Reproducible analysis via SMCEM for multivariate probit models

Supplement to Sequential Monte Carlo EM for multivariate probit models

Giusi Moffa and Jack Kuipers

Software

This document is compiled with `knitr_1.2` under R version `3.0.0`. The complete analysis is also run under R version `3.0.0`, with `mvtnorm` package for multivariate normal and t distributions.

1 Six cities data analysis

Loading libraries, sources and data

```
rm(list = ls(all.names = TRUE))
require(mvtnorm)
require(xtable)
pkgfolder <- "../smcPbitDemo/"
datafile <- paste0(pkgfolder, "sixcities.RData")
### file with data to model
tosave <- TRUE
savefile <- "smcSixCdate.RData"
### file to save R workspace after processing From within the working
### directory
source(paste0(pkgfolder, "resample.R"))
source(paste0(pkgfolder, "siginit.R"))
source(paste0(pkgfolder, "obseval.R"))
source(paste0(pkgfolder, "PbitEMstart.R"))
source(paste0(pkgfolder, "PbitEMloop.R"))
source(paste0(pkgfolder, "Mstep.R"))
source(paste0(pkgfolder, "smcEM.R"))
source(paste0(pkgfolder, "evallike.R"))
source(paste0(pkgfolder, "fisher.R"))
load(datafile)  ### Read data
# source(paste0(pkgfolder,'buildsixcities.R'))
dep_prev()
```

Initializations

```
doProjection <- FALSE
# projection to sigma_11=1 form is performed after M step - projection is
# not needed if constrained is TRUE
constrained <- TRUE
```

```
# Maximisation is performed constrained - either in correlation form or
# with sigma_11=1 depending on useinvariance
useinvariance <- FALSE
# Maximisation uses invariant Q tilde function instead of standard Q
fixM <- FALSE
# keep the number of particles constant or not
nr <- 40
M <- 2000
refineM <- 2 * M
startM <- 100
nc <- length(citygam[, 1])
nobs <- length(citygam[1, ])
nX <- length(cityX[[1]][1, ]) - 1
avg.gam <- apply(citygam, 1, mean)  ### mean vector
cor.gam <- cor(t(citygam))  ### correlation matrix
set.seed(101)
```

Compute starting point for the EM algorithm

```
pwsig <- siginit(avg.gam, cor.gam)  ### covariance matrix pairwise estimate
updsig <- pwsig
obs <- c(t(citygam))
covmat <- matrix(0, nrow = nc * nobs, ncol = nX)
covmataux <- matrix(0, nrow = nobs, ncol = nX)
for (i in 1:nc) {
    for (j in 1:nobs) covmataux[j, ] <- cityX[[j]][i, 2:(nX + 1)]
    covmat[((i - 1) * nobs + 1):(i * nobs), ] <- covmataux
}
indprobit <- glm(obs ~ covmat, family = binomial(link = "probit"), na.action = na.pass)
indbe <- coef(indprobit)  ### coefficients from fitting independent probit models
updbe <- indbe
```

Initialise variables for the results

```
loglike <- rep(NA, nr)
regcoeff <- list()
regcoeff[[1]] <- updbe
covest <- list()
covest[[1]] <- updsig
```

First step of the SMCEM algorithm

```
res <- PbitEMstart(citygam, cityX, nc, nX, updbe, updsig, M)
loglike[1] <- res$loglike
### record sigma and beta at previous step for sequential sampling
prevBe <- updbe
prevSig <- updsig
### update sigma and beta with new estiamte
```

```
updbe <- res$estbe
updsig <- res$sigem
covest[[2]] <- updsig
regcoeff[[2]] <- updbe
```

Maximum likelihood estimation via SMCEM, loop

```
### parameters for kernel adaptation, sf defined in smcEM
log.fac <- 0  ### work on log to ensure scaling factor stays positive
alpha.wanted <- 0.3  ### target acceptance probability for control of adaptive MH
sf.ga <- 7  ### Stepsize to adapt the scaling factor
mh <- 4  ### number of mcmc steps within smc
```

```
### sequential monte carlo EM ###
for (k in 2:nr) {
    res <- PbitEMloop(citygam, cityX, nc, nX, updbe, prevBe, updsig, prevSig,
        M)
    loglike[k] <- res$loglike
    ### record sigma and beta at previous step for sequential sampling
    prevBe <- updbe
    prevSig <- updsig
    updbe <- res$estbe  ### update beta
    updsig <- res$sigem  ### update sigma
    covest[[k + 1]] <- updsig
    regcoeff[[k + 1]] <- updbe
}
sixCres <- list(loglike = loglike, estbe = updbe, estsig = updsig, regcoeff = regcoeff,
    covest = covest)
```

```
if (tosave) save.image(file = savefile)
```

Maximum likelihood estimates

[1] "Obtained with 2000 particles, and over 40 iterations"

	beta0	beta1	beta2	beta3
Beta	-1.122	-0.078	0.157	0.038

Table 1: Maximum likelihood estimates of the regression coefficients

	s1	s2	s3	s4
s1	1	0.581	0.523	0.578
s2		1	0.682	0.558
s3			1	0.624
s4				1

Table 2: Maximum likelihood estimates of the correlation coefficients, matrix form

Evaluate log-likelihood

```
peLogLike <- matrix(evallike(citygam, cityX, nc, updbe, updsig, refineM))
```

	x
log-likelihood	-795.027

Table 3: Estimated log-likelihood value

Evaluate standard errors

```
best <- matrix(0, nrow = nc, ncol = nobs)
for (i in 1:nobs) best[, i] <- cityX[[i]] %*% updbe

if (!constrained || useinvariance) dimInf <- length(updbe) + nc * (nc + 1)/2 -
    1 else dimInf <- length(updbe) + nc * (nc - 1)/2
totinfo <- matrix(0, nrow = dimInf, ncol = dimInf)
totHess <- matrix(0, nrow = dimInf, ncol = dimInf)
totestquad <- matrix(0, nrow = dimInf, ncol = dimInf)
totGradSq <- matrix(0, nrow = dimInf, ncol = dimInf)
totquval <- matrix(0, nrow = dimInf, ncol = dimInf)

for (j in 1:nobs) {
    samp <- obseval(citygam[, j], best[, j], updsig, refineM)
    obsInfo <- fisher(updbe, updsig, cityX[[j]], samp$samp, samp$W)
    totquval <- totquval + obsInfo$quval  ### total second moment expectation
    totGradSq <- totGradSq + obsInfo$gradsq  ### total expectation of gradient quadratic form
    totestquad <- totestquad + obsInfo$quad  ### total variance estimate
    totHess <- totHess + obsInfo$Hess  ### total hessian estimate
    totinfo <- totinfo + obsInfo$info  ###
}

myse <- sqrt(diag(solve(totinfo)))
```

	par1	par2	par3	par4	par5	par6	par7	par8	par9	par10
se	0.062	0.031	0.101	0.051	0.067	0.071	0.074	0.057	0.075	0.069

Table 4: Standard errors of the estimated parameters

	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
1	0.032	0.058	0.068	0.066	0.074	0.101

Table 5: Summary of the distribution of the standard errors of the estimated parmaters

2 Simulated data analysis

Loading libraries, sources and data

```
rm(list = ls(all.names = TRUE))
require(mvtnorm)
require(xtable)
pkgfolder <- "../smcPbitDemo/"
datafile <- paste0(pkgfolder, "simPbit8Data1k.RData")  ### file with data to model
### file with data to model
tosave <- TRUE
savefile <- "smcemPbitDemo.RData"  ### file to save R workspace after processing
### file to save R workspace after processing From within the working
### directory
source(paste0(pkgfolder, "resample.R"))
source(paste0(pkgfolder, "siginit.R"))
source(paste0(pkgfolder, "obseval.R"))
source(paste0(pkgfolder, "PbitEMstart.R"))
source(paste0(pkgfolder, "PbitEMloop.R"))
source(paste0(pkgfolder, "Mstep.R"))
source(paste0(pkgfolder, "smcEM.R"))
source(paste0(pkgfolder, "evallike.R"))
source(paste0(pkgfolder, "fisher.R"))
load(datafile)  ### Read data
# source(paste0(pkgfolder,'buildsixcities.R'))
dep_prev()
```

Initializations

```
doProjection <- FALSE
# projection to sigma_11=1 form is performed after M step - projection is
# not needed if constrained is TRUE
constrained <- TRUE
# Maximisation is performed constrained - either in correlation form or
# with sigma_11=1 depending on useinvariance
useinvariance <- TRUE
# Maximisation uses invariant Q tilde function instead of standard Q
fixM <- FALSE
# keep the number of particles constant or not
nr <- 40
M <- 4000
refineM <- 1 * M
startM <- 100
nc <- length(citygam[, 1])
nobs <- length(citygam[1, ])
nX <- length(cityX[[1]][1, ]) - 1
avg.gam <- apply(citygam, 1, mean)  ### mean vector
cor.gam <- cor(t(citygam))  ### correlation matrix
```

Compute starting point for the EM algorithm

```
pwsig <- siginit(avg.gam, cor.gam)  ### covariance matrix pairwise estimate
updsig <- pwsig
obs <- c(t(citygam))
covmat <- matrix(0, nrow = nc * nobs, ncol = nX)
covmataux <- matrix(0, nrow = nobs, ncol = nX)
for (i in 1:nc) {
    for (j in 1:nobs) covmataux[j, ] <- cityX[[j]][i, 2:(nX + 1)]
    covmat[((i - 1) * nobs + 1):(i * nobs), ] <- covmataux
}
indprobit <- glm(obs ~ covmat, family = binomial(link = "probit"), na.action = na.pass)
indbe <- coef(indprobit)  ### coefficients from fitting independent probit models
updbe <- indbe
```

Initialise variables for the results

```
loglike <- rep(NA, nr)
regcoeff <- list()
regcoeff[[1]] <- updbe
covest <- list()
covest[[1]] <- updsig
```

First step of the SMCEM algorithm

```
res <- PbitEMstart(citygam, cityX, nc, nX, updbe, updsig, M)
loglike[1] <- res$loglike
### record sigma and beta at previous step for sequential sampling
prevBe <- updbe
prevSig <- updsig
### update sigma and beta with new estiamte
updbe <- res$estbe
updsig <- res$sigem
covest[[2]] <- updsig
regcoeff[[2]] <- updbe
```

Maximum likelihood estimation via SMCEM, loop

```
### parameters for kernel adaptation, sf defined in smcEM
log.fac <- 0  ### work on log to ensure scaling factor stays positive
alpha.wanted <- 0.3  ### target acceptance probability for control of adaptive MH
sf.ga <- 7  ### Stepsize to adapt the scaling factor
mh <- 4  ### number of mcmc steps within smc
```

```
### sequential monte carlo EM ###
for (k in 2:nr) {
    res <- PbitEMloop(citygam, cityX, nc, nX, updbe, prevBe, updsig, prevSig,
        M)
    loglike[k] <- res$loglike
```

```
    ### record sigma and beta at previous step for sequential sampling
    prevBe <- updbe
    prevSig <- updsig
    updbe <- res$estbe  ### update beta
    updsig <- res$sigem  ### update sigma
    covest[[k + 1]] <- updsig
    regcoeff[[k + 1]] <- updbe
}
simPbitRes <- list(loglike = loglike, estbe = updbe, estsig = updsig, regcoeff = regcoeff,
    covest = covest)
```

```
if (tosave) save.image(file = savefile)
```

Maximum likelihood estimates

[1] "Obtained with 4000 particles, and over 40 iterations"

	beta0	beta1	beta2	beta3	beta4	beta5	beta6
Beta	0.998	0.421	-0.285	0.264	-0.186	0.089	-0.061
Real beta	1.000	0.300	-0.300	0.200	-0.200	0.100	-0.100

Table 6: Maximum likelihood estimates of the regression coefficients and real values from which the data are simulated

	s1	s2	s3	s4	s5	s6	s7	s8
s1	1	0.017	0.011	0.248	0.11	0.176	0.213	0.391
s2		1.126	0.071	0.195	0.099	0.198	0.361	0.323
s3			1.424	0.055	0.122	0.142	0.249	0.363
s4				1.113	0.234	0.171	0.219	0.509
s5					1.139	0.232	0.241	0.409
s6						0.841	0.39	0.53
s7							0.937	0.501
s8								0.774

Table 7: Maximum likelihood estimates of the correlation coefficients, matrix form

	s1	s2	s3	s4	s5	s6	s7	s8
s1	1	0.1	0.1	0.1	0.1	0.2	0.2	0.4
s2		1.2	0.1	0.1	0.1	0.2	0.3	0.4
s3			1.2	0.1	0.2	0.2	0.3	0.4
s4				1.1	0.2	0.2	0.3	0.5
s5					1.1	0.2	0.3	0.5
s6						0.9	0.4	0.6
s7							0.9	0.6
s8								0.8

Table 8: Real covariance matrix from which the data are simulated

Evaluate log-likelihood and distance between real and estimated parameters

```
peLogLike <- matrix(evallike(citygam, cityX, nc, updbe, updsig, refineM))
smcemFrob <- sqrt(sum((updsig[upper.tri(updsig, diag = TRUE)] - pbitcov[upper.tri(pbitcov,
    diag = TRUE)])^2) + sum((updbe - pbitcoeff)^2))
smcemFrobNorm <- smcemFrob/sqrt(length(updbe) + length((updsig[upper.tri(updsig,
    diag = TRUE)])))
peLogLike <- cbind(peLogLike, smcemFrobNorm)
```

	log-likelihood	distance
1	-3278.996	0.067

Table 9: Estimated log-likelihood value and square root of the mean squared distance of the estimated parameters from the real ones

Evaluate standard errors

```
best <- matrix(0, nrow = nc, ncol = nobs)
for (i in 1:nobs) best[, i] <- cityX[[i]] %*% updbe

if (!constrained || useinvariance) dimInf <- length(updbe) + nc * (nc + 1)/2 -
    1 else dimInf <- length(updbe) + nc * (nc - 1)/2
totinfo <- matrix(0, nrow = dimInf, ncol = dimInf)
totHess <- matrix(0, nrow = dimInf, ncol = dimInf)
totestquad <- matrix(0, nrow = dimInf, ncol = dimInf)
totGradSq <- matrix(0, nrow = dimInf, ncol = dimInf)
totquval <- matrix(0, nrow = dimInf, ncol = dimInf)

for (j in 1:nobs) {
    samp <- obseval(citygam[, j], best[, j], updsig, refineM)
    obsInfo <- fisher(updbe, updsig, cityX[[j]], samp$samp, samp$W)
    totquval <- totquval + obsInfo$quval  ### total second moment expectation
    totGradSq <- totGradSq + obsInfo$gradsq  ### total expectation of gradient quadratic form
    totestquad <- totestquad + obsInfo$quad  ### total variance estimate
    totHess <- totHess + obsInfo$Hess  ### total hessian estimate
    totinfo <- totinfo + obsInfo$info  ###
}

myse <- sqrt(diag(solve(totinfo)))
```

	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
1	0.047	0.068	0.073	0.082	0.081	0.203

Table 10: Summary of the distribution of the standard errors of the estimated parmaters