Appendix: Predicting Good Surf Conditions Using Neural Networks

Using h2o on H20

Aaron Politsky

December 11, 2015

get_data.R

```
# read in data
col.names <- c("YY","MM","DD","hh","mm",
               "WDIR","WSPD","GST","WVHT","DPD","APD",
               "MWD","PRES","ATMP","WTMP","DEWP","VIS","TIDE")

data.path <- "data/"

asdf <- expand.grid(years, buoys)

library(plyr)
filenames <- alply(expand.grid(buoys, years), 1, function(row)
  as.character(paste0(row$Var1, "h", row$Var2, ".txt.gz")))

data.list <- lapply(filenames, function(filename)
  read.table(gzfile(description = paste0(data.path, filename)),
             col.names = col.names)
)
names(data.list) <- filenames
```

clean.data.helper.R

```
library(xts)
clean.data <- function(data) {
  data$datetime <-
    strptime(with(data[, 1:5],
                  paste(YY, MM, DD, hh, mm, sep="-")),
             format = "%Y-%m-%d-%H-%M", tz="GMT")

  data$WVHT <- ifelse(data$WVHT==99, NA, data$WVHT)
  data$DPD <-  ifelse(data$DPD==99, NA, data$DPD)
  data$MWD <-  ifelse(data$MWD==999, NA, data$MWD)

  data$WDIR <- ifelse(data$WDIR==999, NA, data$WDIR)
  data$WSPD <- ifelse(data$WSPD==99, NA, data$WSPD)
  data$GST <- ifelse(data$GST==99, NA, data$GST)
```

```r
  data$APD <- ifelse(data$APD==99, NA, data$APD)
  data$PRES <- ifelse(data$PRES==9999, NA, data$PRES)
  data$ATMP <- ifelse(data$ATMP==999, NA, data$ATMP)
  data$WTMP <- ifelse(data$WTMP==999, NA, data$WTMP)
  data$DEWP <- ifelse(data$DEWP==999, NA, data$DEWP)
  data$VIS <- ifelse(data$VIS==99, NA, data$VIS)
  data$TIDE <- ifelse(data$TIDE==99, NA, data$TIDE)

  data
}


go.surf <- function(data) {
  conditions <-
    (data$WVHT > 2) & (data$WVHT < 6) &
    (data$DPD > 12) & (data$DPD < 20)
  conditions
}

make.lag.list <- function(xts.data, lags, lag.hrs, offset.hrs=0) {
  lapply((0:lags)*lag.hrs*3600, function(lagsecs) {
    xts(xts.data, index(xts.data) + lagsecs + offset.hrs*3600)
  })
}

merge.lag.list <- function(lag.list) {
  base <- lag.list[[1]]
  for(l in lag.list[-1]) {
    base <- merge(base, l)
  }
  base
}
```

clean_data.R

```r
source('get_data.R')
source('clean.data.helper.R')

#empty.timestamp.seq <- seq(from=as.POSIXct(begin.2014), length.out=365*24*2, by = "30 mins")


waimea.12 <- clean.data(waimea.12)
waimea.13 <- clean.data(waimea.13)
waimea.14 <- clean.data(waimea.14)

waimea <- rbind(waimea.12, waimea.13, waimea.14)

waimea.12$go.surf <- go.surf(waimea.12)
waimea.13$go.surf <- go.surf(waimea.13)
waimea.14$go.surf <- go.surf(waimea.14)
```

```r
ak70.12 <- clean.data(ak70.12)
ak70.13 <- clean.data(ak70.13)
ak70.14 <- clean.data(ak70.14)

ak70 <- rbind(ak70.12, ak70.13, ak70.14)

sapply(ak70.12, function(x) mean(is.na(x))) # solid data
sapply(ak70.13, function(x) mean(is.na(x))) # missing wind data
sapply(ak70.14, function(x) mean(is.na(x))) # missing wind data, 20% missing wave data

ak72.12 <- clean.data(ak72.12)
ak72.13 <- clean.data(ak72.13)
ak72.14 <- clean.data(ak72.14)

ak72 <- rbind(ak72.12,
              ak72.13,
              ak72.14)

sapply(ak72.12, function(x) mean(is.na(x)))
sapply(ak72.13, function(x) mean(is.na(x)))
# sparse 2013 data
sapply(ak72.14, function(x) mean(is.na(x)))
# good 2014 data, despite it starting in June.

library(xts)

ak.active.cols <- c("WVHT", "DPD", "APD", "PRES", "ATMP", "WTMP", "DEWP")

xt.ak70.12 <- xts(ak70.12[,ak.active.cols], order.by = ak70.12$datetime)
xt.ak70.13 <- xts(ak70.13[,ak.active.cols], order.by = ak70.13$datetime)
xt.ak70.14 <- xts(ak70.14[,ak.active.cols], order.by = ak70.14$datetime)
sapply(xt.ak70.12, function(x) mean(is.na(x)))
sapply(xt.ak70.13, function(x) mean(is.na(x)))
sapply(xt.ak70.14, function(x) mean(is.na(x)))

xt.ak72.12 <- xts(ak72.12[,ak.active.cols], order.by = ak72.12$datetime)
xt.ak72.13 <- xts(ak72.13[,ak.active.cols], order.by = ak72.13$datetime)
xt.ak72.14 <- xts(ak72.14[,ak.active.cols], order.by = ak72.14$datetime)
xt.ak72    <- rbind(xt.ak72.12, xt.ak72.13, xt.ak72.14)

sapply(xt.ak72.12, function(x) mean(is.na(x)))
sapply(xt.ak72.13, function(x) mean(is.na(x)))
sapply(xt.ak72.14, function(x) mean(is.na(x)))

wa.active.cols <- c("WVHT", "DPD", "APD", "MWD", "go.surf")

sapply(waimea, function(x) mean(is.na(x)))

xt.wa.12 <- xts(waimea.12[,wa.active.cols], order.by = waimea.12$datetime)
xt.wa.13 <- xts(waimea.13[,wa.active.cols], order.by = waimea.13$datetime)
xt.wa.14 <- xts(waimea.14[,wa.active.cols], order.by = waimea.14$datetime)

sapply(xt.wa.12, function(x) mean(is.na(x)))
sapply(xt.wa.13, function(x) mean(is.na(x)))
sapply(xt.wa.14, function(x) mean(is.na(x)))

#xt.ak70.13 <- na.locf(xt.ak70.13, m)
#xt.ak70.14 <- na.locf(xt.ak70.14)
#xt.wa.13 <- na.locf(xt.wa.13)
#xt.wa.14 <- na.locf(xt.wa.14)

# create 6 days of 6h lags?
days <- 6
lag.hrs <- 6
lags <- days*24/lag.hrs

lagged.ak70.14.list <- make.lag.list(xt.ak70.14, lags, lag.hrs, ak70.14$datetime)

lagged.ak72.12.list <- make.lag.list(xt.ak72.12, lags, lag.hrs, ak72.12$datetime)
lagged.ak72.14.list <- make.lag.list(xt.ak72.14, lags, lag.hrs, ak72.14$datetime)

merged.ak70.14 <- xt.ak70.14
for(i in 1:lags) {
  merged.ak70.14 <- merge(merged.ak70.14, lagged.ak70.14.list[[i]])
}
merged.ak72.12 <- xt.ak72.12
for(i in 1:lags) {
  merged.ak72.12 <- merge(merged.ak72.12, lagged.ak72.12.list[[i]])
}
merged.ak72.14 <- xt.ak72.14
for(i in 1:lags) {
  merged.ak72.14 <- merge(merged.ak72.14, lagged.ak72.14.list[[i]])
}



lagged.ak70.13 <-
  lapply((1:lags)*lag.hrs*3600, function(lagsecs) {xts(xt.ak70.13, ak70.13$datetime+lagsecs)})
merged.ak70.13 <- xt.ak70.13
for(i in 1:lags) {
  merged.ak70.13 <- merge(merged.ak70.13, lagged.ak70.13[[i]])
}

base.ak.to.wa.lag <- 84*3600

lagged.wa.14 <- lapply((1:lags)*lag.hrs*3600 + base.ak.to.wa.lag, function(lagsecs) {
  xts(xt.wa.14, waimea.14$datetime+lagsecs)
})

lagged.wa.13 <- lapply((1:lags)*lag.hrs*3600 + base.ak.to.wa.lag, function(lagsecs) {
  xts(xt.wa.13, waimea.13$datetime+lagsecs)
})

lagged.wa.12 <- lapply((1:lags)*lag.hrs*3600 + base.ak.to.wa.lag, function(lagsecs) {
  xts(xt.wa.12, waimea.12$datetime+lagsecs)
})
```

sf.trial.R

```r
source('clean.data.helper.R')

# 46059: 359NM west of San Francisco
# 46237: San Francisco Bar
# 46002:  275NM West of Coos Bay, OR
buoys <- c("46059", "46002", "46237")
years <- c("2008", "2011")

source('get_data.R')

data.list <- lapply(data.list, clean.data)
sapply(data.list, function(d)
  sapply(d, function(x) mean(is.na(x))))

# SF 237 doesn't measure wind and has very few missing wave height and period readings
# West SF has good wind, wave, and weather data.
# West OR has great wind and wave data in '08, but only wave data in '11

sapply(data.list, dim)
# we have about twice as many output rows.  This could be to time-resoultion

# split out data from 2011 for now.
data.2011 <- data.list[c(4:6)]
data.2008 <- data.list[c(1:3)]

lapply(data.2008, function(x) x[1:15,c(4:5)])
lapply(data.2011, function(x) x[1:15,c(4:5)])
# at least at the beginning, our data appears to be either hourly or
# half-hourly.  This does not pose a problem in itself.

# create our go.surf condition:
data.2008[[3]]$go.surf <- go.surf(data.2008[[3]])
table(data.2008[[3]]$go.surf)
data.2011[[3]]$go.surf <- go.surf(data.2011[[3]])
table(data.2011[[3]]$go.surf)

# Converting to time series
library(xts)

input.cols <- c("WVHT", "DPD",  "APD",  "PRES", "ATMP", "WTMP")
input.list.2008 <- data.2008[c(1,2)]
output.list.2008 <- data.2008[3]
input.list.2011 <- data.2011[c(1,2)]
output.list.2011 <- data.2011[3]

xts.input.list.2008 <- lapply(input.list.2008, function(x)
  xts(x[,input.cols], order.by = x$datetime)
)
xts.input.list.2011 <- lapply(input.list.2011, function(x)
  xts(x[,input.cols], order.by = x$datetime)
)
```

```r
sapply(xts.input.list.2008, function(x) {
  plot(x$WVHT)
  plot(x$DPD)
})
sapply(xts.input.list.2011, function(x) {
  plot(x$WVHT)
  plot(x$DPD)
})

output.cols <- c("WVHT", "DPD",  "APD",  "MWD",  "WTMP", "go.surf")
xts.output.list.2008 <- lapply(output.list.2008, function(x)
  xts(x[,output.cols], order.by = x$datetime)
)
xts.output.list.2011 <- lapply(output.list.2011, function(x)
  xts(x[,output.cols], order.by = x$datetime)
)

library(ggplot2)
merged <- merge(xts.input.list.2011[[1]],
                xts.input.list.2011[[2]],
                xts.output.list.2011[[1]])
merged <- na.locf(merged, maxgap = 2)
sapply(merged, function(x) mean(is.na(x)))
# not bad.

window <- "2011-10/2011-11"
ggplot(merged[window], aes(x=index(merged[window]))) +
  geom_line(aes(y=WVHT, color = "Source CA")) +
  geom_line(aes(y=WVHT.1, color="Source OR")) +
  geom_line(aes(y=WVHT.2, color="Dest SF")) +
  scale_color_discrete(name="Buoy") +
  ylab("Wave Height") + xlab("Date")

window <- "2011-11-1/2011-11-15"
ggplot(merged[window], aes(x=index(merged[window]))) +
  geom_line(aes(y=WVHT, color = "Source CA")) +
  geom_line(aes(y=WVHT.1, color="Source OR")) +
  geom_line(aes(y=WVHT.2, color="Dest SF")) +
  scale_color_discrete(name="Buoy") +
  ylab("Wave Height") + xlab("Date") +
  ggtitle("Detecting Swell Travel Time")

window <- "2011-11-10/2011-11-12"
ggplot(merged[window], aes(x=index(merged[window]))) +
  geom_line(aes(y=WVHT, color = "Source CA")) +
  geom_line(aes(y=WVHT.1, color="Source OR")) +
  geom_line(aes(y=WVHT.2, color="Dest SF")) +
  scale_color_discrete(name="Buoy") +
  ylab("Wave Height") + xlab("Date")

# lets work on our output.  First, standardize its time.
# Create an empty timestamp sequence
xts.empty.2008 <-
```

```
  xts(order.by =
        seq(from=as.POSIXct("2008-01-01", tz="GMT"),
            to = as.POSIXct("2009-01-01", tz="GMT"),
            by = "30 mins")
  )
xts.empty.2011 <-
  xts(order.by =
        seq(from=as.POSIXct("2011-01-01", tz="GMT"),
            to = as.POSIXct("2012-01-01", tz="GMT"),
            by = "30 mins")
  )

# how do they compare in length?
length(index(xts.empty.2008))
length(index(xts.output.list.2008[[1]]))
length(index(xts.empty.2011))
length(index(xts.output.list.2011[[1]]))

# we are missing some values.  Let's identify the distribution of missing time
count(diff(index(xts.output.list.2008[[1]])))
# Mostly 30min intervals, a number of hour-long ones that we can fill in, and the occasional longer int
count(diff(index(xts.output.list.2011[[1]])))

# Let's merge them together now
xts.merged.output.2008 <- merge(xts.empty.2008, xts.output.list.2008[[1]])
xts.merged.output.2011 <- merge(xts.empty.2011, xts.output.list.2011[[1]])

# And fill in gaps of up to 120 min
xts.output.2008 <- na.locf(xts.merged.output.2008, maxgap = 3)
xts.output.2011 <- na.locf(xts.merged.output.2011, maxgap = 3)

# now we have some observations that are on :30  and between half hours.  toss the in-betweeners
xts.output.2008 <- merge(xts.empty.2008, xts.output.2008, join = "inner")
xts.output.2011 <- merge(xts.empty.2011, xts.output.2011, join = "inner")
# now we have our output series on the :30 and :60

sapply(xts.output.2008, function(x) mean(is.na(x)))
sapply(xts.output.2011, function(x) mean(is.na(x)))

# Let's convert to hourly data, the easy way
xts.output.2008 <- xts.output.2008[seq(1, nrow(xts.output.2008), by = 2)]
xts.output.2011 <- xts.output.2011[seq(1, nrow(xts.output.2011), by = 2)]

##############
#### Input ###

xts.empty.2008 <-
  xts(order.by =
        seq(from=as.POSIXct("2008-01-01", tz="GMT"),
            to = as.POSIXct("2009-01-01", tz="GMT"),
            by = "1 hour")
  )
xts.empty.2011 <-
```

```
  xts(order.by =
        seq(from=as.POSIXct("2011-01-01", tz="GMT"),
            to = as.POSIXct("2012-01-01", tz="GMT"),
            by = "1 hour")
  )


length(index(xts.empty.2008))
length(index(xts.input.list.2008[[1]]))
length(index(xts.input.list.2008[[2]]))
length(index(xts.empty.2011))
length(index(xts.input.list.2011[[1]]))
length(index(xts.input.list.2011[[2]]))

# some missing observations. especially in OR in 2011

count(diff(index(xts.input.list.2008[[1]])))
# Nice. only a few hourly obs are missing

count(diff(index(xts.input.list.2008[[2]])))
# only a couple outings over two hours.  Plus that big one in the summer.

count(diff(index(xts.input.list.2011[[1]])))
count(diff(index(xts.input.list.2011[[2]])))

# Let's merge them together now
xts.merged.input.2008 <- merge(xts.empty.2008,
                          xts.input.list.2008[[1]],
                          xts.input.list.2008[[2]])
sum(complete.cases(xts.merged.input.2008))
xts.merged.input.2011 <- merge(xts.empty.2011,
                               xts.input.list.2011[[1]],
                               xts.input.list.2011[[2]])
sum(complete.cases(xts.merged.input.2011))

# And fill in gaps of up to 120 min
xts.input.2008 <- na.locf(xts.merged.input.2008, maxgap = 3)
sum(complete.cases(xts.input.2008))
xts.input.2011 <- na.locf(xts.merged.input.2011, maxgap = 3)
sum(complete.cases(xts.input.2011))

# now we have some observations that are on our hourly grid  toss the in-betweeners
xts.input.2008 <- merge(xts.empty.2008, xts.input.2008, join = "inner")
xts.input.2011 <- merge(xts.empty.2011, xts.input.2011, join = "inner")
sum(complete.cases(xts.input.2008))
sum(complete.cases(xts.input.2011))

# let's create a window of time in our input buoys:  how about 3 days of lags
#input.lag.list <- make.lag.list(xts.input, lags=72, lag.hrs = 1)
input.lag.list.2008 <- make.lag.list(xts.input.2008, lags=12, lag.hrs = 1, offset.hrs = 10)
input.2008 <- merge.lag.list(input.lag.list.2008)
count(diff(index(input.2008)))
```

```r
input.lag.list.2011 <- make.lag.list(xts.input.2011, lags=12, lag.hrs = 1, offset.hrs = 10)
input.2011 <- merge.lag.list(input.lag.list.2011)
count(diff(index(input.2011)))

dim(input.2008)
sum(complete.cases(input.2008))
# plenty of complete rows
plot(y=complete.cases(input.2008), x=index(input.2008))

dim(input.2011)
sum(complete.cases(input.2011))
plot(y=complete.cases(input.2011), x=index(input.2011))
# not so many, but winter is OK

# Train on 2008, test on 2011
train <- merge(xts.output.2008$go.surf, input.2008)
test  <- merge(xts.output.2011$go.surf, input.2011)

sum(complete.cases(train))
sum(complete.cases(test))

mean(train$go.surf, na.rm=T)
mean(test$go.surf, na.rm=T)
# enough, reasonably balanced

library(h2o)

#set.seed(99)
#train.indices <- sample(1:nrow(all), size = nrow(all)*.75)
#train <- all[train.indices,]
#test <- all[-train.indices,]

#mean(train$go.surf, na.rm=T)
#mean(test$go.surf, na.rm=T)
# balanced

#########################
# modeling

# start or connect to h2o server
h2oServer <- h2o.init(max_mem_size="4g", nthreads=-1)

# we need to load data into h2o format
train_hex = as.h2o(data.frame(x=train[,-1], y=as.factor(train[,1])))
test_hex = as.h2o(data.frame(x=test[,-1], y=as.factor(test[,1])))

predictors <- 1:(ncol(train_hex)-1)
response <- ncol(train_hex)

hyper.params <-
  list(
    epochs=c(2),
    hidden=list(#c(1024, 1024, 1024), #c(1024,512,256),
                 #c(512, 512),
                 c(1024), #c(512), c(256), c(128),
                 c(64)),
    activation=c("Tanh", "TanhWithDropout")
  )


hyper.params <-
  list(
    epochs=c(2),
    hidden=list(c(1024, 1024, 1024),# c(1024,512,256),
                 #c(512, 512),
                 c(1024), #c(512), c(256), c(128),
                 c(64)),
    activation=c("Tanh", "TanhWithDropout")
  )

set.seed(99)
system.time(
  dl.grid <- h2o.grid(
    algorithm = "deeplearning",
    x=predictors, y=response,
    training_frame=train_hex,
    classification_stop=-1,  # Turn off early stopping
    l1=1e-5,
    hyper_params = hyper.params
  )
)
summary(dl.grid)
dl.grid.models <- lapply(dl.grid@model_ids, function(id) h2o.getModel(id))
#model.paths.12hr.10offset.ca.or.sf.2008.train <-
#  lapply(dl.grid.models, function(m) h2o.saveModel(m, path="models"))
#save(model.paths.12hr.10offset.ca.or.sf.2008.train,
#     file="model.paths.12hr.10offset.ca.or.sf.2008.train.Rda")
load(file = "model.paths.12hr.10offset.ca.or.sf.2008.train.Rda")
dl.grid.models <- lapply(model.paths.12hr.10offset.ca.or.sf.2008.train, function(p) h2o.loadModel(p))

ptest.list <- lapply(dl.grid.models, function(m) h2o.performance(m, test_hex))
cm.test.list <- lapply(ptest.list, function(ptest) h2o.confusionMatrix(ptest))

library(plyr)
ptest.df <- ldply(cm.test.list,
                  function(cm)
                    c(tot.test.error.rate = cm$Error[3]))
ptest.df <- cbind(ptest.df, expand.grid((hyper.params)))
ptest.df
best.model.index <- which.min(ptest.df$tot.test.error.rate)
best.dl.model <- dl.grid.models[[best.model.index]]
cm.test.list[[best.model.index]]

plot(ptest.list[[best.model.index]])
prediction.2011 <- as.data.frame(h2o.predict(dl.grid.models[[best.model.index]], newdata = test_hex))
```

```r
hyper.params <-
  list(
    epochs=c(2),
    hidden=list(c(1024, 1024, 1024),# c(1024,512,256),
                #c(512, 512),
                c(1024), #c(512), c(256), c(128),
                c(64)),
    activation=c("Tanh", "TanhWithDropout")
  )

set.seed(99)
system.time(
  dl.grid <- h2o.grid(
    algorithm = "deeplearning",
    x=predictors, y=response,
    training_frame=train_hex,
    classification_stop=-1,  # Turn off early stopping
    l1=1e-5,
    hyper_params = hyper.params
  )
)

summary(dl.grid)
dl.grid.models.short <- lapply(dl.grid@model_ids, function(id) h2o.getModel(id))
#model.paths.short <- lapply(dl.grid.models.short, function(m) h2o.saveModel(m, path="models"))
#save(model.paths.short, file="model.paths.short.Rda")

ptest.list.short <- lapply(dl.grid.models.short, function(m) h2o.performance(m, test_hex))
cm.test.list.short <- lapply(ptest.list.short, function(ptest) h2o.confusionMatrix(ptest))

library(plyr)
ptest.df.short <- ldply(cm.test.list.short,
                  function(cm)
                    c(tot.test.error.rate = cm$Error[3]))
ptest.df.short <- cbind(ptest.df.short, expand.grid((hyper.params)))

best.model.index.short <- which.min(ptest.df.short$tot.test.error.rate)
best.dl.model.short <- dl.grid.models.short[[best.model.index.short]]
cm.test.list.short[[best.model.index.short]]

plot(ptest.list.short[[best.model.index.short]])
# not bad at all.
```

alaska_hawaii.R

```r
source('clean.data.helper.R')

# West to East:
# 46070:  SOUTHWEST BERING SEA SOUTHWEST BERING SEA - 142NM NNE OF ATTU IS, AK
# 46072:  CENTRAL ALEUTIANS 230NM
# 46073:  SOUTHEAST BERING SEA -205NM WNW of Dutch Harbor, AK
# 46066:  SOUTH KODIAK, AK  (good years:  2003, 2007, 2010, 2014)

# 51201:  Waimea, North Shore of Oahu, HI
#buoys <- c("46070", "46072", "46073", "46066", "51201")
buoys <- c("46070", "46066", "51201")
years <- c("2007", "2014")
source('get_data.R')

data.list <- lapply(data.list, clean.data)
sapply(data.list[c(1:(length(data.list)/2))], function(d)
  sapply(d[,c(6:15)], function(x) mean(is.na(x))))
# '70 has poor wind data
# most others look good.

sapply(data.list[-c(1:(length(data.list)/2))], function(d)
  sapply(d[,c(6:15)], function(x) mean(is.na(x))))
# '70 again, has poor wind data.  we would need to exclude wind from the model
# '70 is missing 20% of its wave data.  we may be able to deal with this,
#    depending on when it occurs.  Let's find out

input.cols <- c("WVHT", "DPD", "APD", "PRES", "ATMP", "WTMP")

sapply(data.list, dim)
# potential trouble with '27 in '07

# train on 07, test on 14
train.list <- data.list[c(1:length(buoys))]
train.input.list <- train.list[1:(length(buoys)-1)]
train.output.list <- train.list[length(buoys)]

test.list  <- data.list[-c(1:length(buoys))]
test.input.list <- test.list[1:(length(buoys)-1)]
test.output.list <- test.list[length(buoys)]

library(xts)

xts.train.input.list <- lapply(train.input.list, function(x)
  xts(x[,input.cols], order.by = x$datetime)
)
xts.test.input.list <- lapply(test.input.list, function(x)
  xts(x[,input.cols], order.by = x$datetime)
)

sapply(names(xts.train.input.list), function(name) {
  plot(xts.train.input.list[[name]]$WVHT, main = name)
})
sapply(names(xts.train.input.list), function(name) {
  plot(xts.train.input.list[[name]]$DPD, main = name)
})

# in 2007, 66, 73, and 70 are great, 72 is bad

sapply(names(xts.test.input.list), function(name) {
  plot(xts.test.input.list[[name]]$WVHT, main = name)
})
sapply(names(xts.test.input.list), function(name) {
  plot(xts.test.input.list[[name]]$DPD, main = name)
})

# in 2008, 66 stops in september, 72 is good, 70 fades in october
# in 2014, 66 is great, but 70 fades in november
#  Let's use 66 and 70, which are geographically distant,
#    train on 2007, test on 2014

#########################################
#  Input Prep

lapply(xts.train.input.list, function(x) count(diff(index(x))))
lapply(xts.test.input.list, function(x) count(diff(index(x))))
# input is hourly

xts.empty.train <-
  xts(order.by =
        seq(from=as.POSIXct("2007-01-01", tz="GMT"),
            to = as.POSIXct("2008-01-01", tz="GMT"),
            by = "60 mins")
  )
xts.empty.test <-
  xts(order.by =
        seq(from=as.POSIXct("2014-01-01", tz="GMT"),
            to = as.POSIXct("2015-01-01", tz="GMT"),
            by = "60 mins")
  )

dim(xts.train.input.list[[1]])
dim(xts.train.input.list[[2]])
dim(xts.test.input.list[[1]])
dim(xts.test.input.list[[2]])

lapply(xts.train.input.list, function(x) count(diff(index(x))))
# training data is pretty tight given it was active

lapply(xts.test.input.list, function(x) count(diff(index(x))))
# as is test, but a bit less so


# merge into our standardized time
xts.merged.train.input <-
```

```r
  merge(xts.empty.train,
        xts.train.input.list[[1]],
        xts.train.input.list[[2]])
mean(complete.cases(xts.merged.train.input))
dim(xts.merged.train.input)

xts.merged.test.input <-
  merge(xts.empty.test,
        xts.test.input.list[[1]],
        xts.test.input.list[[2]])
mean(complete.cases(xts.merged.test.input))
dim(xts.merged.test.input)
# some time issues

# fill some gaps up to a limit
xts.train.input <- na.locf(xts.merged.train.input, maxgap = 4)
mean(complete.cases(xts.train.input))
plot(complete.cases(xts.merged.train.input))
plot(complete.cases(xts.merged.train.input))

xts.test.input  <- na.locf(xts.merged.test.input, maxgap = 4)
mean(complete.cases(xts.test.input))
plot(complete.cases(xts.merged.test.input))
plot(complete.cases(xts.test.input))
dim(xts.test.input) # we still need to collapse down to the grid

# now we need to only keep those obs that are on our hourly grid.  toss the in-betweeners
xts.train.input <- merge(xts.empty.train, xts.train.input, join = "inner")
mean(complete.cases(xts.train.input))
xts.test.input <- merge(xts.empty.test, xts.test.input, join = "inner")
mean(complete.cases(xts.test.input))
# interpolate with a spline curve up to a point
before <- xts.test.input
after <-  na.spline(xts.test.input, maxgap = 36)
merged <- merge(after, before)
window <- "2014-09"
ggplot(merged[window], aes(x=index(merged[window]))) +
  geom_line(aes(y=WVHT, color="After")) +
  geom_line(aes(y=WVHT.2, color = "Before")) +
  scale_color_discrete(name="") +
  ylab("Wave Height") + xlab("Date") +
  ggtitle("Imputing Data Using Cubic Spline Interpolation")

xts.test.input <- na.spline(xts.test.input, maxgap = 36) # needed to do something
mean(complete.cases(xts.test.input))
plot(complete.cases(xts.test.input))

dim(xts.train.input)
dim(xts.test.input)
# very nice.

# Lets create a lagged window of input.  Since we are far from our destination,
# we can and will need to use a wider window centered at a longer time away from
# the destination due to swell travel time


input.train.window <-
  merge.lag.list(
    make.lag.list(xts.train.input,
                  lags=96,
                  lag.hrs = 2,
                  offset.hrs = 96))

dim(input.train.window)
mean(complete.cases(input.train.window))

input.test.window <-
  merge.lag.list(
    make.lag.list(xts.test.input,
                  lags=96,
                  lag.hrs = 2,
                  offset.hrs = 96)
  )
plot(complete.cases(input.test.window))
dim(input.test.window)
mean(complete.cases(input.test.window))


#########################################
#  Output Prep

train.output.list[[1]]$go.surf <- go.surf(train.output.list[[1]])
test.output.list[[1]]$go.surf <- go.surf(test.output.list[[1]])

xts.train.output.list <- lapply(train.output.list, function(x)
  xts(x$go.surf, order.by = x$datetime)
)
xts.test.output.list <- lapply(test.output.list, function(x)
  xts(x$go.surf, order.by = x$datetime)
)

xts.train.output <- xts.train.output.list[[1]]
xts.test.output <- xts.test.output.list[[1]]

names(xts.train.output) = "go.surf"
names(xts.test.output) = "go.surf"

length(index(xts.empty.train))
length(index(xts.train.output))
length(index(xts.empty.test))
length(index(xts.test.output))
# probably double time

# we are missing some values.  Let's identify the distribution of missing time
count(diff(index(xts.train.output)))
count(diff(index(xts.test.output)))
```

```r
# Let's merge them together now
xts.merged.train.output <- merge(xts.empty.train, xts.train.output)
xts.merged.test.output  <- merge(xts.empty.test,  xts.test.output)

xts.train.output <- na.locf(xts.merged.train.output, maxgap = 3)
xts.test.output  <- na.locf(xts.merged.test.output,  maxgap = 3)

# now we have some observations that are on :30  and between half hours.  toss the in-betweeners
xts.train.output <- merge(xts.empty.train, xts.train.output, join = "inner")
xts.test.output <- merge(xts.empty.test, xts.test.output, join = "inner")
# now we have our output series on the :30 and :60

dim(xts.train.output)
dim(xts.test.output)
mean(is.na(xts.train.output))
mean(is.na(xts.test.output))

count(xts.test.output[complete.cases(input.train.window)])


##### Training and Testing


train <- merge(xts.train.output$go.surf, input.train.window)
test  <- merge(xts.test.output$go.surf,   input.test.window)

mean(train$go.surf, na.rm=T)
mean(test$go.surf, na.rm=T)

library(h2o)

# start or connect to h2o server
h2oServer <- h2o.init(max_mem_size="4g", nthreads=-1)

# we need to load data into h2o format
train_hex = as.h2o(data.frame(x=train[,-1], y=as.factor(train[,1])))
test_hex = as.h2o(data.frame(x=test[,-1], y=as.factor(test[,1])))

predictors <- 1:(ncol(train_hex)-1)
response <- ncol(train_hex)

hyper.params <-
  list(
    epochs=c(2),
    hidden=list(#c(1024, 1024, 1024),
      #c(1024,512,256)
      c(64,64,64),
      c(64,64),
    #  c(512, 512),
      #,c(1024), #c(512), c(256), c(128)
      c(256),
      c(64)
    ),
    activation=c("Tanh", "TanhWithDropout"),
    #    input_dropout_ratio=c(0, .2),
    #    l1 = c(1e-5, 0.2)
    #activation=c("TanhWithDropout"),
    input_dropout_ratio=c(0, .2),
    l1 = c(1e-5, 0.2)
  )
expand.grid(hyper.params)

set.seed(99)
runtime <- system.time(
  dl.grid <- h2o.grid(
    algorithm = "deeplearning",
    x=predictors, y=response,
    training_frame=train_hex,
    classification_stop=-1,  # Turn off early stopping
    #l1=1e-5,
    variable_importances = T,
    hyper_params = hyper.params
  )
)
summary(dl.grid)
dl.grid.models <- lapply(dl.grid@model_ids, function(id) h2o.getModel(id))
#model.paths <-
#  lapply(dl.grid.models, function(m) h2o.saveModel(m, path="models"))
#save(model.paths,
#     file="model.paths.12.11.150pm.Rda")
#save(hyper.params,
#     file="hyper.params.12.11.150pm.Rda")
load(file="model.paths.12.11.150pm.Rda")
load(file="hyper.params.12.11.150pm.Rda")
#load(file = "model.paths.12hr.10offset.ca.or.sf.2008.train.Rda")
dl.grid.models <- lapply(model.paths, function(p) h2o.loadModel(p))

ptest.list <- lapply(dl.grid.models, function(m) h2o.performance(m, test_hex))
cm.test.list <- lapply(ptest.list, function(ptest) h2o.confusionMatrix(ptest))

library(plyr)
ptest.df <- ldply(cm.test.list,
                  function(cm)
                    c(tot.test.error.rate = cm$Error[3]))
ptest.df <- cbind(ptest.df, expand.grid((hyper.params)))
ptest.df
best.model.index <- which.min(ptest.df$tot.test.error.rate)
best.dl.model <- dl.grid.models[[best.model.index]]
cm.test.list[[best.model.index]]

plot(ptest.list[[best.model.index]], main = "True Positive Rate vs. False Positive Rate\n96 hour Foreca
#prediction.2011 <- as.data.frame(h2o.predict(dl.grid.models[[best.model.index]], newdata = test_hex))
print(runtime)
```