# Chen Simulation 2d

Qi

2023-04-15

In this file, we simulate the data and evaluate the performance of the 2-D non-stationary data. The data follows a process like follows:

$$Y(s) = sin\{30(\bar{s} - 0.9)^4\}cos\{2(\bar{s} - 0.9)\} + (\bar{s} - 0.9)/2$$

where $s = (s_x, s_y)^T \in R^2$ and $\bar{s} = \frac{s_x+s_y}{2}$. And the range of the coordinates are $[0, 1]$ and there are 900 samples from the grid 30 by 30 in surface $[0, 1]^2$. And here a three level multi-resolution model is used to generate the basis function, the basis are generated in the grid of 10 by 10, 19 by 19 and 37 by 37.

```
sim_coords <- expand.grid(seq(0,1,length.out = 30),seq(0,1,length.out = 30))
sim_sbar <- (sim_coords[,1] + sim_coords[,2])/2
sim_y <- sin(30*(sim_sbar-0.9)^4) * cos(2*(sim_sbar-0.9)) + (sim_sbar-0.9)/2
p_obs <-
ggplot() +
  geom_raster(aes(x = sim_coords[,1], y = sim_coords[,2], fill = sim_y)) +
  scale_fill_viridis_c() +
  labs(x = "Longitude",  y = "Latitude")

  basis_dist_1 <- spDists(as.matrix(sim_coords), as.matrix(basis_1))
  basis_dist_2 <- spDists(as.matrix(sim_coords), as.matrix(basis_2))
  basis_dist_3 <- spDists(as.matrix(sim_coords), as.matrix(basis_3))
  theta_1 <- 2.5* diff(seq(from = 0, to = 1, length.out = 10))[1]
  theta_2 <- 2.5* diff(seq(from = 0, to = 1, length.out = 19))[1]
  theta_3 <- 2.5* diff(seq(from = 0, to = 1, length.out = 37))[1]

  basis_fun_1 <- matrix(nychka_fun(basis_dist_1, theta = theta_1), nrow = nrow(sim_coords))
  basis_fun_2 <- matrix(nychka_fun(basis_dist_2, theta = theta_2), nrow = nrow(sim_coords))
  basis_fun_3 <- matrix(nychka_fun(basis_dist_3, theta = theta_3), nrow = nrow(sim_coords))
```

```
# Kriging with Matern function

# likfit(coords = sim_coords, data = sim_y, ini.cov.pars = c(1,0.1), fix.kappa = FALSE)
```

By applying the likfit function, we can get the MLE of the estimation of the Matern kernel parameters, and the MLE is $\phi = 4.2183, \kappa = 0.5$. That's just exponential correlation function with range 4.2183. And the $\sigma^2 = 0.1234$, with an intercept $\beta_0 = -0.1168$

Therefore the estimated processes of this is :

$$\hat{Y}(s) = -0.1168 + \epsilon(s)$$

where $\epsilon(s) \sim GP(\phi = 4.2183, exp, \sigma^2 = 0.1234)$

```r
pair_dist_2d <- spDists( as.matrix(sim_coords) )
cov_mat <- 0.1234 * exp(-pair_dist_2d/4.2183)
train_all_index <- sample(1:10, 900, replace = TRUE)
krig_mean_all <- rep(NA, 900)
dkrig_mean_all <- rep(NA, 900)
nn_mean_all <- rep(NA, 900)
```

```r
for (curr_index in 1:10) {
  train_index <- which(train_all_index != curr_index)
  train_coords <- sim_coords[train_index,]
  train_y <- sim_y[train_index]
  test_coords <- sim_coords[-train_index,]
  test_y <- sim_y[-train_index]

  # Classical Kriging
  exp_sig_11 <- cov_mat[train_index, train_index]
  exp_sig_12 <- cov_mat[train_index, -train_index]
  exp_sig_21 <- t(exp_sig_12)
  exp_sig_22 <- cov_mat[-train_index, -train_index]
  krig_mean_all[-train_index] <- -0.1168 + exp_sig_21 %*% solve(exp_sig_11) %*% matrix( as.numeric(trai




# dnn_mean_all

  x_tr <- array_reshape( as.matrix(train_coords), c(length(train_y), 2))
  x_te <- array_reshape( as.matrix(test_coords), c(length(test_y), 2))

  y_tr <- train_y
  y_te <- test_y


model_dnn <- keras_model_sequential()
model_dnn %>%
  layer_dense(units = 256, activation = 'relu', input_shape = c(ncol(x_tr))) %>%
  layer_dropout(rate = 0.4) %>%
  layer_dense(units = 256, activation = 'relu') %>%
  layer_dropout(rate = 0.4) %>%
  layer_dense(units = 128, activation = 'relu') %>%
  layer_dropout(rate = 0.2) %>%
  layer_dense(units = 1, activation = 'linear')

model_dnn %>% compile(
  loss = "mse",
  optimizer = optimizer_adam(),
  metrics = list("mse")
)

mod_train_dnn <- model_dnn %>%
  fit(x = x_tr, y = y_tr, epochs = 30, batch_size = 16)
```

```r
  nn_mean_all[-train_index] <- predict(model_dnn, x_te)


  x_tr <- cbind(train_coords, basis_fun_1[train_index,],
                basis_fun_2[train_index,],basis_fun_3[train_index,])

  x_te <- cbind(test_coords, basis_fun_1[-train_index,],
                basis_fun_2[-train_index,],basis_fun_3[-train_index,])


  x_tr <- array_reshape( as.matrix(x_tr), c(length(train_y), ncol(x_tr)))
  x_te <- array_reshape( as.matrix(x_te), c(length(test_y), ncol(x_tr)))



model_dk <- keras_model_sequential()
model_dk %>%
  layer_dense(units = 256, activation = 'relu', input_shape = c(ncol(x_tr))) %>%
  layer_dropout(rate = 0.4) %>%
  layer_dense(units = 256, activation = 'relu') %>%
  layer_dropout(rate = 0.4) %>%
  layer_dense(units = 128, activation = 'relu') %>%
  layer_dropout(rate = 0.2) %>%
  layer_dense(units = 1, activation = 'linear')

model_dk %>% compile(
  loss = "mse",
  optimizer = optimizer_adam(),
  metrics = list("mse")
)

mod_train_dk <- model_dk %>%
  fit(x = x_tr, y = y_tr, epochs = 30, batch_size = 16)

  dkrig_mean_all[-train_index] <- predict(model_dk, x_te)
}
```

```r
p_krig <-
  ggplot() +
  geom_raster(aes(x = sim_coords[,1], y = sim_coords[,2], fill = krig_mean_all)) +
  scale_fill_viridis_c() +
  labs(x = "Longitude",  y = "Latitude")

p_dnn <-
  ggplot() +
  geom_raster(aes(x = sim_coords[,1], y = sim_coords[,2], fill = nn_mean_all)) +
  scale_fill_viridis_c() +
  labs(x = "Longitude",  y = "Latitude")

p_dk <-
  ggplot() +
  geom_raster(aes(x = sim_coords[,1], y = sim_coords[,2], fill = dkrig_mean_all)) +
  scale_fill_viridis_c() +
```

```r
  labs(x = "Longitude",  y = "Latitude")
```

```r
mean((krig_mean_all - sim_y)^2)
```

```
## [1] 0.000276571
```

```r
mean((nn_mean_all - sim_y)^2)
```

```
## [1] 0.003453909
```

```r
mean((dkrig_mean_all - sim_y)^2)
```

```
## [1] 0.00267727
```