Modeling Basics

## What is Statistical Learning?

**Statistical Learning** - Statistical learning refers to a vast set of tools for understanding data. (James et al., 2021, p. 1)

 We want to use data to solve a problem. This may be done through inference or predictive modeling. Predictive modeling is synonymous with machine learning.  
   
 How we program in predictive modeling/machine learning differs from the traditional programming paradigm. Traditionally, programmers have taken data and rules and tried to combine them to get an answer. In machine learning, we try to take data and a set of answers to obtain the rules. These rules can then be used to predict for a future observation.  
 

Is there a difference between *machine* learning and *statistical* learning?

 Answers will vary, depending on who you talk to. However, statistical learning tends to involve more understanding of our data and the inferences we can make. This includes when we do predictive modeling.  
   
   
 Goals of statistical learning:

* Inference

  This includes classical ideas like hypothesis testing, confidence intervals, and prediction intervals. 

* Prediction/Classification

 This situation applies when we have a ‘response variable’. That is, a variable that characterizes performance. This response variable may be categorical or numeric. A categorical response is often binary but not always. When we have a categorical response, we call it a classification task. When we have a numeric response, we call it a regression task.  

* Pattern Finding

This situation is for when we don’t have a response variable. Instead we just want to understand relationships within the data.

### Types of statistical learning:

#### Supervised Learning – situation where we have a response variable

Consider a [data set](https://archive.ics.uci.edu/dataset/560/seoul+bike+sharing+demand) on bike sharing (UCI, 2020) [available for download](https://www4.stat.ncsu.edu/online/datasets/SeoulBikeData.csv) in .csv format, which is the raw data format.

library(tidyverse) #a package that we would need to install

#creating a tibble, which is a special data frame, called bike\_share  
bike\_share <- read\_csv("https://www4.stat.ncsu.edu/online/datasets/SeoulBikeData.csv",  
 local = locale(encoding = "latin1"))  
bike\_share[1:5, ] |>   
 select(`Rented Bike Count`, Date, `Temperature(°C)`) |>   
 kable()

| Rented Bike Count | Date | Temperature(°C) |
| --- | --- | --- |
| 254 | 01/12/2017 | -5.2 |
| 204 | 01/12/2017 | -5.5 |
| 173 | 01/12/2017 | -6.0 |
| 107 | 01/12/2017 | -6.2 |
| 78 | 01/12/2017 | -6.0 |

Let’s introduce some common notation and ideas for applying a basic statistical learning method to predict the Rented Bike Count.

* = Sample size = 8760
* - usually used to reference a particular *observation* (or row). Here, i = 1, 2, …, 8760.
* = **response variable** = Rented Bike Count

We could write the responses from the data as a column vector. In terms of the R object bike\_share, this is one of the columns there.

* - index usually used to denote a particular *predictor or explanatory variable*. Really any column that isn’t the response variable. Here, j = 1, 2, …, 13.
* = **predictor variable** = Temperature(°C) (for instance)

 Again, we can consider a particular predictor variable as a column vector that represents one of the columns of our bike\_share tibble.  will represent observation i of variable j.

* = Number of predictors = 13
* = set of predictor variables (sometimes we’ll include an intercept column if this is used as a *design matrix*)
* Considering together is essentially what our data frame in R (or python) looks like

We then try to model our response variable as a some function of the predictors!

   

is some random error term that, on average, takes on 0. We’ll use the notation to represent this population level average. Here is our assumption.

is some function that defines our relationship between Y and X. We want to learn about this f function by assuming different structures for it.  
   
    
   
   
   
   
   
   
   
   
 

##### Simple Linear Regression (SLR)

Perhaps the most simplistic model is the simple linear regression (SLR) model.

Where X is a predictor, Y is our response, and are our regression coefficients. These betas are unknown and we need to use the data to estimate or fit them. is assumed to have mean 0.

We may also choose to assume that the individual terms are iid – or independent and identically distributed – Noraml with mean 0 and variance .

bike\_share |>  
 ggplot(aes(x = `Temperature(°C)`, y = `Rented Bike Count`)) +  
 geom\_point(size = 0.5) +  
 geom\_smooth(method = "lm")

|  |
| --- |
| A scatterplot between temperature on the x-axis and rented bike count on the y-axis. The plots shows a general upward trend with more spread in the rented bike count as the temperature increases. The overlayed SLR line has a positive slope and shows a strong linear relationship. The line does not fit the data well for temperatures below -10 degrees as it predicts a negative rented bike count in that region.  Scatterplot with fitted SLR model overlayed |

* If we are interested in inference, we likely want to understand which predictors are important for understanding our response and what that relationship looks like (or can be approximated by at least).
  + This requires many assumptions we’d need to check but can easily be fit in your favorite statistical software.

SLR\_fit <- lm(`Rented Bike Count` ~ `Temperature(°C)`, data = bike\_share)  
summary(SLR\_fit)$coefficients |>  
 knitr::kable()

|  | Estimate | Std. Error | t value | Pr(>|t|) |
| --- | --- | --- | --- | --- |
| (Intercept) | 329.9525 | 8.5410613 | 38.63132 | 0 |
| Temperature(°C) | 29.0811 | 0.4861734 | 59.81631 | 0 |

The Estimate values from the output are the fitted or estimated regression coefficients. and . The fitted regression equation is then .

The p-value corresponding to the Temperature slope coefficient is very close to 0. This p-value is used for testing . If the null hypothesis is true then we have no relationship between x and y.

* If we are interested in prediction, we want to understand how well our model predicts the response.
  + If the response is quantitative, we have a regression task
  + If the response is categorical, we have a classification task

In the code below, we use the tibble function to create a tibble for predicting the bike count response for a temperature of 0 and for a temperature of 30.

predict(SLR\_fit,   
 newdata = tibble(`Temperature(°C)` = c(0, 30)),   
 se.fit = TRUE,   
 interval = "prediction") |>  
 knitr::kable()

| fit | lwr | upr |
| --- | --- | --- |
| 329.9525 | -735.5635 | 1395.469 |
| 1202.3855 | 136.8153 | 2267.956 |

| Se.fit |
| --- |
| 8.541061 |
| 10.147604 |

Df = 8758

Residual.scale = 543.4984

Either way, we must pick a method for modeling the response and **fit** or **estimate** that model! Some models are generally better for inference and some generally are better for prediction. We have to understand our goals and then consider what makes sense for our problem.

* To **fit** the model we can usually think about optimizing some **loss function**.

For SLR, we usually try to minimize the sum of squared errors over all possible values. An error is also called a residual and is defined as an observed value minus a predicted value. Notation-wise, this is where . We want to find the estimates for the beta coefficients that minimizes the sum of the squares of these residuals over all of the observations in the data set.

The loss function minimization is then given by

##### Parametric vs Non-Parametric Models

The SLR model is an example of a **parametric model**. A parametric model generally imposes some kind of structure on the relationship between and an .

bike\_share |>  
 ggplot(aes(x = `Temperature(°C)`, y = `Rented Bike Count`)) +  
 geom\_point(size = 0.5) +  
 geom\_smooth(method = "lm", formula = y ~ poly(x, 4))

|  |
| --- |
| A scatterplot between temperature on the x-axis and rented bike count on the y-axis. The plots shows a general upward trend with more spread in the rented bike count as the temperature increases. The overlayed fourth degree fitted line follows the bulk of the data. For temperatures at the lower range (-20 degrees) the line starts off near zero, slowly increasing until around 10 degrees where it increases more rapidly. Around 25 degrees the line begins to decrease with a sharp downturn near 35 degrees.  Scatterplot with fitted quartic polynomial regression model overlayed |

 The equation assumed for this relationship between Y and X is   
   
   
   
 

There is a tradeoff between simplicity of the model (easier interpretation) and complexity (a larger number of parameters – the betas here). With a model that is too complex we may **overfit** to the data we fit (or **train**) our model on.

A **non-parametric model** seeks to find an estimate of that is ‘close’ without being ‘too wiggly’ or ‘too rough’. These models typically require more observations to be accurate and can also suffer from overfitting.

###### k Nearest Neighbors (kNN)

An example of a simplistic non-parametric model is the k Nearest Neighbors (kNN) model. In this model, for any point of interest, we determine a way to find the closest observations in the data used to fit the model. When doing a regression task, we then usually use the mean of the response values for those observations as our prediction.

bike\_share[1:6, ] |>  
 select(`Rented Bike Count`, `Temperature(°C)`) |>  
 arrange(`Temperature(°C)`) |>  
 kable()

| Rented Bike Count | Temperature(°C) |
| --- | --- |
| 100 | -6.4 |
| 107 | -6.2 |
| 173 | -6.0 |
| 78 | -6.0 |
| 204 | -5.5 |
| 254 | -5.2 |

If our k is three and we want to predict for an temperature of -6.1 degrees, we would find the three closest observations (in terms of temperature). Here, those would be -6.2, -6, and -6. We then take the three corresponding bike count values for these observations and add them. Here, that gives 107 + 173 + 78, all divided by three.

* Again, our statistical software takes care of computations for us…

library(kknn)  
knn\_fit <- kknn(`Rented Bike Count` ~ `Temperature(°C)`, train = bike\_share, test = bike\_share, k = 200)  
knn\_estimates <- mutate(bike\_share, knn\_est = fitted(knn\_fit)) |>  
 arrange(knn\_est)  
knn\_estimates |>  
 ggplot(aes(x = `Temperature(°C)`, y = `Rented Bike Count`)) +  
 geom\_point(size = 0.5) +  
 geom\_line(aes(y = knn\_est), color = "blue", linewidth = 2)

|  |
| --- |
| A scatterplot between temperature on the x-axis and rented bike count on the y-axis. The plots shows a general upward trend with more spread in the rented bike count as the temperature increases. The overlayed kNN model follows the middle of the data closely, wiggling quite a bit for larger values of temperature - indicating the model may be a bit too flexible there.  Scatterplot with fitted kNN model with k = 200 overlayed |

Of course the *optimal* value of isn’t clear! This is referred to as a **tuning parameter** (or **hyperparameter**). We’ll discuss how to use our data and a model metric to choose the value of a tuning parameter shortly.

|  |
| --- |
| Figure shows flexibility on the x-axis (low to high) and interpretability on the y-axis (low to high). In a roughly downward line from top left to bottom right we see the following methods: Subset selection, LASSO, Least Squares, Generalized Additive Models, Trees, Bagging, Boosting, Support Vector Machines, and Deep Learning.  Figure taken from James et al. (2021) p. 25. |

##### Curse of Dimensionality

You might think that we should just always use a non-parametric model since they are more flexible and can adjust to fit the data more easily.

* First, parametric models will often perform pretty well and be much more interpretable while also lending themselves to inference more easily.
* Second, we have to concern ourselves with the **curse of dimensionality** (Bellman, 1961)
  + This curse takes many forms… but the main idea is that as we increase our number of dimensions in the predictor space, we need more and more observations to have the same idea about how the response may act at that combination of the predictors

As an example, consider having one quantitative predictor that takes on values between 0 and 1. Suppose we have 20 observations of this predictor.

set.seed(11)  
first\_variable <- runif(20)  
sort(first\_variable)

[1] 0.0005183129 0.0140479084 0.0646897766 0.0864958912 0.1232162013  
 [6] 0.1576602489 0.1751129227 0.2772497942 0.2899750092 0.3306109945  
[11] 0.4407502718 0.4801340534 0.4817655359 0.5106083730 0.5736857215  
[16] 0.7339874927 0.8510418669 0.8806991728 0.9071829736 0.9548492255

You might say these 20 points cover the range from 0 to 1 pretty well. We’d be happy finding neighbors that are ‘close’ that our prediction could ‘borrow’ strength from.

However, consider having a second predictor but still only 20 observations.

second\_variable <- runif(20)  
plot(x = first\_variable, y = second\_variable,   
 xlim = c(0,1),  
 ylim = c(0,1))

|  |
| --- |
| Scatterplot showing 20 randomly created pairs of points. Both variables taking on the range 0 to 1.", fig.alt = "A scatterplot is shown of 20 randomly generated values between 0 and 1 and 20 other randomly generated values between 0 and 1. No real pattern exists but large gaps exist between some points indicating that there aren't as many close values in two dimensions as compared to one dimension.  Scatterplot showing 20 randomly created pairs of points. Both variables taking on the range 0 to 1. |

##### Considerations when Prediction is our Goal

The biggest idea to understand when prediction is our goal is that we want our model to be good at predicting for observations it **was not trained on**!

First, we need to understand what it means to predict well.

* We define a **model metric** that determines our model performance.
  + For the regression task, common metrics include mean square error or MSE and root mean square error or RMSE. Suppose we have a vector of responses and a matrix of corresponding predictor variables that the model was not trained on. MSE would be equal to the sum over all observations of the actual response, minus the predicted value, , squared. This summation is then divided by the number of observations to obtain the MSE. RMSE is simply the square root of this quantity.
  + For the classification task, common metrics include negative log loss. Given a set of responses taking on 0 and 1, values, and corresponding estimated probabilities of taking on 1 from the model, value, the negative log loss is given by

   
 

Second, if we train (or fit) our model on the entire data set, we don’t have any data to test on!

* One way to get around this is to do data splitting. That is, we split our data into a training and test set.
* We may choose to use simple random sampling to randomly allocate 80% of our observations to a training set and 20% of our observations to a test set. Other common proportions of training set to test set values is 70% to 30%.
* If we are concerned about making sure certain subgroups of the observations are proportionally represented in the training and test sets, we may instead use a stratified random sampling technique to assign the observations.

We mentioned that many models require us to determine a **tuning parameter** value. We don’t want to use our test set in this process! There are a few major options to turn to:

* Split the training data further into an **internal training** set and a **validation set** (sometimes called a **holdout set**).
* Use cross-validation on the training set to determine an appropriate tuning parameter value

|  |
| --- |
| Training/Test set idea with five fold cross-validation on the training set visualization", fig.alt = "The full data set is depicted being split into two parts: a training and test set. The training set is shown to then be split into five distinct folds (or sets). Four of the folds are used for training and the fifth used for testing. The fold that acts as a test set rotates until all folds are used as the test set once.  Training/Test set idea with five fold cross-validation on the training set visualization |

* Using bootstrap resampling. One way to do this is to random sample with replacement from our observations. For instance, consider having five observations, labeled 1, 2, 3, 4, and 5. We would take a random sample of size five with replacement from these observations. One sample might give us observation 2, 3, 2, 4, and 2. We would train our model on these observations and then test on the observations that were left out. In this case, observations 1 and 5.

#### Models are Approximations

Regardless of the model we choose, how we choose to fit and/or tune it, we never really assume the model is correct. We hope that the model is useful!

All models are wrong; some models are useful. (Box, Hunter, and Hunter, 2005)

#### Unsupervised Learning – no response, our goal is pattern finding

Consider a data set on Graft-versus-Host-Disease (GvHD) from the mclust package in R (Srucca, et al., 2023) (see ?GvHD after installing and loading the mclust package in R). Information about the data:

Two samples of this flow cytometry data, one from a patient with the GvHD, and the other from a control patient. The GvHD positive and control samples consist of 9083 and 6809 observations, respectively. Both samples include four biomarker variables, namely, CD4, CD8b, CD3, and CD8. The objective of the analysis is to identify CD3+ CD4+ CD8b+ cell sub-populations present in the GvHD positive sample.

Let’s just consider the patient with GvHD.

#install.packages(mclust) #install on your machine if you don't have this package  
library(mclust)  
library(dplyr)  
GvHD.pos[1:5, ] |>  
 kable()

| CD4 | CD8b | CD3 | CD8 |
| --- | --- | --- | --- |
| 308 | 220 | 157 | 339 |
| 319 | 271 | 223 | 350 |
| 215 | 148 | 119 | 221 |
| 104 | 49 | 284 | 178 |
| 262 | 167 | 144 | 156 |

Here we don’t have a (response) variable. This makes the problem inherently more difficult!

|  |
| --- |
| Exploratory graphs of the four variables under consideration.", fig.alt = "A four by four matrix of graphs is shown. Along the rows and column the variables are shown in order from CD4, CD8b, CD3, to CD8. The graphs along the diagonal show density plots for each variable. We see that each variable is roughly bell-shaped and right skewed. The CD4 variable has a small peak prior to its main peak and the CD8 variable has a small peak in the upper range. The lower off diagonal plots show scatterplots between the variables. There is generally a positive upward relationship between all of the variables but definite groups of points seem to exist. The upper off diagonal plots show contours associated with those scatterplots. These further highlight that, for some pairs of variables, there are separate groupings of points that seem to exist.  Exploratory graphs of the four variables under consideration. |

* May just be to group similar observations together (clustering)
  + If we can group them, can the groups be put into some kind of framework for interpretation? (Requires subject matter expertise!)

clusters <- Mclust(GvHD.pos, G = 3)  
GvHD\_cluster <- GvHD.pos |>  
 mutate(cluster = as.factor(clusters$classification))  
GvHD\_cluster[1:5, ] |>  
 kable()

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| | CD4 | CD8b | CD3 | CD8 | cluster | | --- | --- | --- | --- | --- | | 308 | 220 | 157 | 339 | 1 | | 319 | 271 | 223 | 350 | 2 | | 215 | 148 | 119 | 221 | 1 | | 104 | 49 | 284 | 178 | 3 | | 262 | 167 | 144 | 156 | 1 |   Exploratory graphs of the four variables under consideration with cluster groupings included.", fig.alt = "A four by four matrix of graphs is shown similar to the previous figure. However, this figure includes a coloring according to the cluster associated with each observation (1, 2, or 3). With this, the groupings of points noted earlier are now clearly placed into different clusters indicating an ability for a subject matter expert to interpret the results.  Exploratory graphs of the four variables under consideration with cluster groupings included. |

* May want to do dimension reduction!
  + Perhaps to find a better way to visualize the data. This is often done using principal components analysis, discussed more later in the course.
  + To deal with collinearity amongst the variables.

#### Semi-supervised Learning

In some data sets we have only part of our data set with a response variable (or label) associated. In this case, we often use the observations with responses to some how predict the missing responses. However, analysis of this type of data will depend on your goals. We won’t look into this situation in our course!

### Concept Map of Statistical Learning

Ameen, for this concept map, I would listen to the corresponding video. I tried to be very descriptive in that video when connecting the ideas. Listening to that video would be a much better explanation than I could give writing it out. Please let me know if you have any questions about it!

### References

* James, G., Witten, D., Hastie, T., and Tibshirani, R., *An Introduction to Statistical Learning*, second edition, 2021.
* Seoul Bike Sharing Demand [Dataset]. (2020). UCI Machine Learning Repository. https://doi.org/10.24432/C5F62R.
* Scrucca L, Fraley C, Murphy TB, Raftery AE (2023). *Model-Based Clustering, Classification, and Density Estimation Using mclust in R*. Chapman and Hall/CRC. ISBN 978-1032234953, doi:10.1201/9781003277965 <https://doi.org/10.1201/9781003277965>, <https://mclust-org.github.io/book/>.
* Bellman, Richard Ernest (1961). Adaptive control processes: a guided tour. Princeton University Press. ISBN 9780691079011
* Hastie, T., Tibshirani, R., and Friedman, J., *The Elements of Statistical Learning*, second edition, 2009.
* Box, G.E.P., Hunter, W., and Hunter, S., *Statistics for Experimenters*, second edition, 2005.