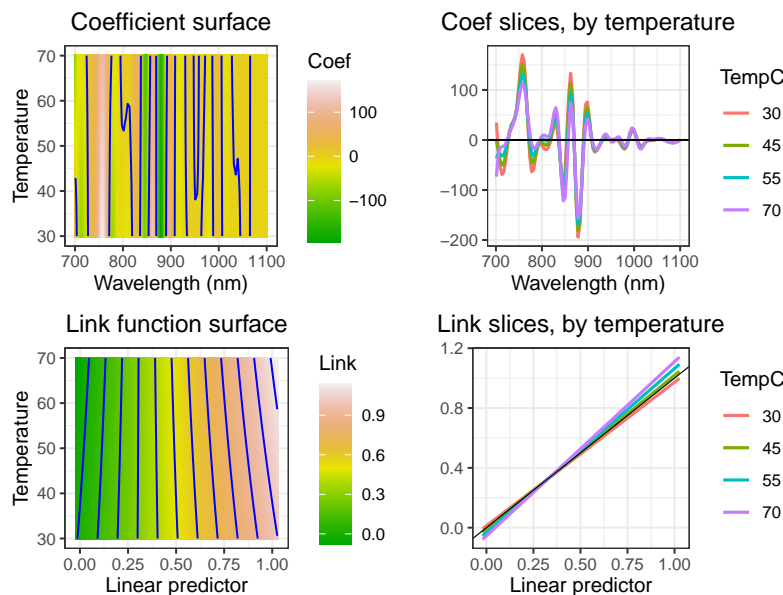


Varying-coefficient and varying-link single-index fit (Mixture data)



The coefficient surface (top, left) is sliced at specific levels of the covariate t , producing varying signal coefficients (top, right); The link function surface (bottom, left) is also sliced at specific levels of t , producing varying link functions (bottom, right). R code in `f-VSISR.R`

```
# Varying-coefficient and varying-link single-index fit (Mixture data)
# A graph in the book 'Practical Smoothing. The Joys of P-splines'
# Paul Eilers and Brian Marx, 2019

# Load libraries
library(fields) # Needed for plotting
library(data.table)
library(ucminf)
library(gridExtra)
library(ggplot2)
library(JOPS)
library(reshape2)

# Get the data
Dat <- Mixture

# Dimensions: observations, temperature index, signal
m <- 34
p1 <- 401
p2 <- 12

# Stacking mixture data, each mixture has 12 signals stacked
# The first differenced spectra are also computed.
mixture_data <- matrix(0, nrow = p2 * m, ncol = p1)
for (ii in 1:m)
{
  mixture_data[((ii - 1) * p2 + 1):(ii * p2), 1:p1] <-
    t(as.matrix(Dat$xspectra[ii, , ]))
  d_mixture_data <- t(diff(t(mixture_data)))
}

# Response (typo fixed) and index for signal
y_mixture <- Dat$fractions
y_mixture[17, 3] <- 0.1501 #(typo: already fixed in data Mixture)
```

```

index_mixture <- Dat$w1

# Select response and replicated for the 12 temps
# Column 1: water; 2: ethanediol; 3: amino-1-propanol
y <- as.vector(y_mixture[, 2])
y <- rep(y, each = p2)

# Search and design parameters
bdegs = c(3, 3, 3, 3)
pords <- c(2, 2, 2, 2)
nsegs <- c(27, 7, 7, 7)
mins <- c(701, 30)
maxs <- c(1100, 70)
M1_index <- seq(from = 701, to = 1100, by = 1)
M2_index <- c(30, 35, 37.5, 40, 45, 47.5, 50, 55, 60, 62.5, 65, 70)

Pars_ <- rbind(
  c(mins[1], maxs[1], nsegs[1], 3, 1, pords[1]),
  c(mins[2], maxs[2], nsegs[2], 3, 1, pords[2])
)

int <- T
max_iter = 50
Resol <- 100

# Defining x as first differenced spectra, number of channels.
x <- d_mixture_data
p1 <- p1 - 1

# Indices for train, valid, test sets
train <- c(1:13, 20, 22, 24)
valid <- c(14:19, 21, 23, 25)
test <- c(26:34)
ggroup <- c(rep(1, 13), rep(2, 6), 1, 2, 1, 2, 1, 2, rep(3, 9))
ggroup <- rep(ggroup, each = 12)
train_stack <- which(ggroup == 1)
valid_stack <- which(ggroup == 2)
test_stack <- which(ggroup == 3)

# Subset data (y, t, X) for train, valid, test sets
x_train <- x[train_stack, ]
x_valid <- x[valid_stack, ]
x_test <- x[test_stack, ]
y_train <- y[train_stack]
y_valid <- y[valid_stack]
y_test <- y[test_stack]

# Sort combined valid and test sets, use evens and odds
oo_ <- order(c(y_valid, y_test))
y_comb <- c(y_valid, y_test)[oo_]
x_comb <- (rbind(x_valid, x_test))[oo_, ]
onetwo <- rep(c(1:2), 9)
onetwo <- rep(onetwo, each = 12)
evens_ <- which(onetwo == 2)
odds_ <- which(onetwo == 1)
y_valid <- y_comb[evens_]
x_valid <- x_comb[evens_, ]
y_test <- y_comb[odds_]
x_test <- x_comb[odds_, ]
t_var <- rep(M2_index, m)
t_var_train <- t_var[train_stack]
t_var_valid <- t_var[valid_stack]
t_var_test <- t_var[test_stack]

# Wrapper function used in svcm clever search for log(lambda) vector
ucminfwrapper <- function(vec) {
  lambdas = 10^vec

```

```

fit <- sim_vcpsr(
  y_train, x_train, t_var_train, x_index = M1_index, nsegs,
  bdegs, lambdas, pords, max_iter = max_iter,
  mins = mins, maxs = maxs)
tmp1 <- predict(fit, X_pred = x_valid, t_pred = t_var_valid)
out <- sqrt(mean(((tmp1 - y_valid)^2)))
out
}

# Search for best (log) lambdas
op = ucminf(c(-5, 3, 3, 3), ucminfwrapper, control = list(trace=1)) #c(-5, 3, 3, 3)
opt_lam = c(10^op$par)

# Single-index VC model using optimal tuning, combining train and valid sets
fit <- sim_vcpsr(c(y_train, y_valid),
  rbind(x_train, x_valid), t_var[c(train_stack, valid_stack)],
  x_index = M1_index, nsegs, bdegs = bdegs, opt_lam,
  pords, max_iter = max_iter, mins = mins, maxs = maxs)

# External test set prediction error
pred <- predict(fit, X_pred = x_test, t_pred = t_var_test) # Prediction
test_err <- sqrt(mean((y_test - pred)^2)) # Test error
test_err

oorder <- order(fit$eta)
par(mfrow = c(2, 2))
Pars1 <- Pars_[1, ]
Pars2 <- Pars_[2, ]

# Prepare bases for estimated alpha surface
S_index_ <- seq(from = Pars1[1], to = Pars1[2], length = Resol)
oS <- outer(rep(1, Resol), S_index_)
Bx_ <- bbase(as.vector(oS), Pars1[1], Pars1[2], Pars1[3], Pars1[4])
t_index_ <- seq(from = Pars2[1], to = Pars2[2], length = Resol)
ot <- outer(t_index_, rep(1, Resol))
By_ <- bbase(as.vector(ot), Pars2[1], Pars2[2], Pars2[3], Pars2[4])

# Compute tensor products for estimated alpha surface
B1_ <- kronecker(Bx_, t(rep(1, ncol(By_))))
B2_ <- kronecker(t(rep(1, ncol(Bx_))), By_)
B_ <- B1_ * B2_
A_hat <- B_ %%% fit$alpha
A_hatm <- matrix(A_hat, Resol, Resol, byrow = T)

# Turn coeff matrix into a "long" data frame
Mu <- (A_hatm)
rownames(Mu) <- S_index_
colnames(Mu) <- t_index_
dens <- melt(Mu)
names(dens) <- c("x", "y", "Coef")

# Plot coefficient image with contours
sl <- T
ccol <- "blue"
plt1 <- ggplot(dens, aes(x, y, fill = Coef)) +
  geom_raster(show.legend = sl) +
  scale_fill_gradientn(colours = terrain.colors(20)) +
  xlab("Wavelength (nm)") + ylab("Temperature") +
  ggtitle("Coefficient surface") +
  geom_contour(aes(z = Coef), color = ccol, bins = 2, show.legend = T) +
  JOPS_theme()

xx <- seq(mins[1], maxs[1], length = Resol)
ahat <- A_hatm[, seq(1, Resol, length = 4)]
df <- data.frame(xall = c(xx, xx, xx, xx), yall = c(
  ahat[, 1], ahat[, 2],
  ahat[, 3], ahat[, 4 ]

```

```

), TempC = factor(rep(c(30, 45, 55, 70), each = Resol)))
plt1a <- ggplot(df) +
  geom_line(aes(x = xall, y = yall, colour = TempC, group = TempC), size = .8) +
  labs(x = "Wavelength (nm)", y = " ") +
  ggtitle("Coef slices, by temperature") +
  geom_hline(yintercept = 0) +
  JOPS_theme()

pPars <- rbind(
  c(min(fit$eta), max(fit$eta), nsegs[3], 3, opt_lam[3], pords[3]),
  c(min(fit$t_var), max(fit$t_var), nsegs[4], 3, opt_lam[4], pords[4])
)
eta_index_ <- seq(from = pPars[1, 1], to = pPars[1, 2], length = Resol)
oeta <- outer(rep(1, Resol), eta_index_)
t_index_ <- seq(from = pPars[2, 1], to = pPars[2, 2], length = Resol)
ot <- outer(t_index_, rep(1, Resol))
teta <- ps2DNormal(cbind(fit$eta, fit$t_var, fit$y),
  Pars = pPars,
  XYpred = cbind(as.vector(oeta), as.vector(ot))
)

fit_matrix <- matrix(teta$pred, Resol, Resol, byrow = TRUE)

# Turn link matrix into a "long" data frame
Mu <- (fit_matrix)
rownames(Mu) <- eta_index_
colnames(Mu) <- t_index_
dens <- melt(Mu)
names(dens) <- c("x", "y", "Link")

# Plot link image with contours
sl <- T
ccol <- "blue"
plt2 <- ggplot(dens, aes(x, y, fill = Link)) +
  geom_raster(show.legend = sl) +
  scale_fill_gradientn(colours = terrain.colors(20)) +
  xlab("Linear predictor") + ylab("Temperature") +
  ggtitle("Link function surface") +
  geom_contour(aes(z = Link), color = ccol, show.legend = T) +
  JOPS_theme()

xx <- eta_index_
etahat <- fit_matrix[, seq(1, Resol, length = 4)]
df <- data.frame(xall = c(xx, xx, xx, xx), yall = c(
  etahat[, 1], etahat[, 2],
  etahat[, 3], etahat[, 4 ]
), TempC = factor(rep(c(30, 45, 55, 70), each = Resol)))
plt2a <- ggplot(df) +
  geom_line(aes(x = xall, y = yall, group = TempC, colour = TempC), size = 0.8) +
  labs(x = "Linear predictor", y = " ") +
  ggtitle("Link slices, by temperature") +
  geom_abline(intercept = 0, slope = 1, size = 0.3) +
  JOPS_theme()

# Make and save pdf

grid.arrange(plt1, plt1a, plt2, plt2a, ncol = 2, nrow = 2)

```
