Standardization Considerations for K Nearest Neighbors

Brad Rafferty

6/7/2020

Introduction

As the number of neighbors, K, grows, the decision boundary of the K nearest neighbors model becomes less flexible. While variance will likely decrease as a result, the bias will increase. Although a larger K may help to combat noise in the data, eventually the nearest-neighbors estimate will not adequately capture the underlying response function. A sufficiently-large K can avoid being overly-flexible (high variance) while adequately modeling the underlying function well (low bias).

Standardizing each variable to have the same variance can be expressed in the Euclidian distance sense as follows:

x'_{ij} <- $(x_{ij} - \overline{x}_j)/[variance(x_{ij})_{i=1}^{N}]^{1/2}$

The step gives the predictors **equal influence**. Depending on the nature of the data, this may or may not be a desirable pre-processing step.

- Advantages:
  - Removes dependence on units of the variable (e.g., doesn't matter if height is measured in feet or milimeters) or, in general, the range in each predictor
  - The standardization will improve the performance of nearest neighbors when the predictors that are included and standardized all are important and all have approximately equal influences in reality
- Disadvantages:
  - The target function $f^*(x_1, \ldots x_n)$ does not always have the same degree of dependnece on all variables
  - Including variables in the training data and subsequently standardizing them that are not in $f^*(x_1, \ldots x_n)$ will end up hurting the model!

In higher-dimensional problem, feature selection becomes especially important from a model accuracy standpoint, and also for computational efficiency considerations.

Case Study

Let's say that the variance of x_1 is much greater than that of x_2, e.g. $\sigma_{x_1}^2 = 9\sigma_{x_2}^2$. Our structural model is of the form $\hat{f}(\overline{x}; a) = a_1x_1 + a_2x_2$. The target function $f^*(x_1; a) = a_1x_1 + \epsilon$ (i.e. it is only a function of x_1, but, of course, we do not know this), where ϵ is some irreducible error. In other words, x_2 is measured, but it is not relevant to the target function. We will see in this example that including standardization in the data pre-processing will penalize our nearest-neighbors regression result.

Configure the simulation dataset

```
library(FNN)

N      <- 1000          # Number of observations
var_x2  <- .5           # Variance of x_2
var_x1  <- 9*var_x2     # Variance of x_1
sig <- 0.1              # Standard deviation for random normal noise

# Create the dataset
# Set seeds before every rnorm() to make problem reproducible
set.seed(20200607)
x1  <- rnorm(N, mean = 0, sd = sqrt(var_x1)) # Predictor var 1
set.seed(20200608)
x2  <- rnorm(N, mean = 0, sd = sqrt(var_x2)) # Predictor var 2
set.seed(20200609)
a   <- rnorm(2, mean = 0, sd = 1) # Weight parameters
set.seed(20200610)
del <- rnorm(N, mean = 0, sd = 0.1) # Standard normal noise
y   <- x1 * a[[2]] + del # Response vector / target function

dat <- data.frame(x1 = x1, x2 = x2) # Dataset (response not in dataset)

# Sample the dataset to split into train and test sets
idcs_train <- sample(1:length(y), floor(0.9*length(y))) # Indices of test
data

target_train <- y[idcs_train]
target_test <- y[-idcs_train]
```

Model the dataset with nearest-neighbors, K = 5, no standardization

```
# Without standardization
data_train <- dat[idcs_train,]
data_test <- dat[-idcs_train,]

# Perform KNN regression
pr <- knn.reg(data_train, data_test, target_train, k = 5)

# Pull predicted response from prediction variable
f_hat <- pr$pred

# Calculate loss (mean squared error) of this no-standardization case
loss_no_stand <-  mean( (target_test - f_hat)^2 )
```

Model the dataset with nearest-neighbors, K = 5, standardization

```
# With standardization; see Nearest Neighbors class notes for the expression
# Define standarization (divide by variance for Euclidian distance, i.e. p =
2)
stand <- function(x) {(x-mean(x)) / var(x)^0.5 }

# Apply standardization to each predictor variable in the data
dat_stand <- as.data.frame(lapply(dat[,], stand))

# Index train and test sets per usual
data_train_stand <- dat_stand[idcs_train,]
data_test_stand <- dat_stand[-idcs_train,]

# Perform KNN regression
pr_stand <- knn.reg(data_train_stand, data_test_stand, target_train, k = 3)

# Pull predicted response
f_hat_stand <- pr_stand$pred

# Calculate MSE for standardized case
loss_stand <- mean( (target_test - f_hat_stand)^2 )
```

Compare losses between no standardization and standardization

The loss for the no-standardization case is 0.01185, while the loss for the standardization case is 0.01658.

In other words, standardizing increases the loss by 39.85%!

Takeaway

Standardization is not always a required or even a desirable preprocessing step for implementing a nearest neighbors model.