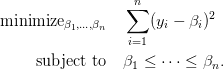
Let’s say we have data such that . (We assume no ties among the x_i‘s for simplicity.) ***Isotonic regression*** gives us a monotonic fit for the y_i‘s by solving the problem



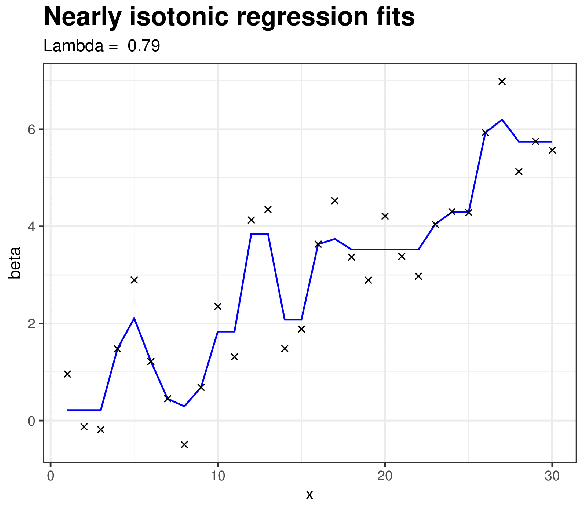
where  and is a user-specified hyperparameter.



It turns out that, due to properties of the optimization problem, the nearly-isotonic regression fit can be computed for ***all*** values in time, making it a practical method for use. See Section 3 and Algorithm 1 of Reference 1 for details. (More accurately, we can determine the nearly-isotonic regression fit for a critical set of values: the fit for any other other value will be a linear interpolation of fits from this critical set.)

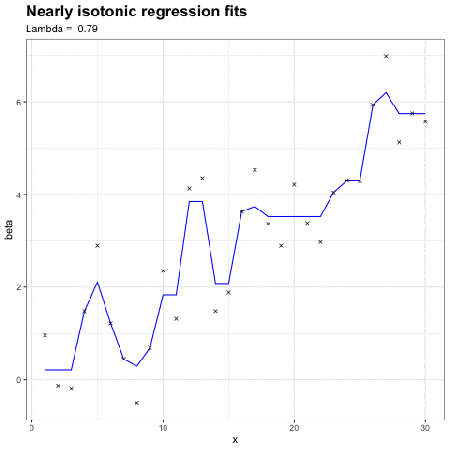
You can perform nearly-isotonic regression in R with the neariso package. The neariso() function returns fits for an entire path of values. The animation below shows how the fit changes as gets larger and larger

***Note 1:*** The formulation for nearly-isotonic regression above assumes that the points are equally spaced. If they are not, one should replace the penalty with



to account for the different-sized gaps. The neariso package only seems to handle the case where the x_i‘s are equally spaced.

***Note 2:*** The animation above was created by generating separate .png files for each value of , then stitching them together using the magick package. My initial hope was to create an animation that would transition smoothly between the different fits using the gganimate package but the transitions weren’t as smooth as I would have imagined them to be:



Does anyone know how this issue could be fixed? Code for the animation is below.

p <- ggplot(df, aes(x = x, y = beta)) +

geom\_path(col = "blue") +

geom\_point(data = truth\_df, aes(x = x, y = y), shape = 4) +

labs(title = "Nearly isotonic regression fits",

subtitle = paste("Lambda = ", "{lambda[as.integer(closest\_state)]}")) +

transition\_states(iter, transition\_length = 1, state\_length = 2) +

theme\_bw() +

theme(plot.title = element\_text(size = rel(1.5), face = "bold"))

animate(p, fps = 5)

Full Code Here

===============

library(neariso)

library(tidyverse)

library(gganimate)

# generate data

set.seed(51)

n <- 30

x <- 1:n

y <- rnorm(n) + 0.2 \* x

plot(x, y)

# perform nearly isotonic fit

fit <- neariso(y)

dim(fit$beta)

# make data frames for animation

df <- data.frame(iter = rep(1:ncol(fit$beta), each = n),

x = rep(1:n, times = ncol(fit$beta)),

beta = c(fit$beta))

truth\_df <- data.frame(x = x, y = y)

lambda <- round(fit$lambda, 2)

# animated plot: transitions don't seem great

p <- ggplot(df, aes(x = x, y = beta)) +

geom\_path(col = "blue") +

geom\_point(data = truth\_df, aes(x = x, y = y), shape = 4) +

labs(title = "Nearly isotonic regression fits",

subtitle = paste("Lambda = ", "{lambda[as.integer(closest\_state)]}")) +

transition\_states(iter, transition\_length = 1, state\_length = 2) +

theme\_bw() +

theme(plot.title = element\_text(size = rel(1.5), face = "bold"))

animate(p, fps = 5)

#anim\_save("neariso\_animation2.gif")

# alternative method for making plot: you will have to uncomment the

# lines below for it to work

library(magick)

for (idx in 1:ncol(fit$beta)) {

ggplot(filter(df, iter == idx), aes(x = x, y = beta)) +

geom\_path(col = "blue") +

geom\_point(data = truth\_df, aes(x = x, y = y), shape = 4) +

labs(title = "Nearly isotonic regression fits",

subtitle = paste("Lambda = ", lambda[idx])) +

theme\_bw() +

theme(plot.title = element\_text(size = rel(1.5), face = "bold"))

#ggsave(paste0("frame\_", idx, ".pdf"))

}

# create the animation

files <- sapply(1:ncol(fit$beta), function(idx) paste0("frame\_", idx, ".pdf"))

# image\_read(files, density = 300) %>% image\_animate(fps = 2) %>%

# image\_write(path = "neariso\_animation.gif")