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)



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



### Goals of statistical learning:

* Inference





* Prediction/Classification



* Pattern Finding

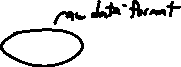


### Types of statistical learning:

#### Supervised Learning



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.



library(tidyverse)  
bike\_share <- read\_csv("https://www4.stat.ncsu.edu/online/datasets/SeoulBikeData.csv",  
 local = locale(encoding = "latin1"))  
print(bike\_share |> select(`Rented Bike Count`, everything()), n = 5)



# A tibble: 8,760 × 14  
 `Rented Bike Count` Date Hour `Temperature(°C)` `Humidity(%)`  
 <dbl> <chr> <dbl> <dbl> <dbl>  
1 254 01/12/2017 0 -5.2 37  
2 204 01/12/2017 1 -5.5 38  
3 173 01/12/2017 2 -6 39  
4 107 01/12/2017 3 -6.2 40  
5 78 01/12/2017 4 -6 36  
# ℹ 8,755 more rows  
# ℹ 9 more variables: `Wind speed (m/s)` <dbl>, `Visibility (10m)` <dbl>,  
# `Dew point temperature(°C)` <dbl>, `Solar Radiation (MJ/m2)` <dbl>,  
# `Rainfall(mm)` <dbl>, `Snowfall (cm)` <dbl>, Seasons <chr>, Holiday <chr>,  
# `Functioning Day` <chr>



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)
* = **response variable** = Rented Bike Count



* - index usually used to denote a particular *predictor* (or column that isn’t the response variable)
* = **predictor variable** = Temperature(°C) (for instance)



* = 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!



##### Simple Linear Regression (SLR)

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



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

|  |
| --- |
| 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).

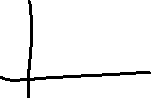
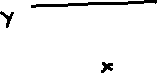


* + This requires many assumptions we’d need to check but can easily be fit in your favorite statistical software.

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



Estimate Std. Error t value Pr(>|t|)  
(Intercept) 329.9525 8.5410613 38.63132 1.288037e-301  
`Temperature(°C)` 29.0811 0.4861734 59.81631 0.000000e+00



* If we are interested in prediction, we want to understand how well our model predicts the response.
  + If the response is quantitative,



* + If the response is categorical,



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



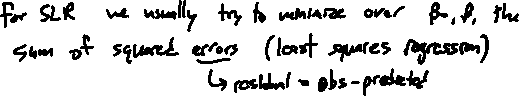
$fit  
 fit lwr upr  
1 329.9525 -735.5635 1395.469  
2 1202.3855 136.8153 2267.956  
  
$se.fit  
 1 2   
 8.541061 10.147603   
  
$df  
[1] 8758  
  
$residual.scale  
[1] 543.4984



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**.



##### 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))

|  |
| --- |
| Scatterplot with fitted quartic polynomial regression model overlayed |



There is a tradeoff between simplicity of the model (easier interpretation) and complexity (a larger number of parameters). 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)`)

# A tibble: 6 × 2  
 `Rented Bike Count` `Temperature(°C)`  
 <dbl> <dbl>  
1 100 -6.4  
2 107 -6.2  
3 173 -6   
4 78 -6   
5 204 -5.5  
6 254 -5.2



* 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)



|  |
| --- |
| 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 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. |

##### 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**!



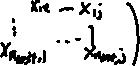
First, we need to understand what it means to predict well.



* We define a **model metric** that determines our model performance.



* + For the regression task, common metrics include



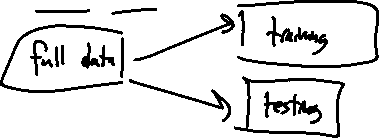
* + For the classification task, common metrics include



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.

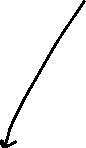
   
   
   
 



We mentioned that many models require us to determine a **tuning parameter** value. We don’t want to use our test set in this process! There are a few major options to turn to:



* 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 |

* Using bootstrap resampling



#### 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



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)  
print(as\_tibble(GvHD.pos), n = 5)



# A tibble: 9,083 × 4  
 CD4 CD8b CD3 CD8  
 <dbl> <dbl> <dbl> <dbl>  
1 308 220 157 339  
2 319 271 223 350  
3 215 148 119 221  
4 104 49 284 178  
5 262 167 144 156  
# ℹ 9,078 more rows



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

|  |
| --- |
| 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))  
print(as\_tibble(GvHD\_cluster), n = 5)

# A tibble: 9,083 × 5  
 CD4 CD8b CD3 CD8 cluster  
 <dbl> <dbl> <dbl> <dbl> <fct>   
1 308 220 157 339 1   
2 319 271 223 350 2   
3 215 148 119 221 1   
4 104 49 284 178 3   
5 262 167 144 156 1   
# ℹ 9,078 more rows



|  |
| --- |
| 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.



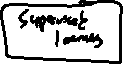
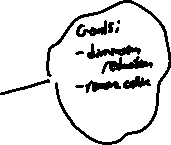
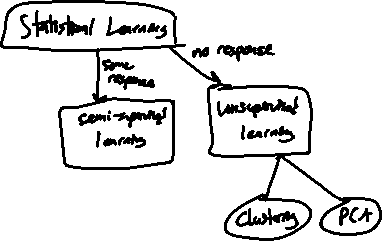
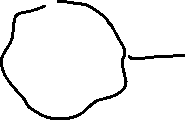
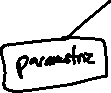
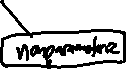
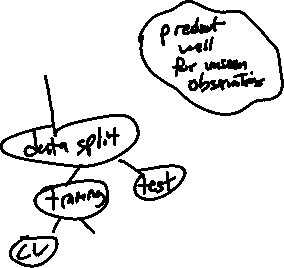
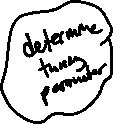
* + To deal with collinearity amongst the variables.



#### 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



### 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.