Introduction and Basic Concepts

Arnab Maity

NCSU Statistics ~ 5240 SAS Hall ~ amaity[at]ncsu.edu

Contents

What is statistical learning? 2

Benefits of statistical learning 3

Overview of model building 4

Statistical models 4

Parameters vs. Hyperparameters 5

Training/Testing the model 6

Supervised learning 8

Regression problems 8

Classification problems 12

Unsupervised learning 14

Clustering 14

Dimension reduction 16

Semi-supervised learning 18

Machine learning with R 19

Resources 19

Communication 19

What is statistical learning?

> *Statistical learning refers to a vast set of tools for understanding data.*
>
> — Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani, An Introduction to Statistical Learning, second edition, 2021, page 1.

Statistical learning[1] has become immensely useful in many fields of study and is being used to develop-cutting edge technologies. Simply speaking, statistical learning is used to discover patterns and relationships within data – these patterns then enable learning algorithms to make predictions/decisions based on new data.

[1] There is a never-ending debate about whether statistical learning and data-driven machine learning are synonymous or not or if one is a subset of the other etc. We will not go into that debate in this course.

There is one critical difference between classical programming and statistical learning. In classical programming, we have a data set we want to analyze, and we have a set of rules that dictates how we should process the data. The program takes these two inputs and provides an answer. For example, in Figure 1, we have handwriting samples (the images) as input data. We want to classify the images into their corresponding digits. Thus, we have to explicitly supply the rules to the program for the classification process (top figure). This setup is an example of classical programming, where the algorithm needs to know what features of the image to look at explicitly – it just acts as automation and does not "learn."

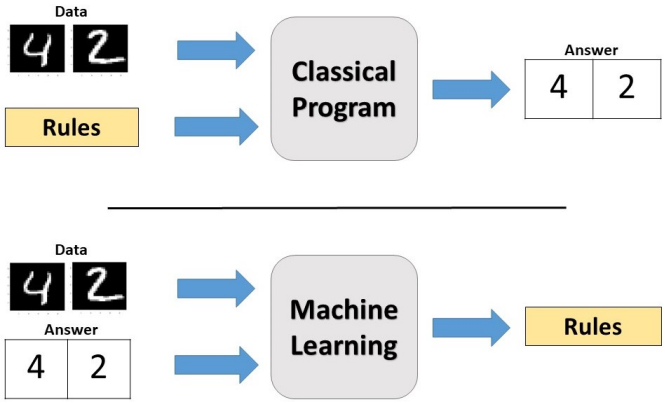

Figure 1: Comparison between classical programming versus maching learning algorithms.

In contrast, machine learning (bottom plot in Figure 1) solves the opposite problem. A machine learning algorithm takes the input data and the corresponding answers and determines the rule that mapped the data to the answers. In our example, a machine learning algorithm takes the images and the true digits and determines the rules to map the images to the correct digits. The algorithm "learns" the rules without them being pre-specified. In other words, a machine learning algorithm is "trained," not explicitly programmed.

> *Machine learning arises from this question: could a computer go beyond "whatever we know how to order it to perform" and learn on its own how*

> *to perform a specified task? Could a computer surprise us? Rather than programmers crafting data-processing rules by hand, could a computer automatically learn these rules by looking at data?*
>
> — F. Chollet and J. J. Allaire, Deep Learning with R, 2018, page 5.

Let us introduce some notations and ideas before proceeding further. We will use the following terminology and symbols throughout this course:

- Predictor variable: X – These are also called independent variables, covariates, features, attributes, predictors.
- Response variable: Y – Also known as dependent variable, target variable, response, outcome.

Typically, we index the units/items/individuals in our sample by i. If we have n units in our sample, we denote the response and the predictor for the i-th unit as Y_i and X_i, respectively, for $i = 1, \ldots, n$. In the image classification example shown in Figure 1, X_i represents the i-th input image, and Y_i represents the corresponding digit.

Labeled and Unlabeled Data

Labeled data: data where both X and Y are present. Here Y are regarded as "labels".

Unlabeled data: data where only X present.

Ideally, there is a function/map $f(\cdot)$ that, maps X_i to Y_i. Often there are also errors involved (e.g., in a regression model we might have $Y_i = f(X_i) + \epsilon_i$ with ϵ_i being random error). Estimation of $f(\cdot)$ is a goal of statistical learning. In Figure 1, $f(\cdot)$ represents the unknown rules. The estimated $f(\cdot)$ will be denoted as $\widehat{f}(\cdot)$.

Benefits of statistical learning

In statistical learning, the reason for estimation of $f(\cdot)$ are two-fold.

- *Prediction*: we can predict the response for a new value of X based on the estimated $f(\cdot)$. Specifically, for a new data point with predictor X, we can form the prediction $\widehat{Y} = \widehat{f}(X)$. In this case, prediction is the primary goal – we do not necessarily worry about interpretability of the model or the actual functional form of $f(\cdot)$.
- *Inference*: often we want to understand the relationship between a response Y and a set of predictors $X_1, \ldots, X_p$. In this case, prediction of Y is not the primary goal – we want a deeper understanding of the relationship itself. Questions such as "which predictors

are associated with Y?", "How is each of the predictors related to Y?", and "Are all predictors are of equal importance in relation to Y?" are of interest.

Depending on whether our priority lies in prediction, inference, or sometimes both, we will use different learning methods. Typically, simpler methods, such as linear models, are often more interpretable and lend themselves well to statistical inference. More complicated procedures such as deep learning, bagging, boosting are mainly used to build prediction models but are not very interpretable. Figure 2 shows a representation of the tradeoff between model flexibility and interpretability.[2]

[2] Figure and caption taken from Introduction to Statistical Learning, 2nd edition, by Hastie et al., 2021.

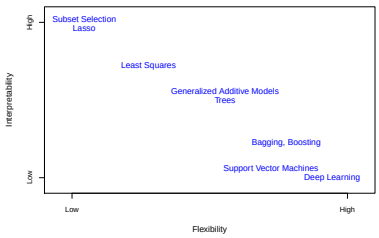

Figure 2: A representation of the tradeoff between flexibility and interpretability, using different statistical learning methods. In general, as the flexibility of a method increases, its interpretability decreases.

Overview of model building

Statistical models

A statistical learning method "learns" about the "rules" is to build statistical models for the observed data. The components of these models are then estimated using appropriate algorithms.

Statistical model

A mathematical specification/formulation that we use to describe the observed data.

Typically, a statistical model is specified by a collection of probability distributions on the observed data and specifying some assumptions on the functional from of $f(\cdot)$. For example, suppose we have response Y and a single predictor X. Consider the *polynomial regression model*,

$$Y_i = \beta_0 + X_i\beta_1 + \cdots + X_i^d\beta_d + \epsilon_i,$$

where d is a non-negative integer, $\beta_0, \ldots, \beta_d$ are unknown weights, and $\epsilon_i \sim N(0, \sigma^2)$ are independent unobservable random errors. Here we have $Y_i|X_i \sim N(\beta_0 + X_i\beta_1 + \cdots + X_i^d\beta_d, \sigma^2)$.

The example above assumes that $f(X) = \beta_0 + X\beta_1 + \cdots + X^d\beta_d$. Thus the function $f(\cdot)$ is fully characterized by $d+1$ many weights $\beta_0, \ldots, \beta_d$. The model itself is fully characterized by $\beta_0, \ldots, \beta_d$, and σ^2. These quantities can be estimated from the data, and are called *parameters*, and these models are called *parametric models*.

Parametric models have the advantage of being interpretable, and computationally easier to fit. However, they are often inadequate to capture the true $f(\cdot)$. Instead, we can choose not to such parametric assumptions, and let $f(\cdot)$ be flexible.[3] Such models are called *nonparametric models*. These models have the potential to fit complicated functions. The disadvantage of nonparametric models are that they

[3] We do still need to make some assumptions of f to prevent severe overfitting of the data. In spline fitting, for example, we often assume f has certain number of derivatives to ensure smoothness of f.

are computationally expensive, and require larger sample size to produce accurate estimate of $f(\cdot)$.

Note that the models are not to be taken as absolute truth. At best, a model can be only an approximation of the original data generating process.

> *All models are wrong; some models are useful.*
>
> — George E. P. Box, William Hunter and Stuart Hunter, Statistics for Experimenters, second edition, 2005, page 440.

Parameters vs. Hyperparameters

We will discuss many learning methods that have *hyperparameters*: these quantities control the model complexity. Hyperparameters are not estimated from the data, rather they need be "tuned" for better performance of the learning method.

Parameters/Hyperparameters

Parameters – are estimated from the data via fitting the model.

Hyperparameters – control the model complexity and are not estimated; rather, they are *tuned* based on model performance.

Consider, for example, the *polynomial regression model*,

$$Y = \beta_0 + X\beta_1 + \cdot + X^d\beta_d + \epsilon,$$

where d is a non-negative integer. Here the degree of the polynomial, d, is set by the user[4]. The quantity d is not estimated from the data – it is tuned by user based on some performance criterion. Here d is a hyperparameter. In contrast, once d has been set, the quantities $\beta_0, \ldots, \beta_d$ are estimated from the data so that the polynomial best fits the data. Thus $\beta_0, \ldots, \beta_d$ are parameters in this model.

[4] For instance, $d = 1$ results in a simple linear regression

Ideally, we will choose a value of hyperparameter, over all possible values the hyperparameter can take, that gives the best performance of the learning method.[5] In more computationally expensive algorithms, however, such an exhaustive search might not be possible, and we might need to try out a few values of the hyperparameter and choose the best among these values. Some possible options for tuning hyperparameters are as follows:

[5] We will learn about how to measure model performance in future chapters.

- *Grid search:* search over a (fine) grid of possible hyperparameter values – computationally expensive.
- *Random grid search:*[6] search over a set of randomly chosen values from all possible values.

[6] Bergstra, James, and Yoshua Bengio. 2012. "Random Search for Hyper-Parameter Optimization." Journal of Machine Learning Research 13 (Feb): 281–305.

- *Early stopping:* stopping the tuning process when the improvement of model performance becomes negligible.
- *Adaptive resampling:*[7] adaptively resamples candidate values based on near optimal model performance.

[7] Kuhn, Max. 2014. "Futility Analysis in the Cross-Validation of Machine Learning Models." arXiv Preprint arXiv:1405.6974.

There are many more options in literature (e.g, Bayesian tuning using Gaussian process) etc., we will use various tuning methods throughout the course.

Training/Testing the model

The process of tuning the hyperparameter(s) and estimating the model parameters are often done iteratively to tune the hyperparameters. This process is called *training the model*.

Ideally, we would have two separate data sets - one for training the model, and one for testing the model performance.

Training and Testing sets

Training set: the data set used to train learning methods. Tasks performed using this data include tuning hyperparameters, estimating model parameters, comparison of several learning methods, development of features and so on.

Test set: the data set used to produce unbiased estimate of the performance of the *final model* chosen using the training set.

In reality, we often have only one data set available, and as such need to create our own training and test sets. This is done by randomly splitting the data in two parts, say a smaller part containing 20% of the data to be used as the test set, and the remaining 80% to be used as the training set. We will briefly discuss the following sampling techniques in later chapters:

- *Simple random sampling (with or without replacement):* randomly select a few units from the data.
- *Stratified sampling:* used when data contain groups (e.g., "Yes"/"No") that are imbalanced. Stratified sampling is used to ensure that the test set has similar group size ratio as the original data.
- *Up-sampling and down sampling methods, or a combination of both*: used when there is severe group size imbalance (e.g., 95% "No" and 5% "Yes"). In this case, sampling is done by over-representing the rare group, or under-representing the abundant group, or a combination of both, e.g., SMOTE.[8]

[8] Synthetic Minority Over-Sampling Technique by Chawla, et al. (2002), Journal of Artificial Intelligence Research 16: 321–57.

It is extremely important that we do not use the test set in the training phase at all.

Locking down the test set

Do not use the test set of any part of the test set **for training** the model. This includes the tuning of hyperparameters as well. The test set should be used only after we have chosen the final model using the training set.

Using the test set to evaluate model performance during the training phase would effectively include the test set in model training. So when we measure the performance of the final model using the test set, it may result in overfitting since we have already used the test set to build the final model.

With us locking down the test set, we need to think about how to tune hyperparameter(s) to obtain the model with best performance metric. One possible way is to use the training set itself to fit the model and evaluate the performance. However, such a strategy has the same drawback as before: the model might give an overly optimistic performance metric on the training set, but in reality might not perform well in general. To this end, we will learn about three approaches we can use:

- *Holdout method:* We split the training set further in a holdout set[9] (used for model evaluation) and the remaining used for fitting the model.
- *v-fold cross validation:* We randomly divide the data in v roughly equal sized parts (called *folds*). We then designate one fold as the validation set, and the remaining $v - 1$ folds combined as training set. This procedure is then repeated for each of the v folds.
- *Bootstrap:* We take random samples from the training set *with replacement* of the same size as the training set (each of these samples is called a *bootstrap sample*). The observations that are no contained in the bootstrap sample[10] are used as validation set, while the bootstrap sample is used for model fitting.

[9] Sometimes also called a *validation set*.

[10] Also called *out-of-bag (OOB) samples*.

There are other resampling techniques that can be applied in various specific situations such as when we have dependent data (e.g., time series) or small sample sizes (where bootstrap might have biased results). We will not go into details about these approaches.

Supervised learning

Learning methods can be broadly categorized into the three types: supervised, unsupervised and semi-supervised learning. In this section, we discuss about supervised learning. The other two categories will be discussed in the next two sections.

The example discussed in Figure 1 and subsequent discussions were mainly geared towards supervised learning methods. These type of tasks have labeled data[11] as input, and outputs a prediction rule, $\widehat{f}(\cdot)$. The estimated function can then be used to form predictions for a new unlabeled data.

[11] Labeled data have both predictor, X, and response variable, Y, available in the data.

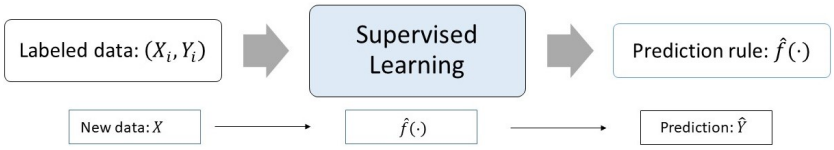

Figure 3: Outline of supervised statistical learning.

Examples of such tasks as *regression* and *classification* problems. The distinction between these two types is based on the nature of the outcome variable, Y: regression problems deal with *numeric* Y, while classification problems involve *categorical* Y.

Regression problems

When a supervised learning deals with problem with a *numeric response*, we call such a problem a regression problem.

Let us consider the `mpg` dataset[12] available in the `ggplot2` library. The dataset contains fuel economy data from 1999 to 2008 for 38 popular cars. A snapshot of the data is shown below.

[12] The `mpg` data is a subset of data that EPA makes available at `https://fueleconomy.gov/` – see the documentation of `mpg` for details.

```
## # A tibble: 234 x 11
##    manufacturer model      displ  year   cyl trans drv     cty   hwy fl    class
##    <chr>        <chr>      <dbl> <int> <int> <chr> <chr> <int> <int> <chr> <chr>
##  1 audi         a4           1.8  1999     4 auto~ f        18    29 p     comp~
##  2 audi         a4           1.8  1999     4 manu~ f        21    29 p     comp~
##  3 audi         a4           2    2008     4 manu~ f        20    31 p     comp~
##  4 audi         a4           2    2008     4 auto~ f        21    30 p     comp~
##  5 audi         a4           2.8  1999     6 auto~ f        16    26 p     comp~
##  6 audi         a4           2.8  1999     6 manu~ f        18    26 p     comp~
##  7 audi         a4           3.1  2008     6 auto~ f        18    27 p     comp~
##  8 audi         a4 quattro   1.8  1999     4 manu~ 4        18    26 p     comp~
##  9 audi         a4 quattro   1.8  1999     4 auto~ 4        16    25 p     comp~
## 10 audi         a4 quattro   2    2008     4 manu~ 4        20    28 p     comp~
## # ... with 224 more rows
```

Suppose we are interested in predicting a car's highway mileage per gallon (`hwy` variable) based on the car's engine displacement in litres (`displ` variable). Here the outcome, `hwy`, is a numeric variable, and thus this is a regression problem. A scatterplot of `hwy` vs `displ` is shown in Figure 4 along with a linear and a nonparametric regression fit.

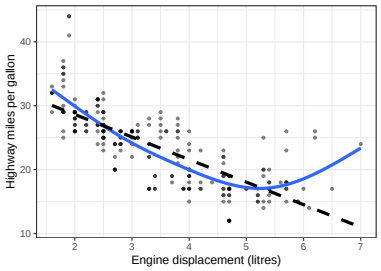

Figure 4: Scatterplot of highway miles per gallon vs. engine displacement (litres). Also shown are a simple linear regression fit and estimated nonparametric regression line.

Here we have fitted two models, one linear and the other nonparametric. In particular, the regression models are as follows:

$$\text{Linear model:} \quad Y_i = \beta_0 + \beta_1 X_i + \epsilon_i,$$
$$\text{Nonparametric model:} \quad Y_i = f(X_i) + \epsilon_i,$$

where $\epsilon_i \sim N(0, \sigma^2)$ are independent random errors. We used *cubic splines*[13] for the nonprametric fit. Note that, since the errors have zero mean, the lines effectively represent $E(Y|X)$ as a function of X:

$$\text{Linear model:} \quad E(Y_i|X_i) = \beta_0 + \beta_1 X_i,$$
$$\text{Nonparametric model:} \quad E(Y_i|X_i) = f(X_i).$$

In this particular data, the regression lines represent how the mean highway mileage changes as function of engine size.

[13] We will learn about splines in our discussion of various regression techniques in later chapters. In a nutshell, cubic splines are piece-wise cubic polynomials used to interpolate the data while also ensuring that the resulting fit is smooth.

Regression problems are not exclusive to continuous outcomes like highway miles per gallon in the previous example. Regression problems might have other type numeric data such as count data. Consider the data `Bikeshare` in the `ISLR2` library, which contains hourly and daily rental of bikes in Capital bikeshare system between the years 2011 and 2012. The data set also contains weather and seasonal variables.[14] A snapshot of the data is shown below.

[14] See `?Bikeshare` for details.

```
## # A tibble: 6 x 15
##   season mnth    day hr    holiday weekday workingday weathersit    temp atemp
##    <dbl> <fct> <dbl> <fct>   <dbl>   <dbl>      <dbl> <fct>        <dbl> <dbl>
## 1      1 Jan       1 0           0       6          0 clear         0.24 0.288
## 2      1 Jan       1 1           0       6          0 clear         0.22 0.273
## 3      1 Jan       1 2           0       6          0 clear         0.22 0.273
## 4      1 Jan       1 3           0       6          0 clear         0.24 0.288
## 5      1 Jan       1 4           0       6          0 clear         0.24 0.288
## 6      1 Jan       1 5           0       6          0 cloudy/misty  0.24 0.258
## # ... with 5 more variables: hum <dbl>, windspeed <dbl>, casual <dbl>,
## #   registered <dbl>, bikers <dbl>
```

Suppose we want to predict the number of bike rentals (`bikers`) using temperature (`temp`). We should note that `bikers` can only take non-nagative integer values. Thus a linear regression model might not be a satisfactory models here since it does not ensure that the

response is integer. One possibility here is to employ a *Poisson regression* model, which assumes

$$Y_i|X_i \sim Poisson\{\lambda(X_i)\}, \text{ with } E(Y_i|X_i) = \lambda(X_i).$$

Since Y_i can not be negative, its expected value conditional on X_i, $\lambda(X_i)$, has to be non-negative as well. In Poisson regression, we can ensure non-negative $\lambda(X)$ by using the following model:

$$log\{\lambda(X_i)\} = \beta_0 + X_i\beta_1, \text{or equivlently } \lambda(X_i) = exp(\beta_0 + X_i\beta_1).$$

A scatterplot of number of bike rentals vs. temperature, and the estimated mean number of bike rentals via Poisson regression in shown in Figure 5.

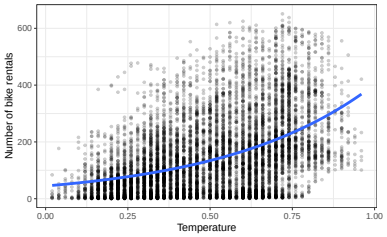

Figure 5: Scatterplot of number of bike rentals vs. temperature overlayed with estimated mean bike rentals as a function of temperature.

The model shown here is far from the optimal model in this situation, and is only shown as a proof of concept. In a proper analysis, one might explore more predictor variables and their relationship with the response.

The two examples shown above are special cases of *generalized linear model*. Recall that we have modeled the conditional mean of Y given X, or the log-transform of the same, as a linear function of X:

$$\begin{aligned} \text{Linear regression:} \quad & E(Y|X) = \beta_0 + X\beta_1 \\ \text{Poisson regression:} \quad & log\{E(Y|X)\} = \beta_0 + X\beta_1 \end{aligned}$$

Also we assumed that Y follows certain distributions:

$$\begin{aligned} \text{Linear regression:} \quad & Y \text{ follows a normal distribution} \\ \text{Poisson regression:} \quad & Y \text{ follows a Poisson distribution} \end{aligned}$$

In general, we can assume that Y follows a distribution in a certain class of distributions called the *exponential family*, and the conditional mean of Y has the form

$$g\{E(Y|X)\} = \beta_0 + X\beta_1,$$

where $g(\cdot)$ is a known function, called a *link function*. This model is called a generalized linear regression model (GLM). Both linear and Poisson regression models are GLMs. There are many other GLMs, such as Logistic regression (which we will discuss in the classification section), Gamma regression, negative binomial regression etc.

We now present a third example of a regression problem with a different flavor than the previous two examples. Consider the `BrainCancer` data[15] in the `ISLR2` library. The data set contains survival times of patients diagnosed with brain cancer, and information on a few other characteristics.[16] A snapshot of data is shown below.

[15] Selingerov´a et al. (2016) Survival of patients with primary brain tumors: Comparison of two statistical approaches. PLoS One, 11(2):e0148733.

[16] See `?BrainCancer` for details.

```
## # A tibble: 6 x 8
##   sex    diagnosis  loc              ki   gtv stereo status  time
##   <fct>  <fct>      <fct>         <int> <dbl> <fct>   <int> <dbl>
## 1 Female Meningioma Infratentorial    90  6.11 SRS         0 57.6
## 2 Male   HG glioma  Supratentorial    90 19.4  SRT         1  8.98
## 3 Female Meningioma Infratentorial    70  7.95 SRS         0 26.5
## 4 Female LG glioma  Supratentorial    80  7.61 SRT         1 47.8
## 5 Male   HG glioma  Supratentorial    90  5.06 SRT         1  6.3
## 6 Female Meningioma Supratentorial    80  4.82 SRS         0 52.8
```

Suppose we want to predict survival time (`time`) based on gross tumor volume, in cubic centimeters (`gtv`). We need to notice another variable of interest: Whether the patient is still alive at the end of the study (`status`, 0=Yes, 1=No). A plot of `time` vs. `gtv` is shown in Figure 6.

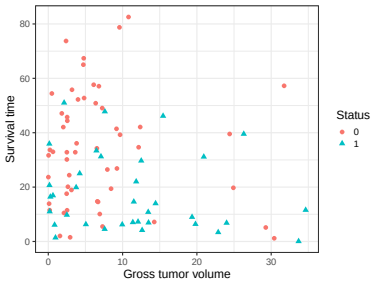

Figure 6: Survival time vs gross tumor size.

While it is tempting to use a simple linear regression in this case, such a model in inappropriate here. Notice that only 53 out of 88 patients were still alive at the end of the study (cyan triangles in the plot). We known the accurate survival times for the patients that passed away. But for the patients that were alive, we can only know that their survival time is *at least* as much as the observed data.

To be specific, for the i-th patient, let T_i be the actual survival time, and C_i is the time when the patient left the study (if they were alive after the study ended). Thus, we must have that $C_i < T_i$, if the patient was alive after the study ended. Here, we are interested in modeling T_i; instead we only observe

$$Y_i = min(T_i, C_i).$$

for the i-th individual. This phenomenon is called *censoring*. The `status` variable, which is an indicator of whether censoring has occured or not, can be defined as

$$\delta_i = \begin{cases} 0, \text{ if } T_i > C_i \\ 1, \text{ otherweise.} \end{cases}$$

Thus our observed data are the pairs (Y_i, δ_i), along with predictors, for $i = 1, \ldots, n$.

Such regression problems do not generally fall into the settings discussed in the previous two examples due to the censoring of the response, and requires different mathematical treatment. We will not go into mathematical details yet - in future chapters, we will learn how to perform regression with survival outcomes by using *Cox's Proportional Hazards Model*. Figure 7 shows a typical output from a survival analysis in form of estimated survival probabilities.

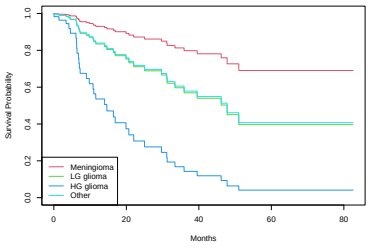

Figure 7: Survival probability for different types of diagnosis.

Classification problems

We call a supervised learning task a *classification problem* is the response variable, Y is categorical. Predicting a categorical response for an item/unit is called *classifying* that item/unit. The methods for classification are often referred to as *classifiers*.

Let us consider the `Default` data in the `ISLR2` package. The dataset[17] contains information on 10,000 customers about whether the customer is a student, whether the customer defaulted on their debt, along with their balance and income. A snpshot of the data is shown below.

[17] This is a simulated data – see `?Default` for more details.

```
## # A tibble: 10,000 x 4
##    default student balance income
##    <fct>   <fct>     <dbl>  <dbl>
##  1 No      No         730. 44362.
##  2 No      Yes        817. 12106.
##  3 No      No        1074. 31767.
##  4 No      No         529. 35704.
##  5 No      No         786. 38463.
##  6 No      Yes        920.  7492.
##  7 No      No         826. 24905.
##  8 No      Yes        809. 17600.
##  9 No      No        1161. 37469.
## 10 No      No           0  29275.
## # ... with 9,990 more rows
```

A few exploratory plots of the data are shown in Figure 8.

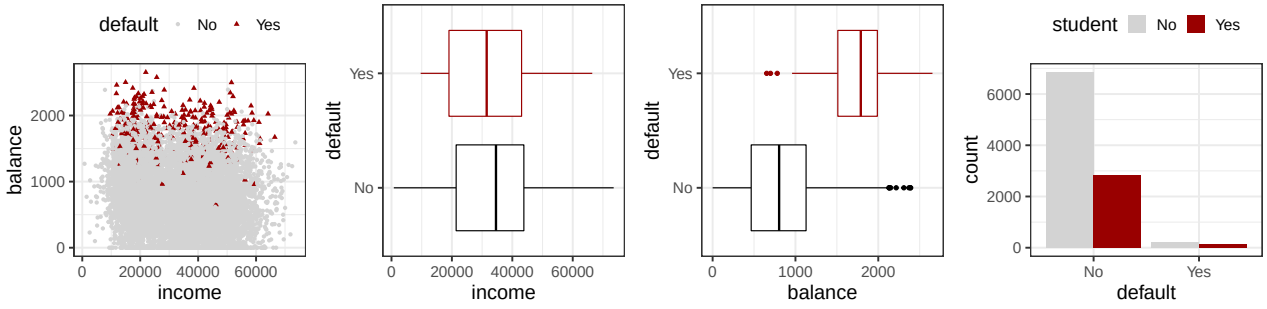

Figure 8: Exploratory plots of the defaults data.

Our goal is to predict whether a customer will default based on their data on the other three characteristics. From a inferential point of view, we might want to know which variable is most associated with default status. This is an example of a *binomial classification problem*, since there are two classes only. The goal of a classification prob-

lem is to produce a rule that classifies new observations to existing classes.

It is tempting to create a binary variable Y (taking the value 0 if the customer does not default, and 1 if the customer does default), pretending that Y is a numeric variable, and run a linear regression. For simplicity, let us only consider balance to be the predictor.[18] Figure 9 shows the plot of default status vs balance, overlayed with the fitted linear regression line (dashed line). We notice the predicted values for lower values of balance are in fact negative! The linear regression model does not take into account that the response is binary, and thus are difficult to interpret.

We will learn various classification techniques in this course. An example of one such classifier is the logistic regression, which models the probabilities of each class as a function of the predictors. Denoting balance as X, the logistic regression posits the model

$$P(Y = 1|X) = \frac{e^{\beta_0 + X\beta_1}}{1 + e^{\beta_0 + X\beta_1}}.$$

The solid line in Figure 9 plots the estimated probabilities as a function of balance. Thus we can build a classification rule as

> Classify a new observation X_{new} to class 1 (Yes) if $\widehat{P}(Y = 1|X_{new}) > 0.5$, classify to 0 ("No") otherwise.

Apart from logistic regression, we will learn many other classifiers, such as, k-nearest neighbors, support vector machine, tree based methods, and so on.

Classification is not limited only for two classes. The example discussed in Figure 1 is essentially a classification problem, but with multiple classes (10 digits). Consider the MNIST image data[19] available in the `dslabs` library. The data set contains image information of handwritten numbers. The set contains 60,000 images for traing and 10,000 images for testing. Figure 10 shows a sample of 10 images in the data set (in grayscale).

[18] It seems customers with higher balance tend to default more.

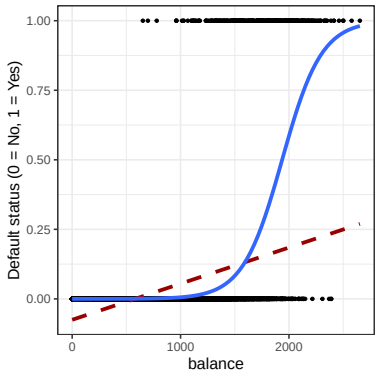

Figure 9: Linear (dashed line) and logistic (solid line) regression fits to the defaults data.

[19] See `http://yann.lecun.com/exdb/mnist/` for the documentation.

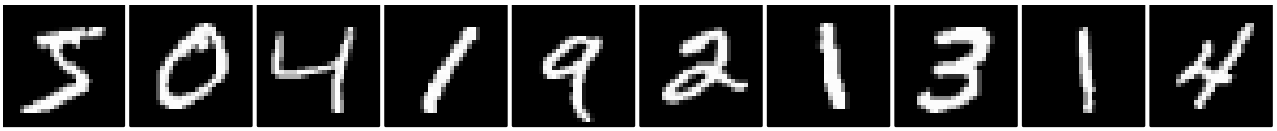

Figure 10: Example images form the MNIST handwriting data.

Our goal is to classify such images to their corresponding digits. In this case, the response is categorical – the digits "0", "1", ..., "9". Such problems are called *multinomial classification problems* as there are multiple categories that the images need to be classified into.

Unsupervised learning

In unsupervised learning, only the predictors X are present, but no response Y is available. In other words, unsupervised learning takes unlabeled data as input. The goal is often to find groups in the data.

Figure 11: Outline of unsupervised statistical learning.

Unsupervised learning tasks can be mainly categorized into two classes: *clustering* and *dimension reduction*. Although, there are other tasks, such as outlier detection (anomaly detection), can also be considered as unsupervised learning.

Clustering

In clustering, we take unlabeled data, and try to *group the units/individuals/observations* into separate *clusters* based on the covariate data.

Cluster

A group of observations that are similar to each other but different than the rest of the (groups of) observations.

Figure 12 demosntrates the basic idea behind clustering. Suppose we have n observations and p features for each observation. We can think of the data as a $n \times p$ matrix, where the features are contained in columns, and observations are in rows. Clustering algorithms group the observations (rows) into separate clusters based on their similarity (measured using features).

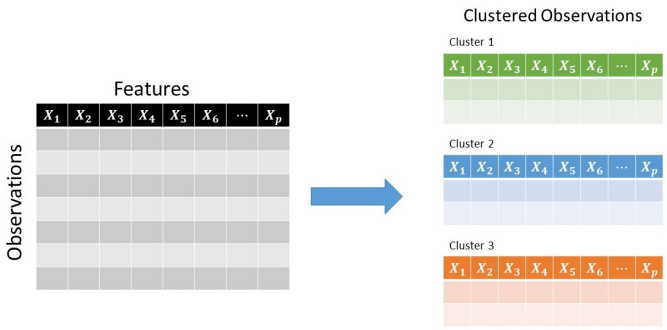

Figure 12: The idea behind clustering. The observations are grouped together based on their similarity to each other. All features are retained.

Notice that there are no pre-defined classes with labels in such tasks. Thus the number of possible clusters (and what they represent)

are also unknown, and has to be set/determined. Figure 13 shows a simulated dataset with two features. The left plot shows the original data which are unlabeled. After applying a clustering algorithm[20], the observations were clustered into two groups.

[20] In this case, we applied K-means method, which we will learn later.

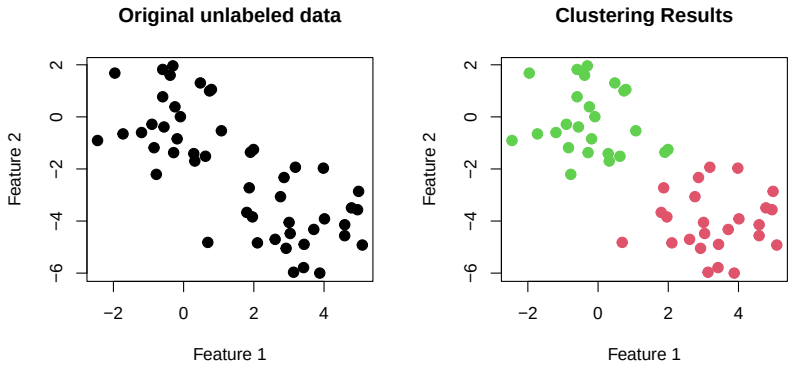

Figure 13: Example of clustering. The clustering algorithm takes unlabeld data, and groups them into clusters. In this example, the number of clusters is set to 2.

As an practical example, let us consider the `GvHD.control` control dataset in the `mclust` package.[21] A snapshot of the data is shown below.

[21] ?see `?GvHD` for more details. Original source is R. R. Brinkman, et al. (2007). High-content flow cytometry and temporal data analysis for defining a cellular signature of Graft-versus-Host Disease. Biology of Blood and Marrow Transplantation, 13: 691-700.

```
## # A tibble: 6,809 x 4
##      CD4  CD8b   CD3   CD8
##    <dbl> <dbl> <dbl> <dbl>
##  1   199   420   132   226
##  2   294   311   241   164
##  3    85    79    14   218
##  4    19     1   141   130
##  5    35    29     6   135
##  6   376   346   138   176
##  7    97   329   527   406
##  8   200   342   145   189
##  9   422   433   163    47
## 10   391   390   147   190
## # ... with 6,799 more rows
```

The dataset represents Graft-versus-Host Disease (GvHD) data. We are only looking at GvHD positive patients. This sample has four biomarkers (CD4, CD8b, CD3, and CD8). The goal is to find whether there are any sub-populations present in the sample. Figure 14, left panel, shows a pairs-plot along with bivariate and univariate densities of the four biomarkers.

After running a clustering algorithm, we might find three clusters, as shown in the right panel in Figure 14. The choice of number

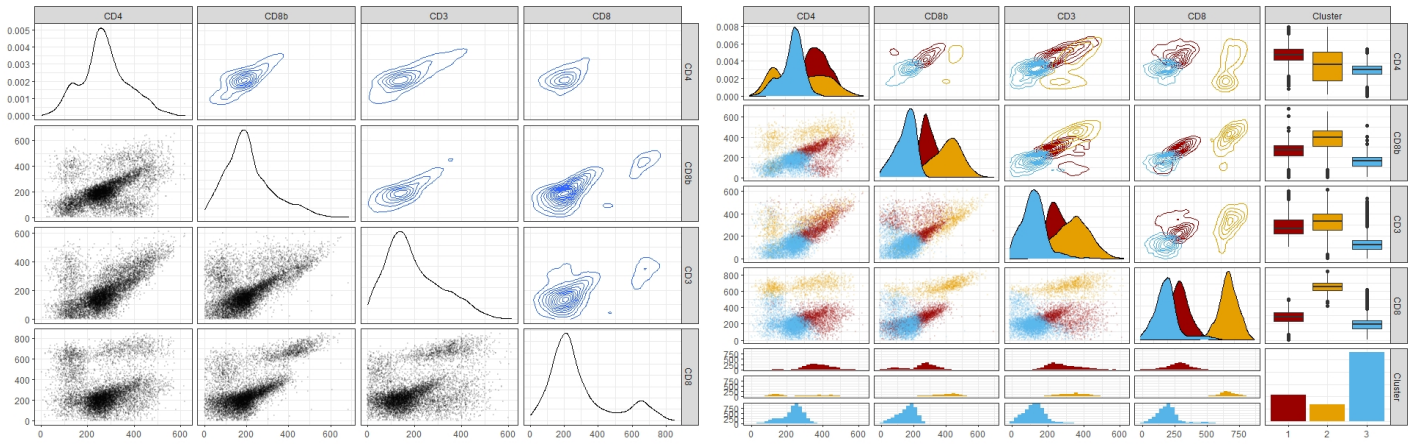

Figure 14: Pairs plot for the GvHD data. The left panel shows the original data. The right panel shows the results of clustering. The colors (red, blue and orange) indicate three different clusters. Box plots and histograms show how the values of the continuous variables vary between clusters.

of clusters can be done by maximizing various measures of model performance. However, in many situations, number of clusters are chosen based on the *interpretation* of the resulting clusters. For example, from the clusering results above, we can see the following:

- The blue cluster has the highest number of patients in it. It seems that, generally, patients in this cluster tend to have smaller values of the four bimarkers compared to the other two clusters.
- In contrast, the orange cluster contains patients having higher `CD8` levels compared to others.

As such we can not know what these patterns mean based only on the numeric results – we require expert knowledge to understand the biological impact of such patterns.

Dimension reduction

The goal of dimension reduction methods is to transform a large number of variables into a set of smaller number of variable. Often, dealing with a large number of predictors can be problem due to various reasons:

- Difficulty in visualizing the data.
- Smaller sample size compared to number of predictors ($p >> n$ *problem*), curse of dimensionality.
- Collinearity among the variables.

By transforming the original variables into a smaller number of new features allows us to bypass some or all of the above mentioned difficulties.

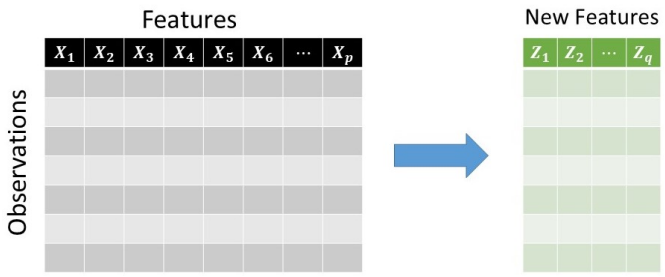

Figure 15: The idea behind dimension reduction. The original features are collaped together to form fewer number of new features. All observations are retained.

Figure 15 presents the main idea of dimension reduction. Suppose we have n observations and p features for each observation. We can think of the data as a $n \times p$ matrix, where the features are contained in columns, and observations are in rows. Let us denote the features by $X_1, \ldots, X_p$. Dimension reduction methods collapse the columns to create new features, $Z_1, \ldots, Z_{iq}$. Ideally, we would have $q < p$. For example, a popular dimension reduction technique is Principal Component Analysis (PCA), which creates new features as linear combination of the original features:

$$Z_1 = a_1 X_1 + \ldots + a_p X_p,$$

$$Z_2 = b_1 X_1 + \ldots + b_p X_p,$$

and so on, where $a_1, \ldots, a_p$, and $b_1, \ldots, b_p$, and so on, are estimated from the data so that the new features capture most of the variation in the observed data. Other methods might use nonlinear transformations as well.

Take for example the `banknote` data in the `mclust` package.[22] The data contains six measurements made on 100 genuine and 100 counterfeit old-Swiss 1000-franc bank notes. A snapshot of the data is shown below.[23]

[22] See `?banknote` for details.

[23] We have omitted the `Staus` variable which tells us whether a note is genuine or counterfeit. This is because unsupervised learning does not require a target/outcome variable.

```
## # A tibble: 200 x 6
##    Length  Left Right Bottom   Top Diagonal
##     <dbl> <dbl> <dbl>  <dbl> <dbl>    <dbl>
##  1   215.  131   131.    9     9.7     141
##  2   215.  130.  130.    8.1   9.5     142.
##  3   215.  130.  130.    8.7   9.6     142.
##  4   215.  130.  130.    7.5  10.4     142
##  5   215   130.  130.   10.4   7.7     142.
##  6   216.  131.  130.    9    10.1     141.
##  7   216.  130.  130.    7.9   9.6     142.
##  8   214.  130.  129.    7.2  10.7     142.
##  9   215.  129.  130.    8.2  11       142.
## 10   215.  130.  130.    9.2  10       141.
## # ... with 190 more rows
```

A pairs plot of the data set is shown in Figure 16, left panel. After performing PCA, we can find two new features, engineered from the original six features, that capture roughly 88% of the total variation (defined as the sum of variances of the variable present in the data) of the data. A plot of these new features in shown in the right panel in Figure 16.

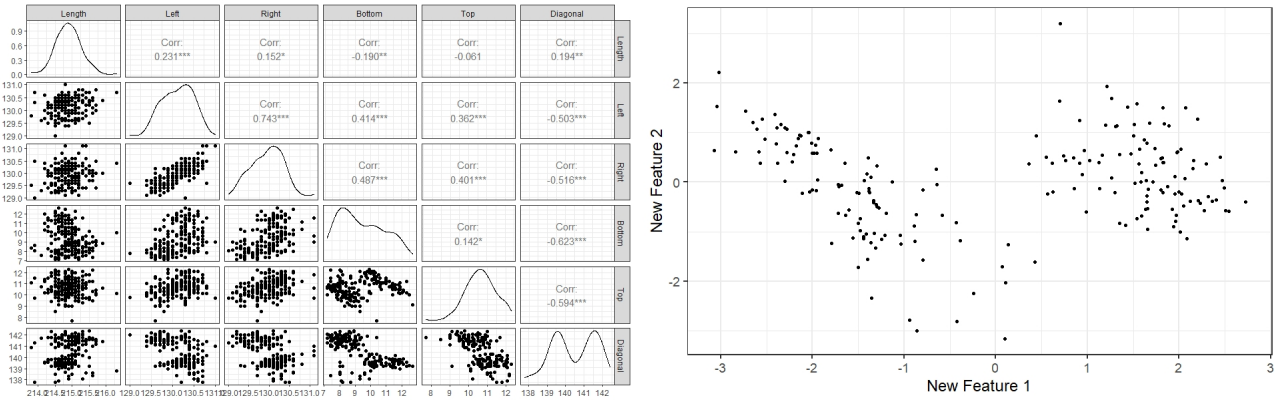

Figure 16: Pairs plot of the variables in the banknotes data (left panel) and new features obtained via pca (right panel).

We have effectively mapped the six variables in the dataset into two new features, and thus reduced the dimension by 4. Each observation (a six dimensional vector) is mapped to a two dimensional point in the new plot, this making visualization much easier.

Semi-supervised learning

These type of tasks fall between supervised and unsupervised problem. They learn from both labeled data (both X and Y are present) and unlabeled data (only X present). Typically, the amount of unlabeled data far exceed the amount of labeled data in such problems.

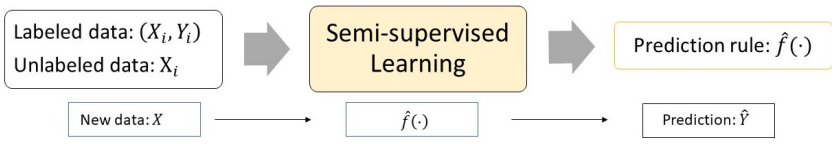

Figure 17: Outline of semi-supervised statistical learning.

Semi-supervised learning arises from the idea that, often, labeling data is resource intensive[24] while unlabeled data are cheaper to obtain. Semi-supervised learning proceeds by first building a model based only on the labeled data, and then predicting the labels (Y) for the unlabeled data. Finally, a new model is trained with all the data combined with the predicted labels.

[24] For example, examining a CAT scan image and determining whether there is an anomaly requires time consuming, expensive and sometimes error prone effort from experts. For large data sets, such tasks might well be infeasible.

Application of semi-supervised learning include speech analysis, text document classification, web-content classification, among many others. Due to time constraint, we will not cover this type of machine learning methods in this course.

Machine learning with R

R is a freely available language for statistical computing and visualization. For this course, please install **R** (this is required) and the IDE **R Studio**:

- R: `https://cran.r-project.org/`
- R Studio: `https://www.rstudio.com/`

While R studio is not required, it is highly recommended as it will make R programming and managing R projects much easier than just using base R.

Resources

We will use various R packages in this course such as *caret*, *h2o*, *mlr3*, among others. See `https://cran.r-project.org/web/views/MachineLearning.html` for a (large) list of machine learning and statistical learning packages available in R. Keep in mind this list is not comprehensive, and there may be other packages out there. If you want to utilize Python's machine learning libraries from, you can use the `reticulate` package to call python from R.

Communication

While writing project reports and homework in usual word processors is acceptable, I would highly encourage you to learn and use `R markdown` for this purpose. See `https://rmarkdown.rstudio.com/` for the capabilities of R markdown.