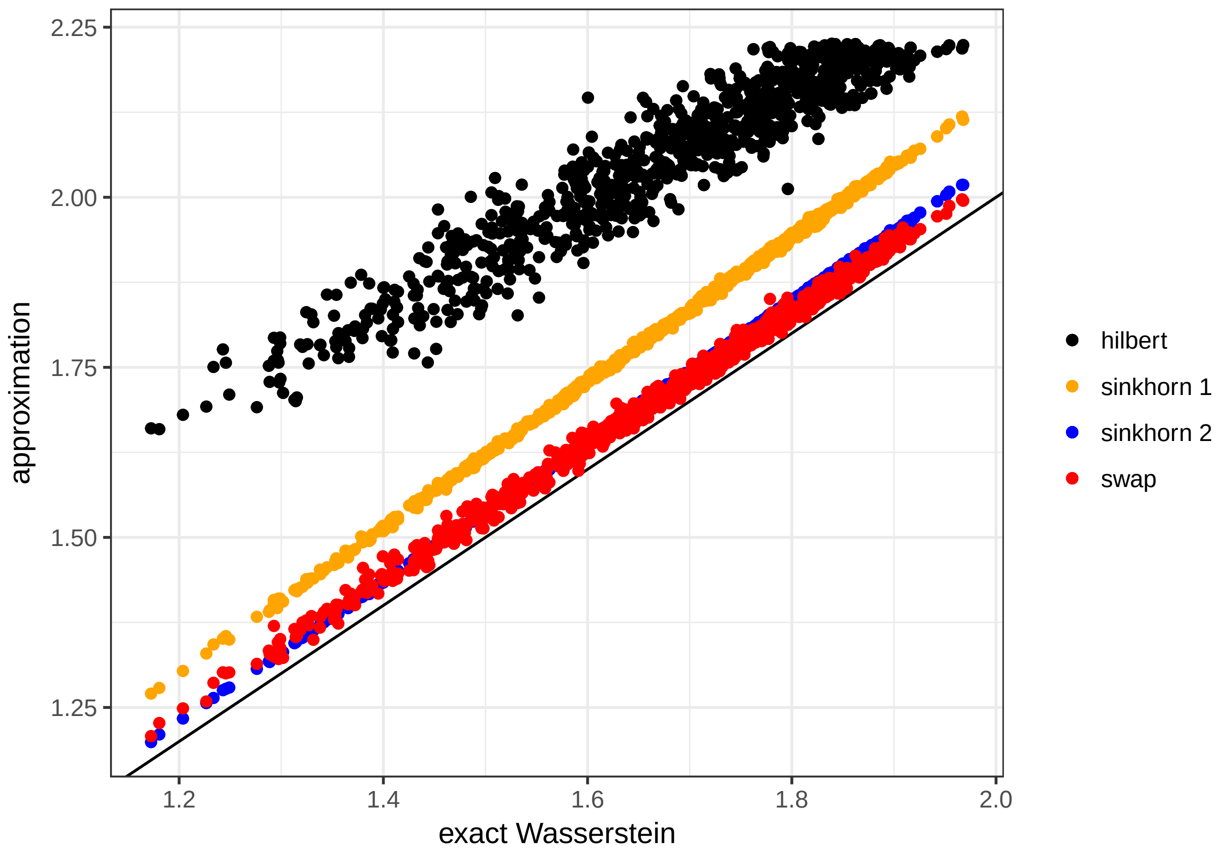


We see that the approximate distances (Hilbert, Sinkhorn, swapping) are all above the exact Wasserstein distances. Furthermore, they seem to be pretty close to the exact distances. However, if we pursue the inference by doing more SMC steps, the parameters concentrate, and then we see more difference between the distances.

```r
wsmcresults_hilbert_continued <- wsmc_continue(wsmcresults_hilbert, maxstep = 5)
```

```
## result file contains the result of 14 steps
## step 15... running for 5 more steps
##   acceptance rates: 53.41797 %, threshold = 3.145116 , min. dist. = 1.662534
##   total # distances calculated: 54026 (for this step: 3937)
##   total time spent: 22.108 seconds (since continue: 1.555 s; this step: 1.555 s)
## step 16... running for 5 more steps
##   acceptance rates: 56.83594 %, threshold = 2.849708 , min. dist. = 1.657344
##   total # distances calculated: 57884 (for this step: 3858)
##   total time spent: 23.516 seconds (since continue: 2.963 s; this step: 1.408 s)
## step 17... running for 5 more steps
##   acceptance rates: 54.10156 %, threshold = 2.584927 , min. dist. = 1.633157
##   total # distances calculated: 61825 (for this step: 3941)
##   total time spent: 25.021 seconds (since continue: 4.468 s; this step: 1.505 s)
## step 18... running for 5 more steps
##   acceptance rates: 53.71094 %, threshold = 2.38006 , min. dist. = 1.546865
##   total # distances calculated: 65917 (for this step: 4092)
##   total time spent: 26.513 seconds (since continue: 5.96 s; this step: 1.492 s)
## step 19... running for 5 more steps
##   acceptance rates: 53.02734 %, threshold = 2.226317 , min. dist. = 1.6591
##   total # distances calculated: 70020 (for this step: 4103)
##   total time spent: 28.206 seconds (since continue: 7.653 s; this step: 1.693 s)
```

```r
y_samples <- wsmcresults_hilbert_continued$latest_y
d_comparison <- foreach(i = 1:length(y_samples), .combine = rbind) %dorng% {
    C <- cost_matrix_L2(obs, y_samples[[i]])
    hilbert <- hilbert_distance(obs, y_samples[[i]], p = 1, ground_p = 2)
    exact <- as.numeric(exact_transport_given_C(w1, w2, C, p = 1))
    sinkhorn1 <- sinkhorn_given_C(w1, w2, C, p = 1, eps = 0.05, niterations = 100)$corrected
    sinkhorn2 <- sinkhorn_given_C(w1, w2, C, p = 1, eps = 0.025, niterations = 1000)$corrected
    swap <- swap_distance(obs, y_samples[[i]], p = 1, ground_p = 2, tolerance = 1e-05)$distance
    data.frame(hilbert = hilbert, exact = exact, swap = swap, sinkhorn1 = sinkhorn1,
        sinkhorn2 = sinkhorn2)
}
g <- qplot(x = d_comparison$exact, y = d_comparison$hilbert, geom = "blank")
g <- g + geom_point(aes(colour = "hilbert")) + geom_abline(slope = 1, intercept = 0)
g <- g + geom_point(aes(x = d_comparison$exact, y = d_comparison$sinkhorn1,
    colour = "sinkhorn 1"))
g <- g + geom_point(aes(x = d_comparison$exact, y = d_comparison$sinkhorn2,
    colour = "sinkhorn 2"))
g <- g + geom_point(aes(x = d_comparison$exact, y = d_comparison$swap, colour = "swap"))
g <- g + xlab("exact Wasserstein") + ylab("approximation") + scale_colour_manual(name = "",
    values = c("black", "orange", "blue", "red"))
g
```



We see that the difference between the Hilbert distance and the exact Wasserstein distances widens. However, the Sinkhorn distance with a small value of ϵ, and the swap distance, are still very close to the Wasserstein distances.