MDTS4214_729_PROBLEMSET5

729_ARANYA PRADHAN

2026-02-26

1 Problem to demonstrate the utility of K nearest neighbour regression over least squares regression

```
library(FNN) # For Fast KNN
```

```
## Warning: package 'FNN' was built under R version 4.5.2
```

```
# Set seed for reproducibility
set.seed(123)

n <- 1000

# (i) Generate features
x1 <- rnorm(n, mean = 0, sd = 2)
x2 <- rpois(n, lambda = 1.5)

# (ii) Generate error term
epsilon <- rnorm(n, mean = 0, sd = 1)

# (iii) Generate target variable
y <- -2 + 1.4 * x1 - 2.6 * x2 + epsilon

# Combine into a data frame
df <- data.frame(x1 = x1, x2 = x2, y = y)

# Split the data
# Training: first 800 observations
train_data <- df[1:800, ]

# Testing: remaining 200 observations
test_data <- df[801:1000, ]

# Verify the split
cat("Training set size:", nrow(train_data), "\n")
```

```
## Training set size: 800
```

```
cat("Test set size:", nrow(test_data), "\n")
```

```
## Test set size: 200
```

1. Fit a multiple linear regression equation of y on x1 and x2. Calculate test MSE:

```r
# Fit the model using the first 800 observations
model <- lm(y ~ x1 + x2, data = train_data)

# View the estimated coefficients
summary(model)
```

```
##
## Call:
## lm(formula = y ~ x1 + x2, data = train_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.0727 -0.6573 -0.0125  0.6921  3.2412
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -2.07300    0.05382  -38.52   <2e-16 ***
## x1           1.38207    0.01767   78.21   <2e-16 ***
## x2          -2.55584    0.02768  -92.34   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.98 on 797 degrees of freedom
## Multiple R-squared:  0.9492, Adjusted R-squared:  0.9491
## F-statistic:  7445 on 2 and 797 DF,  p-value: < 2.2e-16
```

```r
# Predict y for the test set
predictions <- predict(model, newdata = test_data)

# Calculate the Mean Squared Error
test_mse <- mean((test_data$y - predictions)^2)

# Print the result
cat("The Test MSE is:", test_mse, "\n")
```

```
## The Test MSE is: 0.998901
```

2. Fit a KNN model with k = 1, 2, 5, 9, 15. Calculate test MSE for each choice of k.

```r
library(FNN)

# Ensure ks is defined
ks <- c(1, 2, 5, 9, 15)

for (k in ks) {
  # FIX: Use 'train_data' consistently instead of 'train'
  knn_result <- knn.reg(train = train_data[, 1:2],
                        test  = test_data[, 1:2],
                        y     = train_data$y,
                        k     = k)
```

```r
  # Calculate MSE using 'test_data'
  mse_knn <- mean((test_data$y - knn_result$pred)^2)

  cat("KNN (k =", k, ") Test MSE:", mse_knn, "\n")
}
```

```
## KNN (k = 1 ) Test MSE: 2.219793
## KNN (k = 2 ) Test MSE: 1.729587
## KNN (k = 5 ) Test MSE: 1.303978
## KNN (k = 9 ) Test MSE: 1.205371
## KNN (k = 15 ) Test MSE: 1.235776
```

Suppose the data in Step (iii) is generateD AS:

```r
library(FNN)
set.seed(42)
n <- 1000

# (i) & (ii) Generate features and noise
x1 <- rnorm(n, 0, 2)
x2 <- rpois(n, 1.5)
epsilon <- rnorm(n, 0, 1)

# (iii) Complex non-linear target variable
# Note: Adding a small constant to the denominator to avoid division by zero
denominator <- -2 + 1.4*x1 - 2.6*x2 + 2.9*x1^2
y <- 1/(denominator) + 3.1*sin(x2) - 1.5*x1*x2^2 + epsilon

# Split data
train_data <- data.frame(x1, x2, y)[1:800, ]
test_data  <- data.frame(x1, x2, y)[801:1000, ]

# --- (1) Multiple Linear Regression ---
model_lm <- lm(y ~ x1 + x2, data = train_data)
pred_lm  <- predict(model_lm, test_data)
mse_lm   <- mean((test_data$y - pred_lm)^2)

# --- (2) KNN Models ---
ks <- c(1, 2, 5, 9, 15)
knn_results <- data.frame(k = ks, mse = NA)

for (i in 1:length(ks)) {
  knn_fit <- knn.reg(train = train_data[, 1:2],
                     test  = test_data[, 1:2],
                     y     = train_data$y,
                     k     = ks[i])
  knn_results$mse[i] <- mean((test_data$y - knn_results$pred[i])^2) # Note: p
red is in knn_fit
  # Corrected line for the loop:
  knn_results$mse[i] <- mean((test_data$y - knn_fit$pred)^2)
```

```
}

# Output Results
cat("Linear Regression Test MSE:", mse_lm, "\n\n")

## Linear Regression Test MSE: 428.9766

print(knn_results)

##    k      mse
## 1  1 230.9648
## 2  2 195.3812
## 3  5 237.8814
## 4  9 249.0432
## 5 15 284.5141
```

Comparison and Comments: 1. Linear Regression Performance In this scenario, the Linear Regression Test MSE will likely be very high.Bias: The model is “underfitting.” It is trying to fit a flat plane to a surface that has curves (from the x_1^2 and $\sin(x_2)$ terms) and complex twists (from the $x_1 x_2^2$ interaction).Misspecification: Since the model only looks for x_1 and x_2 linearly, it completely misses the functional form of the data.2. KNN PerformanceKNN will likely outperform Linear Regression here, provided k is chosen well.Flexibility: KNN doesn’t care about the formula. it just looks at what happened to similar x_1 and x_2 values nearby.Optimal k: You will likely find that an intermediate k (like 5 or 9) performs best.If $k = 1$, the model is too sensitive to the noise (ϵ).If $k = 15$, the model might start “smoothing out” the sharp curves of your new complex formula too much.