simulation_gp

Joonwon Lee

2023-10-17

Define functions needed for experiments

1. exp_cov(theta,range,d): returns an exponential covariance matrix.

2. mat_cov(theta,range,d): returns a matern covariance matrix.

3. gen_exp_data(n_samples, theta, range): generates a finite sample of gaussian process using an exponential covariance matrix.

4. gen_mat_data(n_samples, theta, range): generates a finite sample of gaussian process using a matern covariance matrix.

```
exp_cov = function(theta,range,d){

  cov_mat = theta * exp(-d/range)
  chol_matrix = chol(cov_mat) # returns L^T where L is lower triangular.
  output = list(cov_mat=cov_mat,chol=chol_matrix)
  return(output)
}
mat_cov = function(theta,range,d){

  cov_mat = theta * (1 + d/range) * exp(-d/range)
  epsilon <- 1e-6  # Small positive constant
  c_regularized <- cov_mat + epsilon * diag(dim(cov_mat)[1])
  chol_matrix = chol(c_regularized) # returns L^T where L is lower triangular.
  output = list(cov_mat=cov_mat,chol=chol_matrix)
  return(output)
}

# Generate finite subset of Gaussian Process
gen_exp_data <- function(n_samples, theta, range){
  # Define the range of the square
  x_min <- 0
  x_max <- 10
  y_min <- 0
  y_max <- 10

  # Generate random x and y coordinates randomly
  x <- runif(n_samples, min = x_min, max = x_max)
  y <- runif(n_samples, min = y_min, max = y_max)
  # x <- seq(0,10, length.out = n)
```

```r
  # y <- seq(0,10, length.out = n)

  # distance matrix
  d <- sqrt(outer(x,x,"-")^2 + outer(y,y,"-")^2 )

  chol_matrix = exp_cov(theta,range,d)$chol
  z_list <- rnorm(n_samples)
  sim_data <- t(chol_matrix) %*% z_list
  # set location matrix and design matrix X.
  locs <- as.matrix(data.frame(x = x, y = y))
  X <- cbind(rep(1,n_samples), locs)
  output = list(sim_data = sim_data, locs = locs, X = X)
  return(output)
}

gen_mat_data <- function(n_samples, theta, range){
  # Define the range of the square
  x_min <- 0
  x_max <- 10
  y_min <- 0
  y_max <- 10
  x <- runif(n_samples, min = x_min, max = x_max)
  y <- runif(n_samples, min = y_min, max = y_max)

  d <- sqrt(outer(x,x,"-")^2 + outer(y,y,"-")^2 )  # Distance matrix

  chol_matrix = mat_cov(theta,range,d)$chol
  z_list <- rnorm(n_samples)
  sim_data <- t(chol_matrix) %*% z_list # simulate data

  locs <- as.matrix(data.frame(x = x, y = y)) # location matrix
  X <- cbind(rep(1,n_samples), locs) # design matrix X

  output = list(sim_data = sim_data, locs = locs, X = X)
  return(output)
}
```

covfun_name option in GpGp

We can fit the model using fit_model function in GpGp package by specifying covariance function as below:

matern15_scaledim: $M(x,y) = \sigma^2(1 + |D^{-1}|x - y||)exp(-||D^{-1}x - y||)$,

exponential_isotropic: $M(x,y) = \sigma^2 exp(-||x - y||/\alpha)$.

Remark: options in fit_model():

1. reorder=FALSE: maxmin ordering is not used unless nrow(locs) > 1e5.
2. m_seq: By default, a 10-neighbor approximaation is maximized , then a 30-neighbor approximation is maximized using 10 neighbor estimates as starting values.

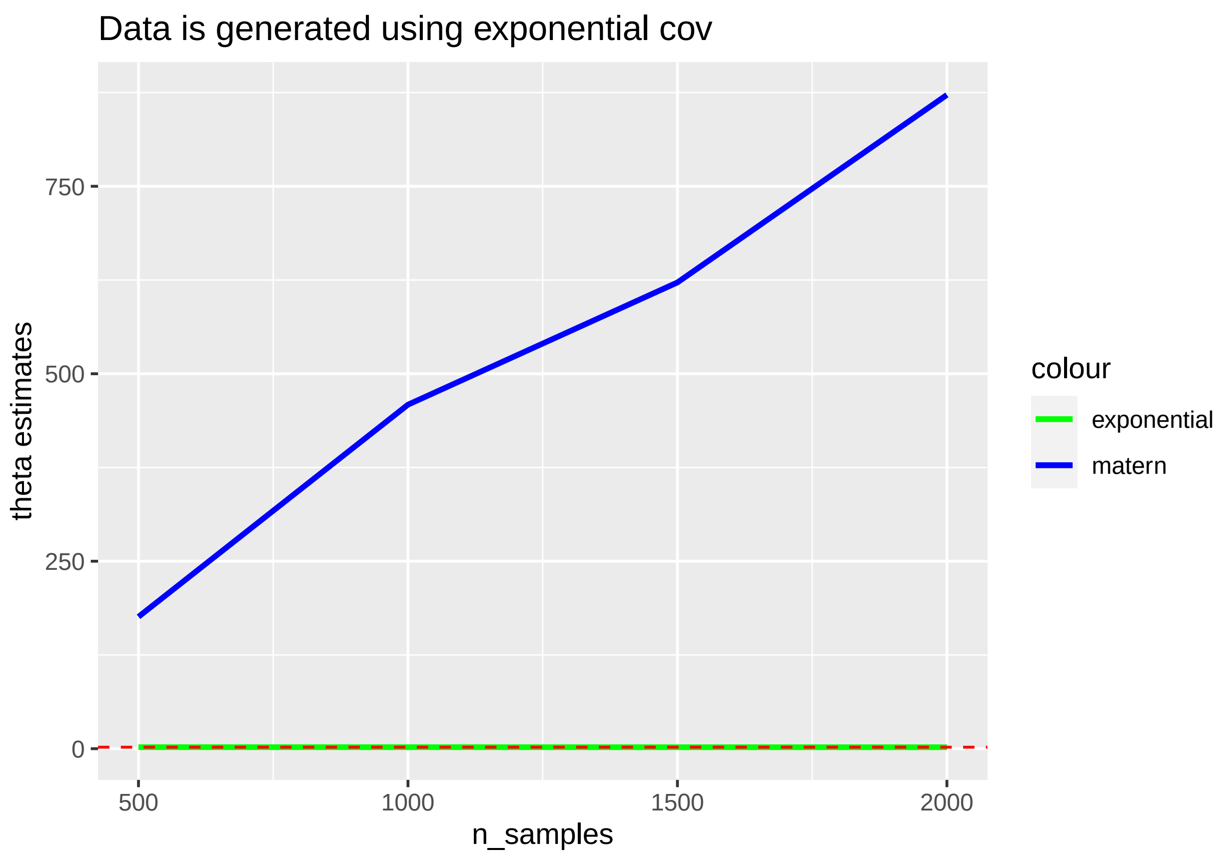
GpGp experiments

1. Truth covariance function is exponential, fitting model is matern.
2. Truth covariance function is matern, fitting model is exponential.

1. Truth covariance function is exponential, fitting model is matern:

Recall that exponential covariance function is: $M(x, y) = \sigma^2 \cdot exp(-||x - y||/\alpha)$.

```
n_samples = seq(500,2000, 500 )
scale_par = matrix(0, nrow = length(n_samples), ncol= 2)
i=1
theta = 2
range = 2
for (n in n_samples){
  # Generate data using exponential_isotropic
  exp_data = gen_exp_data(n_samples=n, theta=theta, range=range)
  # fit the model using 'matern15_scaledim'
  # fixed_parms = c(2,3) means fix range_1 and range_2. It must be used with
  # start_parms.
  mod1 = fit_model(exp_data$sim_data, locs = exp_data$locs, X = exp_data$X ,
                covfun_name = 'matern15_scaledim', fixed_parms = c(2,3), start_parms = c(3,2,2,0))

  # fit the model using the true covariance function
  mod2 = fit_model(exp_data$sim_data, locs = exp_data$locs, X = exp_data$X,
                covfun_name = 'exponential_isotropic',
  fixed_parms = c(2), start_parms = c(3,2,0))
  scale_par[i,1] = mod1$covparms[1]
  scale_par[i,2] = mod2$covparms[1]
  if (i< length(n_samples)){i = i + 1}
  else {}
}

df1 = data.frame(n = n_samples , sp_mat = scale_par[,1], sp_mat2 = scale_par[,2])
gg <- ggplot() +
  geom_line(data = df1, aes(x= n, y=sp_mat, color="matern"), linewidth=1) +
  geom_line(data = df1, aes(x= n, y=sp_mat2, color="exponential"), linewidth=1) +
  geom_hline(yintercept = theta, color = "red", linetype = "dashed") +
  labs(title = "Data is generated using exponential cov", x = "n_samples", y = "theta estimates") +
  scale_color_manual(values = c("matern" = "blue", "exponential" = "green"))

print(gg)
```



2. Truth covariance function is matern, fitting model is exponential:

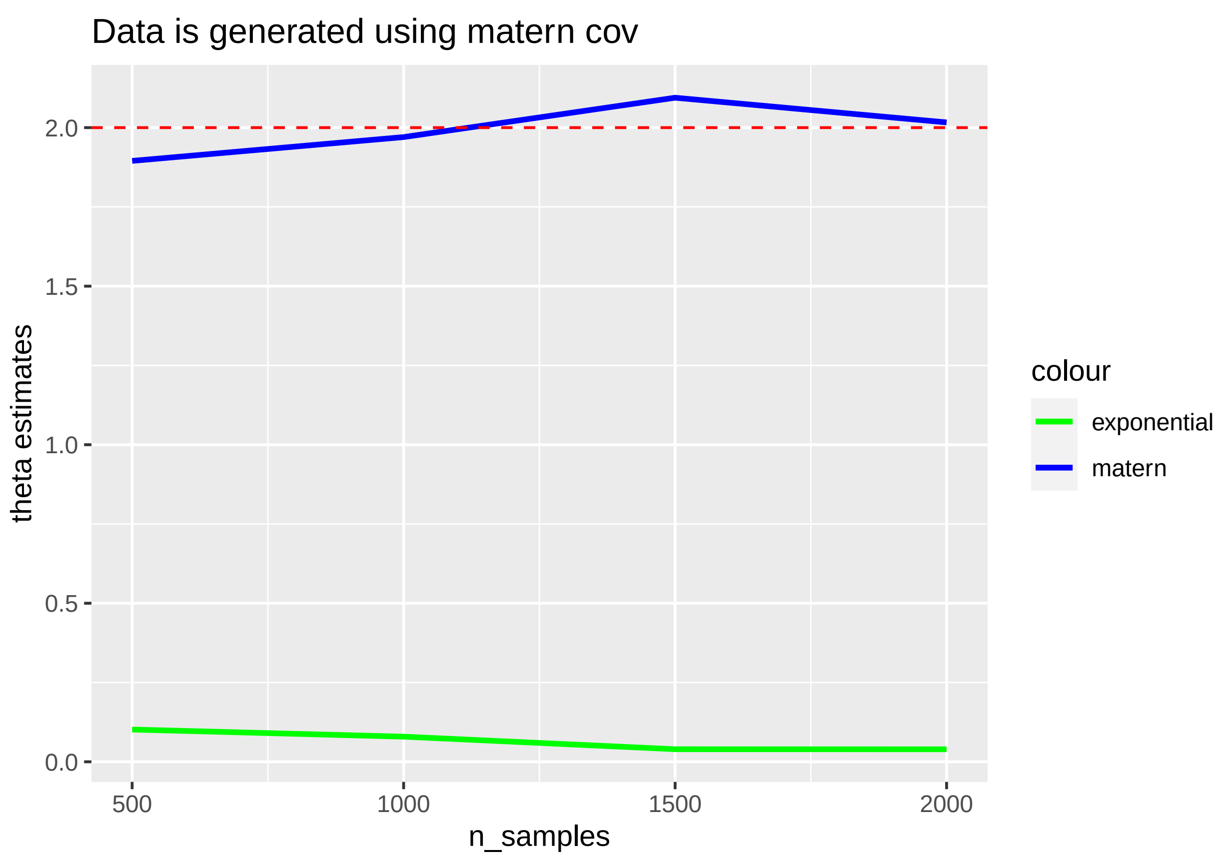
In this section, generate data using $M(x, y) = \sigma^2(1 + ||x - y||/\alpha)exp(-||D^{-1}x - y||/\alpha)$, and fit the model using exponential covariance function.

```
n_samples = seq(500,2000, 500 )
scale_par = matrix(0, nrow = length(n_samples), ncol= 2)
i=1
theta = 2
range = 2
for (n in n_samples){
  mat_data = gen_mat_data(n_samples=n, theta=theta, range=range)

  mod3 = fit_model(mat_data$sim_data, locs = mat_data$locs, X = mat_data$X ,
                  covfun_name = 'matern15_scaledim', fixed_parms = c(2,3), start_parms = c(3,2,2,0))

  mod4 = fit_model(mat_data$sim_data, locs = mat_data$locs, X = mat_data$X,
                  covfun_name = 'exponential_isotropic',
  fixed_parms = c(2), start_parms = c(3,2,0))
  scale_par[i,1] = mod3$covparms[1]
  scale_par[i,2] = mod4$covparms[1]
  if (i< length(n_samples)){i = i + 1}
}

df1 = data.frame(n = n_samples , sp_mat = scale_par[,1], sp_mat2 = scale_par[,2])
```

```r
gg <- ggplot() +
  geom_line(data = df1, aes(x= n, y=sp_mat, color="matern"), size=1) +
  geom_line(data = df1, aes(x= n, y=sp_mat2, color="exponential"), size=1) +
  geom_hline(yintercept = theta, color = "red", linetype = "dashed") +
  labs(title = "Data is generated using matern cov", x = "n_samples", y = "theta estimates") +
  scale_color_manual(values = c("matern" = "blue", "exponential" = "green"))
```

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```r
print(gg)
```



Sanity check using MLE

Given a sample generated from exponential covariance matrix, we can try fitting scale parameter using MLE of covariance matrix:

```r
# Generate 10000 observations of multivariate normal
# having exponential covariance function
n2 = 1000
```

```r
x_min <- 0
x_max <- 10
y_min <- 0
y_max <- 10
x <- runif(n2, min = x_min, max = x_max)
y <- runif(n2, min = y_min, max = y_max)
d <- sqrt(outer(x,x,"-")^2 + outer(y,y,"-")^2 )
theta = 2; range = 2
cov_mat <- theta * exp(-d/range)
data <- mvrnorm(n = n2, mu = rep(0,n2), Sigma = cov_mat)

# mle of covariance matrix
mle_cov = 1/n2 * t(data) %*% data

# fit model using matern15_scaledim: sigma^2 * (1+d) * exp(-d/range)

sigma_sq = diag(mle_cov %*% solve( (1+d/range) * exp(-d/range) ))

# fit model using true exponential: sigma^2 * exp(-d/range)

sigma_sq2 = diag(mle_cov %*% solve(exp(-d/range) ))

cat("Given ", n2, "samples with true [theta, range] =", matrix(c(2,2),nrow=1),
 ",\n\nestimate of (theta, range ) by fitting with matern15_isotropic are :", mean(sigma_sq),
"\n\nestimate of (theta, range ) by fitting with true exponential_isotropic are :", mean(sigma_sq2) )
```

```
## Given  1000 samples with true [theta, range] = 2 2 ,
##
## estimate of (theta, range ) by fitting with matern15_isotropic are : 718.4129
##
## estimate of (theta, range ) by fitting with true exponential_isotropic are : 2.004639
```

Predictions

Given new locations, we can predict new y values using predictions() in GpGp package. Then we can fit a new model using predicted values.

```r
n_new = 1000
x_min <- 0
x_max <- 10
y_min <- 0
y_max <- 10
x_new <- runif(n_new, min = x_min, max = x_max)
y_new <- runif(n_new, min = y_min, max = y_max)
locs_new <- as.matrix(data.frame(x = x_new, y = y_new)) # location matrix
X_new <- cbind(rep(1,n_new), locs_new) # design matrix X

pred = predictions(
fit = mod2,
locs_pred = locs_new,
X_pred  = X_new,
```

```
y_obs = mod2$y,
locs_obs = mod1$locs,
X_obs = mod2$X,
beta = mod2$betahat,
covparms = mod2$covparms,
covfun_name = mod2$covfun_name,
m = 60,
reorder = TRUE,
st_scale = NULL
)

mod5 = fit_model(pred, locs = locs_new, X = X_new,
                  covfun_name = 'exponential_isotropic',
  fixed_parms = c(2), start_parms = c(3,2,0))

cat("expected estimate of theta given new locations: ", mod5$covparms[1])
```