Check Scaling Variance

David Gerard

2016-04-27

Abstract

This is a check on the EM algorithm for the scaling variance.

Check Scale Invariance

Data generating mechanism

```
library(succotashr)
set.seed(94)
p <- 100
k <- 5

pi_vals <- c(0.5, 0.5)
tau_seq <- c(0, 5)
beta <- draw_beta(pi_vals = pi_vals, tau_seq = tau_seq, p = p)
Z <- matrix(rnorm(k), ncol = 1)
alpha <- matrix(rnorm(p * k), nrow = p)
sig_diag <- abs(rnorm(p))
```

The estimate of λ when $\hat{\Sigma}$ is half as big as the true Σ should be twice that of the estimate of λ when the given $\hat{\Sigma}$ is equal to the true Σ.

```
scale_val <- 2 ## true sigma is two times known sigma
E <- rnorm(p) * sqrt(scale_val * sig_diag)
Y <- beta + alpha %*% Z + E

succ_out1 <- succotash_given_alpha(Y = Y, alpha = alpha, sig_diag = sig_diag,
                                   tau_seq = tau_seq, var_scale = TRUE,
                                   lambda_type = "ones")

succ_out2 <- succotash_given_alpha(Y = Y, alpha = alpha, sig_diag = sig_diag * scale_val,
                                   tau_seq = tau_seq, var_scale = TRUE,
                                   lambda_type = "ones")

succ_out1$scale_val / 2
```

```
## [1] 0.964
```

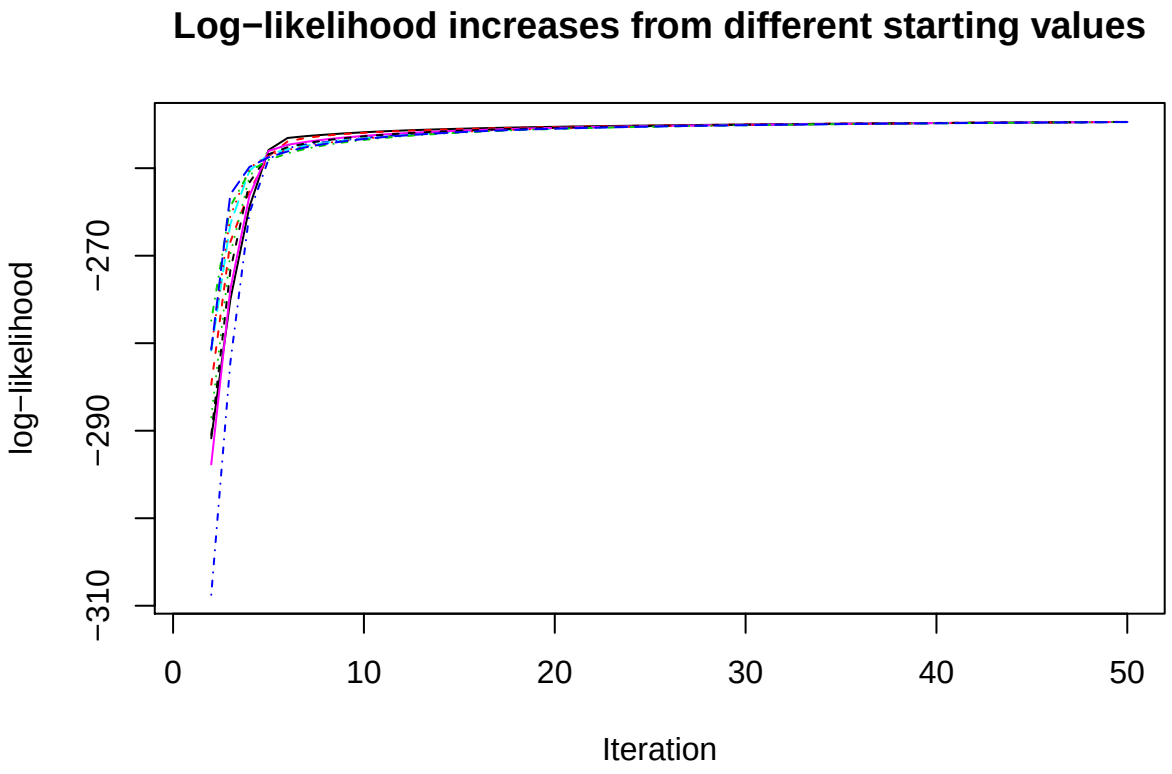
```
succ_out2$scale_val
```

```
## [1] 0.964
```

Check that log-likelihood is increasing.

```
itermax <- 50
llike_vec <- rep(NA, length = itermax)

## default grid to be same as in ASH
tau_min <- min(sig_diag) / 10
tau_max <- 2 * sqrt(max(Y ^ 2 - sig_diag))
if (tau_max < 0) {
    tau_max <- 8 * tau_min
}
tau_current <- tau_min
tau_seq_emp <- c(0, tau_current)
mult_fact <- sqrt(2)
while (tau_current <= tau_max) {
    tau_current <- tau_current * mult_fact
    tau_seq_emp <- c(tau_seq_emp, tau_current)
}
M <- length(tau_seq_emp)
## most mass at 0
pi_init <- rep(NA, length = M)
pi_init[2:M] <- min(1 / p, 1 / M)
pi_init[1] <- 1 - sum(pi_init[2:M])

lambda <- rep(NA, length = M)
lambda[1] <- 10
lambda[2:M] <- 1

scale_vec <- seq(0.5, 2, length = 10)

llike_mat <- matrix(NA, nrow = itermax, ncol = length(scale_vec))
for(scale_index in 1:length(scale_vec)) {
    scale_init <- scale_vec[scale_index]
    Z_init <- rnorm(k)

    pi_Z_new <- c(pi_init, Z_init, scale_init)


    llike_old <-
        succotashr:::succotash_llike(pi_Z = pi_Z_new, lambda = lambda,
                                     alpha = alpha, Y = Y,
                                     tau_seq = tau_seq_emp, sig_diag = sig_diag,
                                     plot_new_ests = FALSE, var_scale = TRUE)
    llike_vec[1] <- llike_old
    for (index in 2:itermax) {
        pi_Z_new <-
            succotashr:::succotash_fixed(pi_Z = pi_Z_new, lambda = lambda, alpha = alpha,
                                         Y = Y,
                                         tau_seq = tau_seq_emp, sig_diag = sig_diag,
                                         plot_new_ests = FALSE, var_scale = TRUE)
        llike_new <-
            succotashr:::succotash_llike(pi_Z = pi_Z_new, lambda = lambda, alpha = alpha,
                                         Y = Y,
```

```
                                        tau_seq = tau_seq_emp, sig_diag = sig_diag,
                                        plot_new_ests = FALSE, var_scale = TRUE)
        llike_mat[index, scale_index] <- llike_new
    }
}
matplot(llike_mat, type = "l", ylab = "log-likelihood", xlab = "Iteration",
        main = "Log-likelihood increases from different starting values")
```

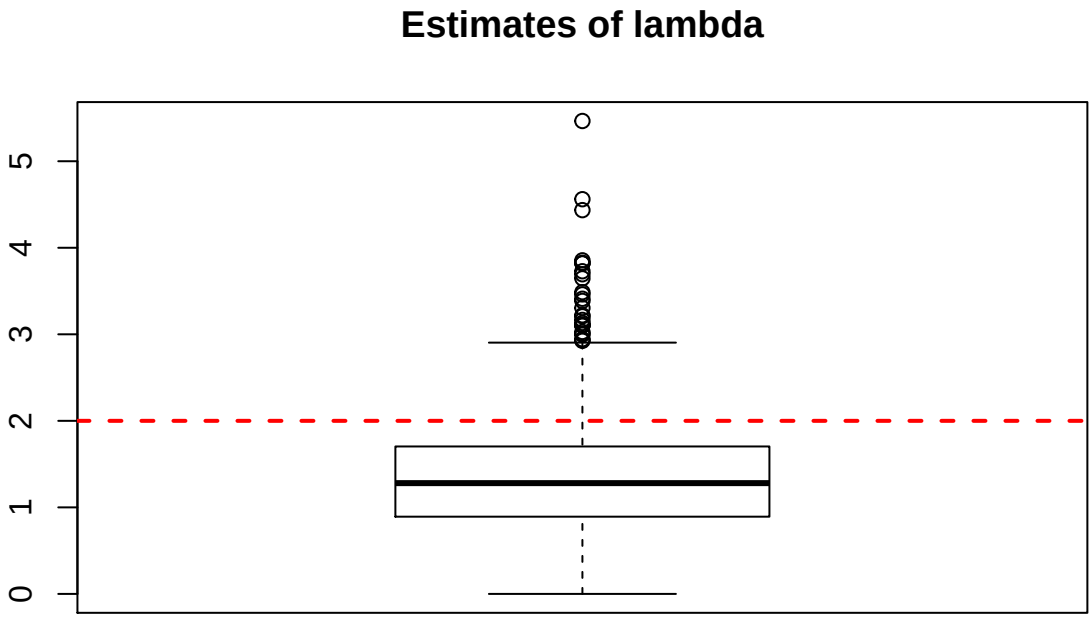


Run simulations

I give SUCCOTASH the true grid, the true α, and the true Σ divided by 2. I don't regularize the mixing proportions. It does pretty well at estimating λ to be near 2 and π_0 to be near 0.5.

```
itermax <- 1000
scale_vec <- rep(NA, length = itermax)
pi0_vec <- rep(NA, length = itermax)
for(iter_index in 1:itermax) {
    scale_val <- 2 ## true sigma is two times known sigma
    E <- rnorm(p) * sqrt(scale_val * sig_diag)
    Y <- beta + alpha %*% Z + E

    succ_out <- succotash_given_alpha(Y = Y, alpha = alpha, sig_diag = sig_diag,
                                      tau_seq = tau_seq, var_scale = TRUE,
                                      lambda_type = "ones")
    pi0_vec[iter_index] <- succ_out$pi0
    scale_vec[iter_index] <- succ_out$scale_val
}
```

```
boxplot(scale_vec, main = "Estimates of lambda")
abline(h = scale_val, col = 2, lwd = 2, lty = 2)
```




```
boxplot(pi0_vec, main = "Estimates of pi0")
abline(h = pi_vals[1], col = 2, lwd = 2, lty = 2)
```