CMSC/LING/STAT 208: Machine Learning

Abhishek Chakraborty [Much of the content in these slides have been adapted from *An Introduction to Statistical Learning: with Applications in R*, James et al.]

Multiple Linear Regression

Response Y and predictor variables $X_1, X_2, \ldots, X_p$. We assume

$$Y = f(\mathbf{X}) + \epsilon = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_p X_p + \epsilon$$

β_j quantifies the association between the j^{th} predictor and the response.

Multiple Linear Regression: Estimating Parameters

We use training data to find $b_0, b_1, \ldots, b_p$ such that

$$\hat{y} = b_0 + b_1\ x_1 + \ldots + b_p\ x_p$$

Observed response: y_i for $i = 1, \ldots, n$

Predicted response: $\hat{y}_i$ for $i = 1, \ldots, n$

Residual: $e_i = \hat{y}_i - y_i$ for $i = 1, \ldots, n$

Mean Squared Error (MSE): $MSE = \dfrac{e_1^2 + e_2^2 + \ldots + e_n^2}{n}$ also known as the **loss/cost function**

Problem: Find $b_0, b_1, \ldots, b_p$ which minimizes MSE

We can use **Gradient Descent** to minimize the MSE with respect to $b_0, b_1, \ldots, b_p$.

Multiple Linear Regression

House Prices dataset

- `size` is in square feet
- `num_bedrooms` is a count
- `price` is in $1,000

```
house_prices <- readRDS("house_prices.rds")   # load dataset

head(house_prices, 6)   # first 6 observations
```

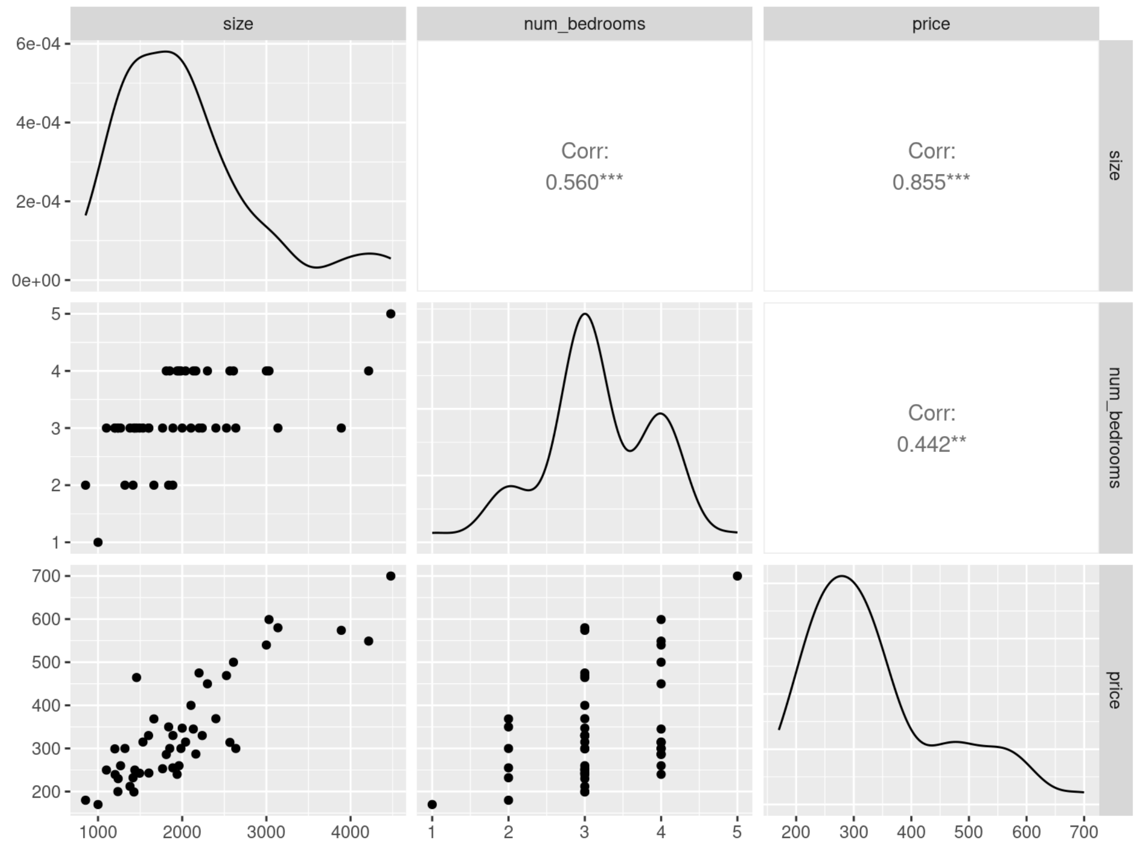
```
##   size num_bedrooms price
## 1 2104            3 399.9
## 2 1600            3 329.9
## 3 2400            3 369.0
## 4 1416            2 232.0
## 5 3000            4 539.9
## 6 1985            4 299.9
```

Multiple Linear Regression

Some Exploratory Data Analysis (EDA)

```r
library(GGally)

ggpairs(data = house_prices)   # correlation plot
```



Multiple Linear Regression in R

House Prices dataset

```r
mlr_model <- lm(price ~ size + num_bedrooms, data = house_prices)   # fit the model

summary(mlr_model)   # produce result summaries of the model
```

```
##
## Call:
## lm(formula = price ~ size + num_bedrooms, data = house_prices)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -130.58  -43.64  -10.83   43.70  198.15
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   89.5978    41.7674   2.145   0.0375 *
## size           0.1392     0.0148   9.409 4.22e-12 ***
## num_bedrooms  -8.7379    15.4507  -0.566   0.5746
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 66.07 on 44 degrees of freedom
## Multiple R-squared:  0.7329, Adjusted R-squared:  0.7208
## F-statistic: 60.38 on 2 and 44 DF,  p-value: 2.428e-13
```

Multiple Linear Regression: Interpreting Parameters

House Prices dataset

- $b_1 = 0.1392$: With `num_bedrooms` remaining fixed, an additional 1 square foot of `size` leads to an increase in `price` by approximately $139.

Multiple Linear Regression: Prediction

House Prices dataset

Prediction of `price` when `size` is 2000 square feet for a house with 3 bedrooms.

```
predict(mlr_model, newdata = data.frame(size = 2000, num_bedrooms = 3))   # obtain prediction
```

```
##        1
## 341.8053
```

Linear Regression: Comparing Models

With the **House Prices** dataset, we create three models with `price` as the response:

- **fit1**: a linear regression model with `num_bedrooms` as the only predictor
- **fit2**: a linear regression model with `size` as the only predictor
- **mlr_model** (already created in the previous slides): a multiple regression model with `size` and `num_bedrooms` as predictors

Linear Regression: Comparing Models

- Mean Squared Error (MSE)

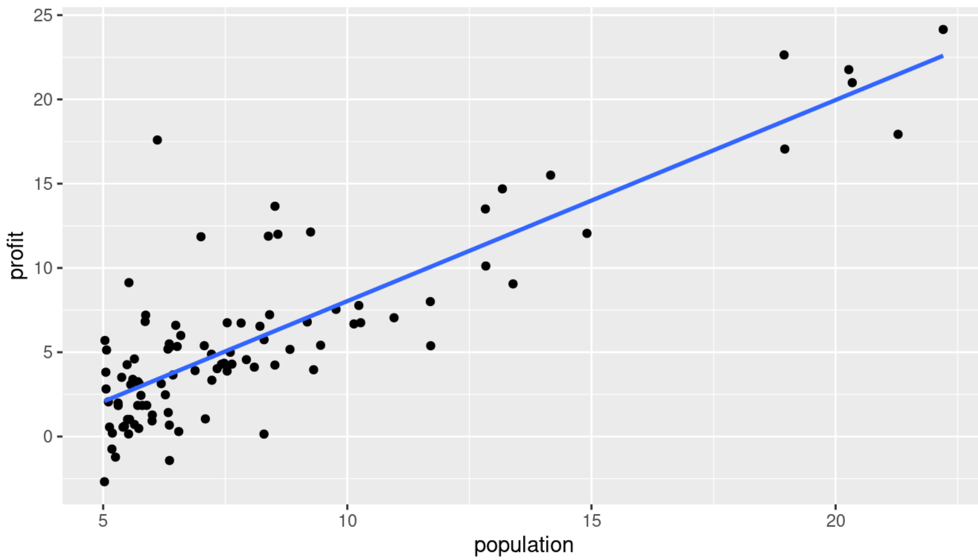
$$MSE = \sum_{i=1}^{n} e_i^2$$

- Residual Standard Error (RSE)

$$RSE = \sqrt{\frac{MSE}{n-p-1}}$$

Regression: Conditional Averaging

Restaurant Outlets Profit dataset



What is a good value of $\hat{f}(x)$ (expected profit), say at $x = 6$?

A possible choice is the **average of the observed responses** at $x = 6$. But we may not observe responses for certain x values.

K-Nearest Neighbors Regression

- Non-parametric approach
- Given a value for K and a test data point x_0,

$$\hat{f}(x_0) = \frac{1}{K} \sum_{x_i \in \mathcal{N}_0} y_i = \text{Average } (y_i \text{ for all } i: \; x_i \in \mathcal{N}_0)$$

where $\mathcal{N}_0$ is known as the **neighborhood** of x_0.

- The method is based on the concept of closeness of x_i's from x_0 for inclusion in the neighborhood $\mathcal{N}_0$. Usually, the **Euclidean distance** is used as a measure of closeness. The Euclidean distance between two p-dimensional vectors $\mathbf{a} = (a_1, a_2, \ldots, a_p)$ and $\mathbf{b} = (b_1, b_2, \ldots, b_p)$ is

$$||\mathbf{a} - \mathbf{b}||_2 = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2 + \ldots + (a_p - b_p)^2}$$

K-Nearest Neighbors Regression (single predictor): Fit

Restaurant Outlets Profit dataset

```
library(caret)   # load the caret package
```

- 1-NN regression

```
knnfit1 <- knnreg(profit ~ population, data = outlets, k = 1)   # 1-nn regression
```

```
predict(knnfit1, newdata = data.frame(population = 6))  # 1-nn prediction
```

```
## [1] 0.92695
```

- 5-NN regression

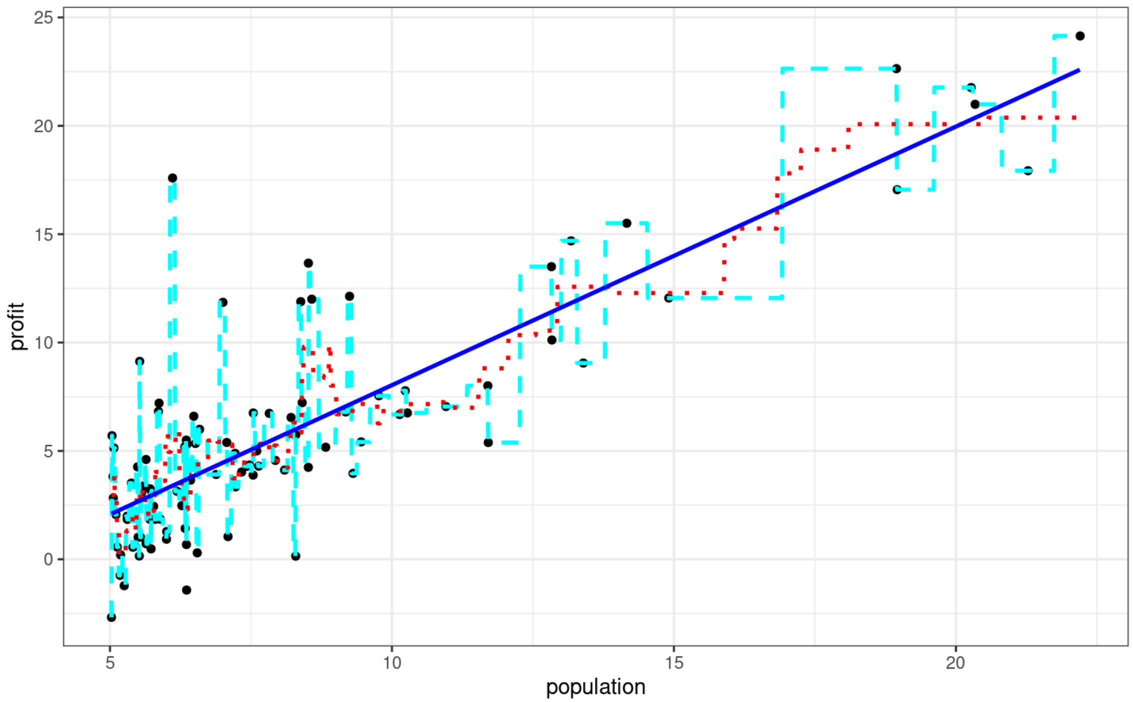
```
knnfit5 <- knnreg(profit ~ population, data = outlets, k = 5)   # 5-nn regression
```

```
predict(knnfit5, newdata = data.frame(population = 6))  # 5-nn prediction
```

```
## [1] 5.76995
```

Regression Methods: Comparison

Restaurant Outlets Profit dataset



dashed cyan: 1-nn fit, dotted red: 5-nn fit, blue: linear regression fit

Question!!!

As k in KNN regression increases,

- the flexibility of the fit ________________ (increases/decreases)
- the bias of the fit ________________ (increases/decreases)
- the variance of the fit ________________ (increases/decreases)

K-Nearest Neighbors Regression (multiple predictors)

It is important to **scale (subtract mean and divide by standard deviation)** the predictors when considering KNN regression so that the Euclidean distance is not dominated by a few of them with large values.

House Prices dataset

```r
# scale predictors
house_prices_scaled <- data.frame(size_scaled = scale(house_prices$size),
                                  num_bedrooms_scaled = scale(house_prices$num_bedrooms),
                                  price = house_prices$price)

head(house_prices_scaled)   # first six observations
```

```
##   size_scaled num_bedrooms_scaled price
## 1  0.13000987          -0.2236752 399.9
## 2 -0.50418984          -0.2236752 329.9
## 3  0.50247636          -0.2236752 369.0
## 4 -0.73572306          -1.5377669 232.0
## 5  1.25747602           1.0904165 539.9
## 6 -0.01973173           1.0904165 299.9
```

K-Nearest Neighbors Regression (multiple predictors)

House Prices dataset

```
library(caret)   # load library
```

```
knnfit10 <- knnreg(price ~ size_scaled + num_bedrooms_scaled, data = house_prices_scaled, k = 10)   # 10-nn regression
```

It is also important to apply scaling to test data points before prediction. Suppose, you want predictions for `size` = 2000 square feet, and `num_bedrooms` = 3, then

```
# obtain 10-nn prediction

predict(knnfit10, newdata = data.frame(size_scaled = (2000 - mean(house_prices$size, na.rm = TRUE))/sd(house_prices$size, na.rm = TRUE),
                                     num_bedrooms_scaled = (3 - mean(house_prices$num_bedrooms))/sd(house_prices$num_bedrooms)))
```

```
## [1] 339.14
```

Linear Regression vs K-Nearest Neighbors

- Linear regression is a parametric approach (with restrictive assumptions), KNN is non-parametric.
- Linear regression works for regression problems (Y numerical), KNN can be used for both regression and classification (Y qualitative).
- Linear regression is interpretable, KNN is not.
- Linear regression can accommodate qualitative predictors and can be extended to include interaction terms as well. Using Euclidean distance with KNN does not allow for qualitative predictors.
- In terms of prediction, KNN can be pretty good for small p, that is, $p \leq 4$ and large n. Performance of KNN deteriorates as p increases - curse of dimensionality.

Classification Problems

- Response Y is qualitative (categorical).
- The objective is to build a classifier $\hat{Y} = \hat{C}(\mathbf{X})$ that assigns a class label to a future unlabeled (unseen) observation and understand the relationship between the predictors and response.
- There can be two types of predictions based on the research problem.
    - Class probabilities
    - Class labels

Classification Problems: Example

Default dataset

```r
library(ISLR2)   # load library
data("Default")   # load dataset
```

```r
head(Default)   # print first six observations
```

```
##   default student   balance    income
## 1      No      No  729.5265 44361.625
## 2      No     Yes  817.1804 12106.135
## 3      No      No 1073.5492 31767.139
## 4      No      No  529.2506 35704.494
## 5      No      No  785.6559 38463.496
## 6      No     Yes  919.5885  7491.559
```

```r
table(Default$default)   # class frequencies
```

```
## 
##   No  Yes 
## 9667  333
```

We will consider `default` as the response variable.

Classification Problems: Example

For some algorithms, we might need to convert the categorical response to numeric (0/1) values.

Default dataset

```
Default$default_id <- ifelse(Default$default == "Yes", 1, 0)   # create 0/1 variable

head(Default, 10)   # print first ten observations
```

```
##    default student   balance    income default_id
## 1       No      No  729.5265 44361.625          0
## 2       No     Yes  817.1804 12106.135          0
## 3       No      No 1073.5492 31767.139          0
## 4       No      No  529.2506 35704.494          0
## 5       No      No  785.6559 38463.496          0
## 6       No     Yes  919.5885  7491.559          0
## 7       No      No  825.5133 24905.227          0
## 8       No     Yes  808.6675 17600.451          0
## 9       No      No 1161.0579 37468.529          0
## 10      No      No    0.0000 29275.268          0
```

K-Nearest Neighbors Classifier

Given a value for K and a test data point x_0,

$$P(Y = j | X = x_0) = \frac{1}{K} \sum_{x_i \in \mathcal{N}_0} I(y_i = j)$$

where $\mathcal{N}_0$ is known as the **neighborhood** of x_0.

For classification problems, the predictions are obtained in terms of **majority vote** (unlike in regression where predictions are obtained by averaging).

K-Nearest Neighbors Classifier: Build Model

Default dataset

response (Y): `default` and predictor (X): `balance`

```
library(caret)   # load package 'caret'

knnfit <- knn3(default ~ balance, data = Default, k = 10)   # fit 10-nn model
```

K-Nearest Neighbors Classifier: Predictions

Default dataset

- One can directly obtain the class label predictions as below.

```
knn_class_preds_1 <- predict(knnfit, newdata = Default, type = "class")   # obtain default class label predictions
```

- Otherwise, one can first obtain predictions in terms of probabilities and then convert them into class label predictions based on a threshold.

```
knn_prob_preds <- predict(knnfit, newdata = Default, type = "prob")   # obtain predictions as probabilities
```

```
threshold <- 0.5   # set threshold

knn_class_preds_2 <- factor(ifelse(knn_prob_preds[,2] > threshold, "Yes", "No"))   # obtain predictions as class labels
```

K-Nearest Neighbors Classifier: Performance

Default dataset

```r
# create confusion matrix

# use the following code only when all predictions are from the same class
# levels(knn_class_preds_1) = c("No", "Yes")

confusionMatrix(data = knn_class_preds_1, reference = Default$default, positive = "Yes")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   No  Yes
##        No  9619  216
##        Yes   48  117
##
##                Accuracy : 0.9736
##                  95% CI : (0.9703, 0.9767)
##     No Information Rate : 0.9667
##     P-Value [Acc > NIR] : 3.947e-05
##
##                   Kappa : 0.4579
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.3514
##             Specificity : 0.9950
##          Pos Pred Value : 0.7091
##          Neg Pred Value : 0.9780
##              Prevalence : 0.0333
##          Detection Rate : 0.0117
##    Detection Prevalence : 0.0165
##       Balanced Accuracy : 0.6732
##
##        'Positive' Class : Yes
##
```