Tutorial: distance calculations

PJ, EB, MG, CR

March 21, 2017

Setting

This script compares different distances between datasets, 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.

```
# load package
library(winference)
# register parallel cores
registerDoMC(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.

```
# 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 x nobservations matrix
```

Distances

We consider three distances between multivariate samples: the exact Wasserstein distance, using to the transport package, the Sinkhorn distance, due to Marco Cuturi, and the Hilbert-based distance proposed in our article. Let's test these distances on two datasets.

```r
# generate a fake data set
fake_obs <- target$simulate(rnorm(dimension))
w1 <- rep(1/nobservations, nobservations)
w2 <- rep(1/nobservations, nobservations)
C <- cost_matrix_L2(obs, fake_obs)
exact_transport_given_C(w1, w2, C, p = 1)
```

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

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

```r
sinkhorn_given_C(w1, w2, C, p = 1, eps = 0.1, niterations = 100)
```

```
## $uncorrected
## [1] 4.563124
##
## $corrected
## [1] 4.563124
```

```r
sinkhorn_given_C(w1, w2, C, p = 1, eps = 0.01, niterations = 100)
```

```
## $uncorrected
## [1] 4.185088
##
## $corrected
## [1] 4.185117
```

```r
sinkhorn_given_C(w1, w2, C, p = 1, eps = 0.01, niterations = 1000)
```

```
## $uncorrected
## [1] 4.18509
##
## $corrected
## [1] 4.18509
```

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

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

Inference using the Hilbert distance

Now let's infer the parameters using the Hilbert distance.

```r
param_algo <- list(nthetas = 1024, nmoves = 1, proposal = mixture_rmixmod(),
    minimum_diversity = 0.5, R = 2, maxtrials = 1e+05)
compute_distance <- function(y_fake) {
    return(hilbert_distance(obs, y_fake, p = 1, ground_p = 2))
}
wsmcresults_hilbert <- wsmc(compute_distance, target, param_algo, maxtime = 20)
```

```
## step 1 completed, in 0.613 seconds, 1024 distances calculated
## step 2... running until time >= 20 seconds
##   acceptance rates: 67.28516 %, threshold = 20.80382 , min. dist. = 4.174539
##   total # distances calculated: 4530 (for this step: 3506)
##   total time spent: 1.942 seconds (for this step: 1.329 seconds)
## step 3... running until time >= 20 seconds
##   acceptance rates: 62.5 %, threshold = 16.14772 , min. dist. = 3.3721
```

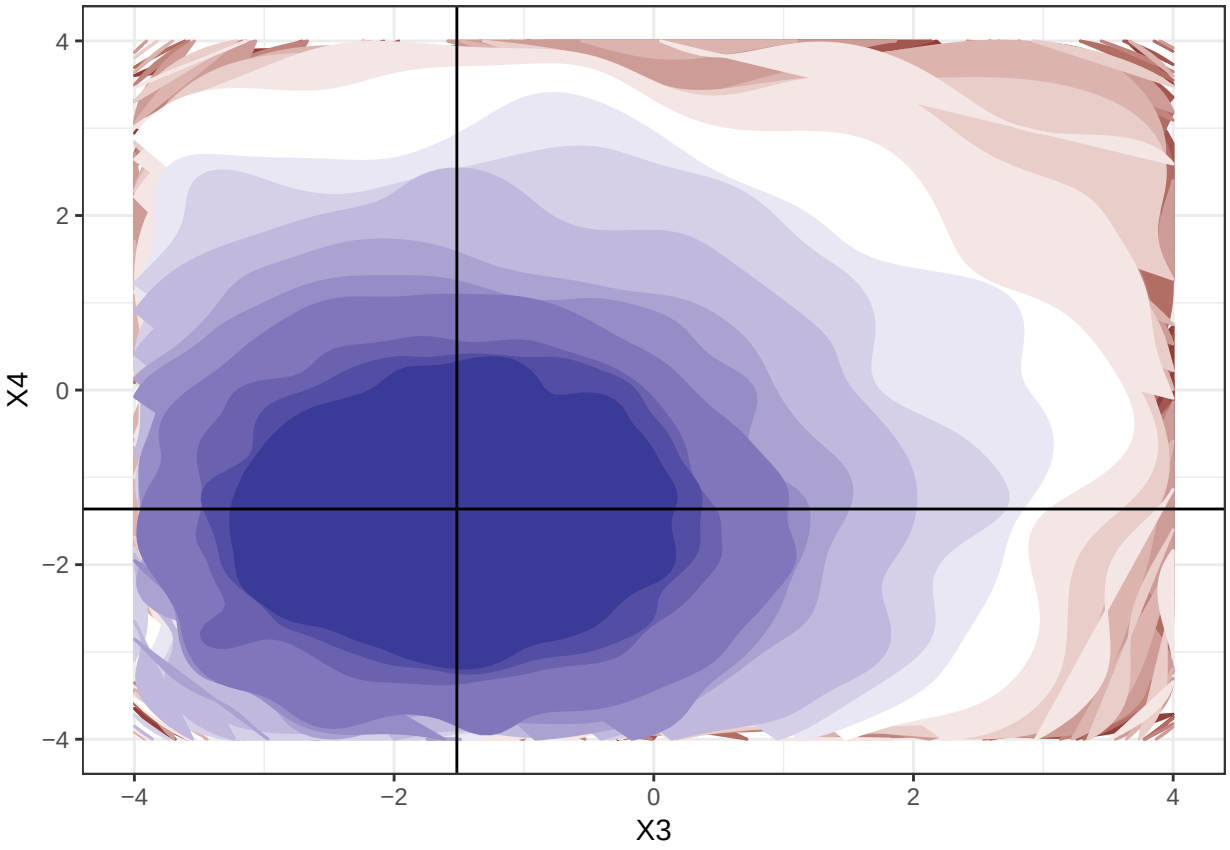
```
##   total # distances calculated: 8152 (for this step: 3622)
##   total time spent: 2.904 seconds (for this step: 0.962 seconds)
## step 4... running until time >= 20 seconds
##   acceptance rates: 59.86328 %, threshold = 13.44046 , min. dist. = 3.3721
##   total # distances calculated: 11863 (for this step: 3711)
##   total time spent: 4.05 seconds (for this step: 1.146 seconds)
## step 5... running until time >= 20 seconds
##   acceptance rates: 62.98828 %, threshold = 11.39733 , min. dist. = 3.439999
##   total # distances calculated: 15567 (for this step: 3704)
##   total time spent: 5.057 seconds (for this step: 1.007 seconds)
## step 6... running until time >= 20 seconds
##   acceptance rates: 56.05469 %, threshold = 9.573353 , min. dist. = 2.106427
##   total # distances calculated: 19316 (for this step: 3749)
##   total time spent: 6.225 seconds (for this step: 1.168 seconds)
## step 7... running until time >= 20 seconds
##   acceptance rates: 54.98047 %, threshold = 8.274334 , min. dist. = 2.077194
##   total # distances calculated: 23111 (for this step: 3795)
##   total time spent: 7.363 seconds (for this step: 1.138 seconds)
## step 8... running until time >= 20 seconds
##   acceptance rates: 58.10547 %, threshold = 7.292736 , min. dist. = 2.416618
##   total # distances calculated: 26896 (for this step: 3785)
##   total time spent: 8.838 seconds (for this step: 1.475 seconds)
## step 9... running until time >= 20 seconds
##   acceptance rates: 56.54297 %, threshold = 6.341073 , min. dist. = 1.981011
##   total # distances calculated: 30780 (for this step: 3884)
##   total time spent: 9.913 seconds (for this step: 1.075 seconds)
## step 10... running until time >= 20 seconds
##   acceptance rates: 55.66406 %, threshold = 5.531723 , min. dist. = 2.024804
##   total # distances calculated: 34622 (for this step: 3842)
##   total time spent: 10.94 seconds (for this step: 1.027 seconds)
## step 11... running until time >= 20 seconds
##   acceptance rates: 54.39453 %, threshold = 4.882592 , min. dist. = 2.099688
##   total # distances calculated: 38394 (for this step: 3772)
##   total time spent: 11.953 seconds (for this step: 1.013 seconds)
## step 12... running until time >= 20 seconds
##   acceptance rates: 54.39453 %, threshold = 4.291613 , min. dist. = 1.7629
##   total # distances calculated: 42252 (for this step: 3858)
##   total time spent: 12.981 seconds (for this step: 1.028 seconds)
## step 13... running until time >= 20 seconds
##   acceptance rates: 54.78516 %, threshold = 3.782556 , min. dist. = 1.771962
##   total # distances calculated: 46040 (for this step: 3788)
##   total time spent: 14.024 seconds (for this step: 1.043 seconds)
## step 14... running until time >= 20 seconds
##   acceptance rates: 57.51953 %, threshold = 3.441533 , min. dist. = 1.802207
##   total # distances calculated: 49904 (for this step: 3864)
##   total time spent: 15.152 seconds (for this step: 1.128 seconds)
## step 15... running until time >= 20 seconds
##   acceptance rates: 54.49219 %, threshold = 3.099701 , min. dist. = 1.681556
##   total # distances calculated: 53698 (for this step: 3794)
##   total time spent: 16.607 seconds (for this step: 1.455 seconds)
## step 16... running until time >= 20 seconds
##   acceptance rates: 53.41797 %, threshold = 2.824453 , min. dist. = 1.670222
##   total # distances calculated: 57575 (for this step: 3877)
##   total time spent: 17.666 seconds (for this step: 1.059 seconds)
```

```
## step 17... running until time >= 20 seconds
##   acceptance rates: 52.92969 %, threshold = 2.59984 , min. dist. = 1.645677
##   total # distances calculated: 61508 (for this step: 3933)
##   total time spent: 19.056 seconds (for this step: 1.39 seconds)
## step 18... running until time >= 20 seconds
##   acceptance rates: 56.25 %, threshold = 2.404827 , min. dist. = 1.622167
##   total # distances calculated: 65583 (for this step: 4075)
##   total time spent: 20.045 seconds (for this step: 0.989 seconds)
```

We can plot the resulting distribution, for instance as follows.

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

```
## Warning: Removed 6158 rows containing non-finite values (stat_density2d).
```

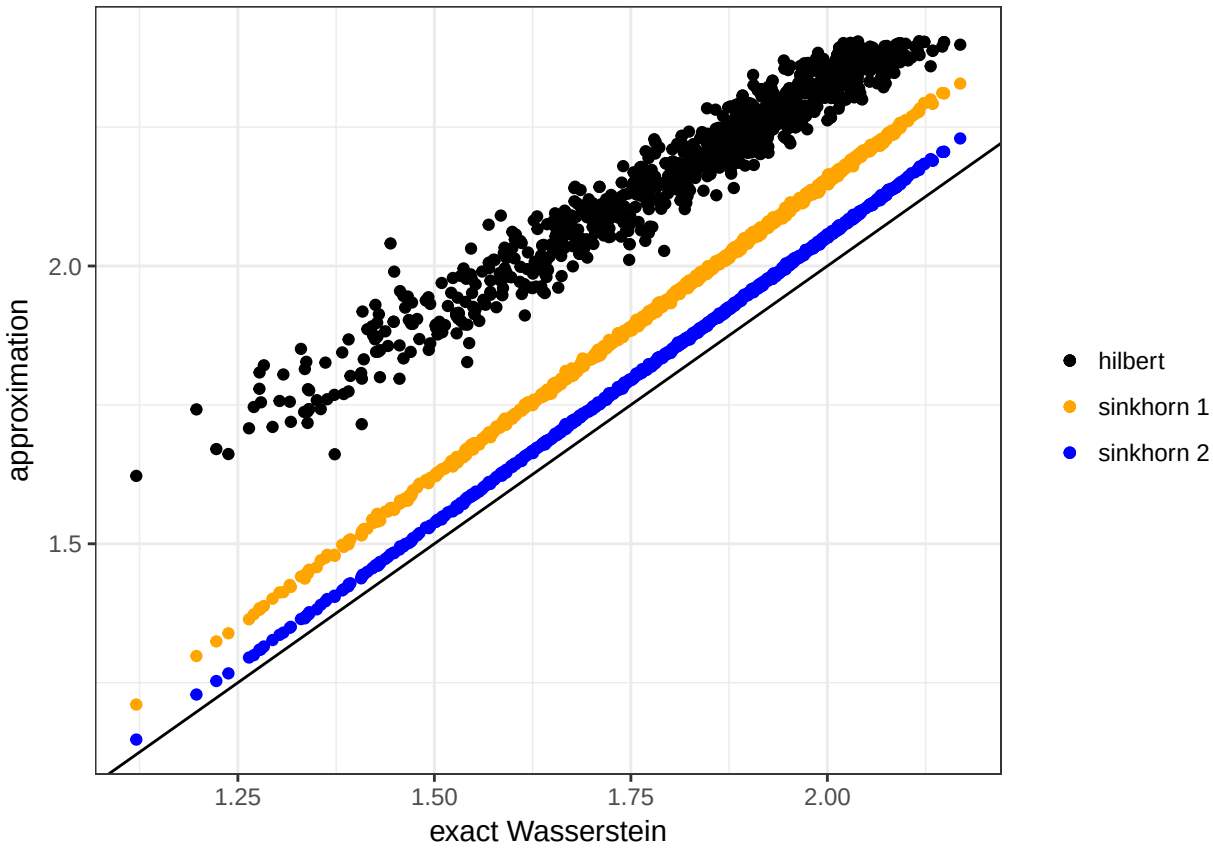


Distance comparison

Based on the resulting samples, we can compare the Hilbert distances to the distances that we would have obtained with the exact Wasserstein and Sinkhorn distances.

```
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
    data.frame(hilbert = hilbert, exact = exact, 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 + xlab("exact Wasserstein") + ylab("approximation") + scale_colour_manual(name = "",
    values = c("black", "orange", "blue"))
g
```



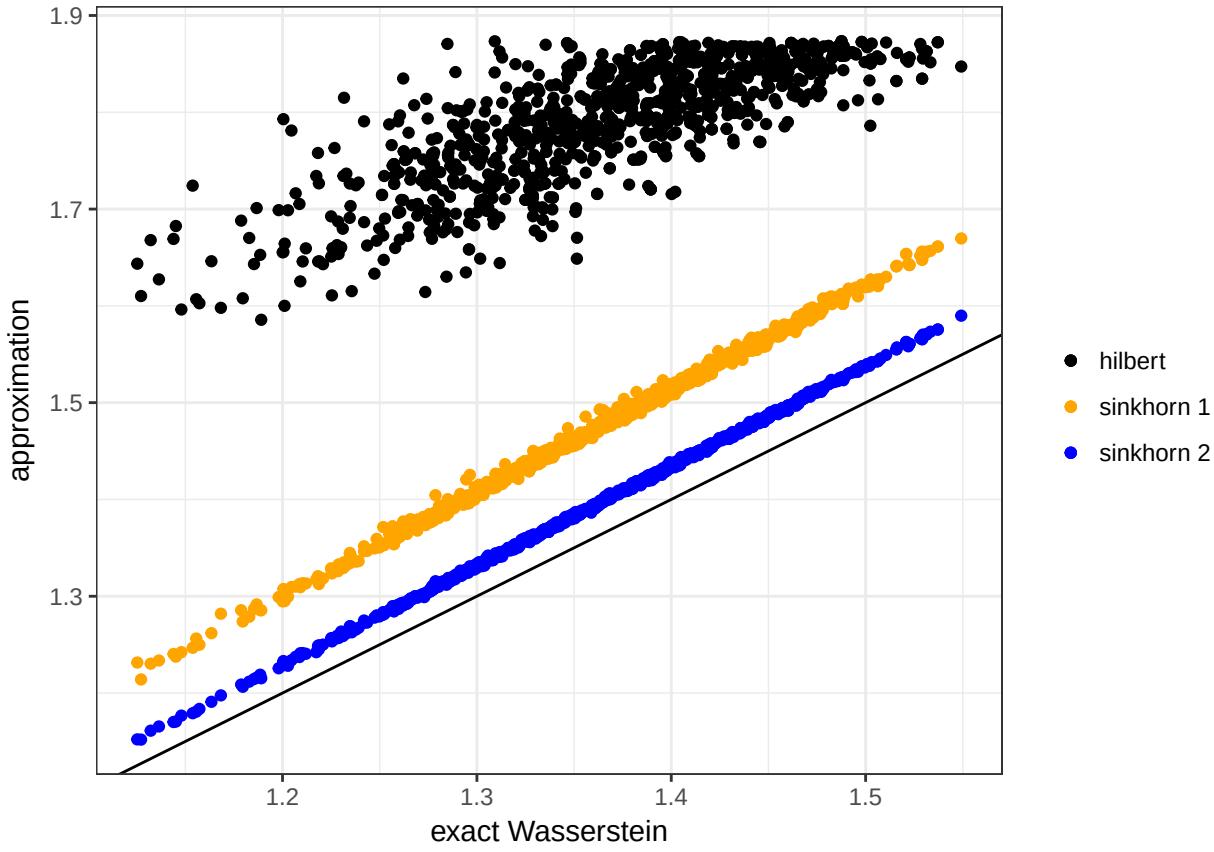
We see that the ordering of the distances align for all methods. However, if we pursue the inference by doing more SMC steps, the parameters concentrate, and then we see more difference between the different distances.

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

```
## result file contains the result of 18 steps
## step 19... running for 5 more steps
##   acceptance rates: 52.34375 %, threshold = 2.239387 , min. dist. = 1.661535
##   total # distances calculated: 69581 (for this step: 3998)
##   total time spent: 21.181 seconds (since continue: 1.136 s; this step: 1.136 s)
## step 20... running for 5 more steps
##   acceptance rates: 51.75781 %, threshold = 2.111975 , min. dist. = 1.60748
```

```
##   total # distances calculated: 73768 (for this step: 4187)
##   total time spent: 22.269 seconds (since continue: 2.224 s; this step: 1.088 s)
## step 21... running for 5 more steps
##   acceptance rates: 50 %, threshold = 2.019765 , min. dist. = 1.599022
##   total # distances calculated: 78094 (for this step: 4326)
##   total time spent: 23.431 seconds (since continue: 3.386 s; this step: 1.162 s)
## step 22... running for 5 more steps
##   acceptance rates: 51.17188 %, threshold = 1.942189 , min. dist. = 1.563693
##   total # distances calculated: 82431 (for this step: 4337)
##   total time spent: 24.517 seconds (since continue: 4.472 s; this step: 1.086 s)
## step 23... running for 5 more steps
##   acceptance rates: 48.24219 %, threshold = 1.873651 , min. dist. = 1.585618
##   total # distances calculated: 87084 (for this step: 4653)
##   total time spent: 25.886 seconds (since continue: 5.841 s; this step: 1.369 s)
```

```
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)$corrected
    data.frame(hilbert = hilbert, exact = exact, sinkhorn1 = sinkhorn1, sinkhorn2 = sinkhorn2)
}
g <- qplot(x = d_comparison$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 + xlab("exact Wasserstein") + ylab("approximation") + scale_colour_manual(name = "",
    values = c("black", "orange", "blue"))
g
```


1.9
1.7
1.5
1.3
approximation
1.2
1.3
1.4
1.5
exact Wasserstein
hilbert
sinkhorn 1
sinkhorn 2