SDS 383D: Exercises 3 – Linear smoothing and Gaussian processes

February 15, 2017

Professor Scott

Spencer Woody

Problem 1

Basics

(A)

Problem 2

In nonparametric regression and spatial smoothing

(A)

R code for `myfuns03.R`

```
###########################################################
######### Created by Spencer Woody on 11 Feb 2017 #########
###########################################################

# ===========================================================================
# Gaussian process ==========================================================
# ===========================================================================

# ===========================================================================
# Gaussian process ==========================================================
# ===========================================================================

my.mvn <- function(n, mu, Sigma) {
    #  Simulate n draws from MVN(mu, Sigma)
    #
    #  Note: this function assumes that X already has an intercept term
    # (or doesn’t, if we want to force OLS through the origin)
    #
    # INPUTS:
    # n is the number of draws
    # mu is the mean vector
    # Sigma is the covariance matrix
    #
    # OUTPUT:
    # x is matrix of n draws from MVN(mu, Sigma) [with n rows, p columns]
    #

    # dimension of MVN
    p <- length(mu)

    # Check if inputs are valid (dimensions match, Sigma is square and p.s.d.)
    cond<- (ncol(Sigma) != p) |
           (nrow(Sigma) != p) |
           (max(eigen(Sigma)$values) <= 0)

    if (cond) {
        return("Try again...")
    }

    # Generate n*p univariate standard normal variables
    z     <- matrix(rnorm(n*p), nrow = p)

    # Create a matrix containing copies of mu
    mumat <- matrix(rep(mu, n), nrow = p)

    # Decompose Sigma into Sigma = L %*% Lt
    Lt <- chol(Sigma)

    # Generate sample with affine transformation of z
    x <- crossprod(Lt, z) + mumat

    return(t(x))
}

ell2 <- function(x) {
    # Compute the ell2 norm of x, a vector in Euclidean space

    return(sqrt(sum(x^2)))
}

C.SE <- function(x.i, x.j, params = NA) {
    # -----------------------------------------------------------------------
    #  Compute the (i, j) element of a squared exp. covariance matrix
    #
    # -----------------------------------------------------------------------
    # INPUTS:
    # x.i and x.j are two vectors in same space (need not be [0, 1])
    # params should be a vector of three hyperparameters
    #      1) b
    #      2) tau1.sq
    #      3) tau2.sq
    #
    # -----------------------------------------------------------------------
    # OUTPUT:
    # c.se is the value of the Matern-5/2 covariance matrix for x.i and x.j
    # -----------------------------------------------------------------------

    if (prod(is.na(params))) {
        return("Must have three valid parameters.")
    }

    if (length(params) != 3) {
        return("Must have three valid parameters.")
    }

    b       <- params[1]
    tau1.sq <- params[2]
    tau2.sq <- params[3]


    b       <- params[1]
    tau1.sq <- params[2]
    tau2.sq <- params[3]

    # Euclidean distance between x.i and x.j
    d <- ell2(x.i - x.j)

    c.se <- tau1.sq * exp(-0.5 * (d / b)^2) + tau2.sq * (x.i == x.j)

    return(c.se)
}

C.M52 <- function(x.i, x.j, params = NA) {
    # -----------------------------------------------------------------------
    #  Compute the (i, j) element of a Matern-5/2 covariance matrix
    #
    # -----------------------------------------------------------------------
    # INPUTS:
    # x.i and x.j are two vectors in same space (need not be [0, 1])
    # params should be a vector of three hyperparameters
    #       1) b
    #       2) tau1.sq
    #       3) tau2.sq
    #
    # -----------------------------------------------------------------------
    # OUTPUT:
    # c.m52 is the value of the Matern-5/2 covariance matrix for x.i and x.j
    # -----------------------------------------------------------------------

    if (prod(is.na(params))) {
        return("Must have three valid parameters.")
    }

    if (length(params) != 3) {
        return("Must have three valid parameters.")
    }

    b       <- params[1]
    tau1.sq <- params[2]
    tau2.sq <- params[3]

    # Euclidean distance between x.i and x.j
    d <- ell2(x.i - x.j)

    c.m52 <- tau1.sq * ( 1 + (5^0.5 * d / b) + (5 / 3 * (d / b)^2) ) *
             exp(-5^0.5 * d / b)  + tau2.sq * (x.i == x.j)

    return(c.m52)
}

make.covmat <- function(x, cov.fun, params = NA) {
    # -----------------------------------------------------------------------
    #  Compute the covariance matrix for a GP, given some cov. function
    #
    # -----------------------------------------------------------------------
    # INPUTS:
    # x is a vector of N values in [0, 1]
    # params should be a vector of three hyperparameters
    #       1) b
    #       2) tau1.sq
    #       3) tau2.sq
    #
    # -----------------------------------------------------------------------
    # OUTPUT:
    # covmat is the covariance matrix of GP
    # -----------------------------------------------------------------------

    if (prod(is.na(params))) {
        return("Must have three valid parameters.")
    }

    if (length(params) != 3) {
        return("Must have three valid parameters.")
    }

    N <- length(x)

    covmat <- matrix(nrow = N, ncol = N)

    for (j in 1:N) {
        for (i in j:N) {
            covmat[i, j] <- cov.fun(x[i], x[j], params = params)
            covmat[j, i] <- covmat[i, j]
        }
    }

    return(covmat)
}
```

R code for `exercises03.R`

```
###########################################################
######### Created by Spencer Woody on 11 Feb 2017 #########
###########################################################

# ===========================================================================
# Gaussian process ==========================================================
# ===========================================================================

library(ggplot2)

source(”myfuns03.R”)

x.seq <- seq(0, 1, length.out = 100)

b <- 1
tau1.sq <- 1e-6
tau2.sq <- 1e-5

myparams <- c(b, tau1.sq, tau2.sq)

xCM52 <- make.covmat(x.seq, C.M52, params = myparams)
xSE <- make.covmat(x.seq, C.SE, params = myparams)
```