MACHINE LEARNING AND STATISTICS

an introduction

Samuel S. Watson

Contents

NOTE: THIS IS AN EARLY-STAGE DRAFT

1 Programming in Python and R

Python and R are the two most commonly used programming environments in data science.

1. **Python**. A popular general purpose language that has also seen widespread adoption in the statistics and machine learning space over the past decade following the introduction of powerful scientific computing packages (NumPy, SciPy, and scikit-learn).
2. **R**. The lingua franca in the statistics community for 15+ years; has extensive tooling available and over 10,000 statistics packages

1.1 Environment and workflow

You can download Python 3 from `anaconda.com/download` and R from `r-project.org`.

Python and R support the same modes of interaction as Julia:

1. *REPL*. Launch a read-eval-print loop from the command line. Any code you enter will be executed immediately, and any values returned by your code will be displayed. To start a session, run `python` or `r` from the terminal.
2. *Script*. Save a file called `example.py` or `example.r` and run `python example.py` or `rscript example.r` from the command line to execute all the code in the script.
3. *Notebook*. Like a REPL, but allows inserting text and math expressions, grouping code into blocks, etc. **Jupyter notebooks*** support Julia, Python, R and many other languages.
4. *IDE*. RStudio is an excellent IDE for R. There are many IDEs for Python, including PyCharm, Visual Studio Code, and Atom.

* Jupyter is a portmanteau of these three language names

1.2 Syntax differences from Julia

The syntax for variable assignent is slightly different in R:

Python

```
age = 41
```

R

```
age <- 41
```

Numerical values are floats by default in R; integer literals are specified with an L suffix, as in `10L`. Python behaves the same as Julia in this respect:

Python

```python
type(2.0) # float
type(2) # int
```

R

```r
typeof(2) # float
typeof(2L) # integer
```

The exponentiation operator is `^` in R and `**` in Python.

To find the length of a string: `nchar("hello")` in R; `len("hello")` in Python,

Concatenating strings: `"Hello " + "World"`, `paste("Hello ","World")`

In Python the boolean values are `True` and `False`, and R they are `TRUE` and `FALSE`. Boolean operators are the same in R as in Julia (`&&` `||` `!`), while in Python they are the corresponding English words: `and` `or` `not`

Blocks in Python are delineated using indentation, while R uses curly braces. Also, conditions in `if` statements are parenthesized in R.

Python

```python
if x > 0:
    print("x is positive")
elif x == 0:
    print("x is zero")
else:
    print("x is negative")
```

R

```r
if (x > 0) {
  print("x is positive")
}
else if (x == 0) {
  print("x is zero")
}
else {
  print("x is negative")
}
```

Python uses the keyword `def` for function definitions. Unlike in Julia, the `return` keyword is required in Python for the function to return a value.

Python

```python
def f(x,y):
    x2 = x * x
    return x2 + (y*x2+1)**(1/2)
```

In R, functions are defined by writing an *anonymous* function like `function(x) {x^2}` and binding it to a name:

R

```r
f  <- function(x,y) {
  x2 <- x * x
  x2 + sqrt(y*x2+1)
}
```

In Python, lists are indexed from *0*.

Python

```python
myList = [1,2,"a",[10,8,9]]
myList[2] # evaluates to "a"
myList[3][2] # evaluates to 9
2 in myList # evaluates to True
```

In R, the more commonly used data type for list-type operations is a *vector*, but R does have lists as well:

R

```r
myList <-list(1,2,"a",list(10,8,9))
myList[3] # evaluates to "a"
myList[4][2] # evaluates to 8
2 %in% myList # evaluates to true
```

NumPy arrays may be defined by calling `np.array` on a Python list, and in R you use the function `c` to define a vector, as in `c(1,2,3)`.

Python

```python
import numpy as np
import timeit
A = np.random.randn(10**6)
L = list(A)
n = 10**3
timeit.timeit("sum(L)","from __main__ import L",number=n)/n # 0.04 seconds
timeit.timeit("np.sum(L)","from __main__ import A, np",number=n)/n  # 0.0005 seconds
```

Python has list comprehensions just like Julia:

Python

```python
perfect_squares = [n**2 for n in range(1,101)]
```

R doesn't have list comprehensions, but you can achieve a similar effect using `sapply`:

R

```r
sapply(1:100,function(x) {return(x*x)})
```

1. Ranges: `range(1,11)` in Python, `1:10` in R. If we want only the odd integers, then we use `range(1,11,2)` in Python and `seq(1,10,2)` in R.
2. Concatenation: `[1,2,3] + [8,9,10]` in Python `c(c(1,2,3),c(8,9,10))` in R
3. Appending: `myList.append(3)` in Python, `myVector <- c(myVector,3)` in R.
4. Slicing*: `A[-3:]` in Python, `A[(length(A)-2):length(A)]` in R.
5. Finding the indices where a certain condition holds: `np.which(A > 2)` in Python, `which(A > 2)` in R.
6. Setting multiple array values: `A[:5] = 1` in Python, `A[1:5] <- 1` in R.

* These examples select the last three elements of `A`

The syntax for `while` and `for` blocks is entirely analogous to syntax for conditional blocks:

Python

```python
def isprime(n):
    for j in range(2,n):
        if n % j == 0:
            return False
    return True
```

R

```r
isprime <- function(n) {
  for (j in 2:(n-1)) {
    if (n %% j == 0) {
      return(FALSE)
    }
  }
  return(TRUE)
}
```

1.3 Package Ecosystems

The scientific computing tools in Python are organized around a relatively small number of fine-tuned and extensive packages. Some of the most important ones are:

1. **NumPy**. Fast multidimensional arrays; the basis for most scientific computing packages in Python.
2. **SciPy**. General purpose tools for scientific computing in Python.
3. **Matplotlib**. Standard plotting package in Python.
4. **Seaborn**. Data visualization library built on matplotlib.
5. **plotnine**. Clone of R's ggplot2.
6. **Pandas**. Data frames: structures for storing and manipulating tabular data).
7. **Scikit-Learn**. Machine learning.

The most commonly used suite of data analysis packages in R is Hadley Wickham's **tidyverse**.

1. **ggplot2**, for data visualisation. Graphics are built as a sum of *layers*, which consist of a data frame, a **geom**, a **stat**, and a mapping from the data to the geom's **aesthetics** (like `x`, `y`, `color`, or `size`). The appearance of the plot can be customized with `scales`, `coords`, and `themes`.
2. **dplyr**, for data manipulation. The primary functions in this package filter rows, sort rows, select columns, add columns, group, and aggregate the columns of a grouped data frame:

```
flights %>%
  filter(month == 1, day < 5) %>%
  arrange(day, distance) %>%
  select(month, day, distance, air_time) %>%
  mutate(speed = distance / air_time * 60) %>%
  group_by(day) %>%
  summarise(avgspeed = mean(speed,na.rm=TRUE))
```

3. **tibble**, a variant of `data.frame` designed to work optimally with tidyverse packages

See *R for Data Science* to learn more about the tidyverse packages.

In Python (with the Anaconda installation), you can install a module using `conda install modulename` from the command line. A module (let's say NumPy) may be imported into a Python session or script using `import numpy as np`. Then functions and variables from that package can be referenced with the given abbreviation (for example, `np.array` makes an array).

In R, do `install.packages("modulename")` from an R session, and then `library(modulename)` to load the package. For example, you can run `install.packages("tidyverse")` to intall all of the tidyverse packages.

1.4 Data structures

Python's approach to data types is quite different from Julia's. The idea is to combine types and related functions into a single object called a **class**. For example:

PYTHON

```
class Album(object):
    def __init__(self,name,artist,year,songs):
        self.name = name
        self.artist = artist
        self.year = year
        self.length = length
```

The `__init__` function is called whenever a new `Album` is created:

JULIA/PYTHON

```
A = Album("Abbey Road","The Beatles",1969,"47:23")
```

The **attributes** of `A` may be then be accessed via `A.name`, `A.artist`, etc. We can also define functions for custom types in the definition of the class:

Python

```python
class Album(object):
    def __init__(self,name,artist,year,songs):
        self.name = name
        self.artist = artist
        self.year = year
        self.length = length

def numYearsAgo(self):
    from datetime import datetime
    return datetime.now().year - self.year
```

1.4.1 File I/O

1.4.1.1 Text files

To read or write text files in Python, it's good practice to use a `with` block in conjunction with the function `open`. This method ensures that the file is properly closed at the end.

Python

```python
# Reading a file:
with open("workfile.txt") as f:
    read_data = f.read()
# Writing a file:
with open("workfile.txt","w") as f:
    f.write("new file contents")
```

If the file is too large to read it all at once, you can read it one line at a time using `readline` . Make sure to close the file when you're done.

Python

```python
f = open("workfile.txt")
first_line = f.readline()
...
f.close()
```

1.4.1.2 CSV files

Julia, Python, and R all use their tabular data storage type a *data frame*. Data frames in Python are supplied by `pandas`, and data frame functionality does not have to be imported in R because it's part of the base language. To create a data frame from a local CSV file (and then write it back to disk):

Python

```python
import pandas
myDataFrame = pandas.read_csv("data.csv")
pandas.write_csv(myDataFrame)
```

R

```r
myDataFrame = read.csv("data.csv")
write.csv(myDataFrame)
```

1.5 Speed

1.5.1 Vectorized operations

Loops written explicitly in Python and R are very slow compared to Julia's loops. As a result, the idiom in Python and R is to disguise loops in vector operations whenever possible. This allows the loop to be performed using optimized code written by package developers, rather than directly in the Python or R interpreter. Vectorizing code can also help with clarity in many cases.

For example, if we have two long lists `A` and `B` and want to calculate the vector sum `A+B`, we could use the list comprehension `[A[k] + B[k] for k in range(n)]`. However, it is preferable to work with NumPy arrays and use the built-in addition operator:

Python

```python
import numpy as np
n = 10000
A = np.random.rand(n)
B = np.random.rand(n)
np.array(A) + np.array(B)
```

If we wanted to calculate the running sum of the elements of A, rather than

```python
[sum(A[:i]) for i in range(len(A))]
```

we should use the NumPy function `cumsum`, as in

Python

```python
import numpy as np
np.cumsum(A)
```

In R we can add vectors directly using the `+` operator, and we can also find the cumulative sum of an array using `cumsum`.

R

```r
A <- runif(10000) # 10000 uniform-[0,1] random samples
cumsum(A)
```

Exercise 1.5.1

In R and in Python, define a random vector of length 1000 and set to zero all of the entries of the vector which are below the median of the entries of the vector.

Do this without writing any loops, using the Python functions `np.where` and `np.median` and the R functions `which` and `median`.

1.5.2 Compilation

In Julia you can sum the first billion integer square reciprocals in a few seconds with a basic loop:

Julia

```julia
function pisum()
    sum = 0.0
    for k = 1:10^9
        sum += 1.0/(k*k)
    end
    sum
end
```

You could write an analogous function in Python or R, but it would take a long time to run. Vectorization doesn't help with this problem, because it's time consuming just to populate an array with a billion entries. Fortunately, there is a Python package which helps achieve some of the same performance benefits you have in Julia: *Numba*. If we load Numba and place a `@jit` decorator before our function, then Numba will compile it into fast machine code for us. For example:

Python

```python
import numba
@numba.jit
def pisum():
    sum = 0.0
    for k in range(1,10**9+1):
        sum += 1.0/(k*k)
    return sum
```

JIT stands for *just-in-time* compilation, which refers to the technique of compiling code on the fly as needed. *Ahead-of-time* compilation saves time when the program runs, but it is less flexible since we have to determine in advance which types we'll need to call the function on. There is an ahead-of-time compilation system for Python: *Cython*. However, using Cython is rather more complex than using Numba, since Cython is a different language with fewer features than regular Python, and extra syntax for communicating information about types to the compiler.

Exercise 1.5.2

Compare the run time for the jitted `pisum` function and that of the regular `pisum` function.

2 Statistics

2.1 Point estimation

In Section refsubsubsec:KDE, we discussed the problem of estimating a distribution given a list of independent samples it. Now we turn to the simpler task of **point estimation**: estimating a single real-valued feature (such as the mean, variance, or maximum) of a distribution. We begin by formalizing the notion of a real-valued feature of a distribution.

Definition 2.1.1: Statistical functional

A statistical functional is any function T from the set of distributions to $[-\infty, \infty]$.

For example,

$$T_1(\nu) = \text{the mean of } \nu$$

is a statistical functional, as is the *maximum* functional

$$T_2(\nu) = F^{-1}(1), \text{where } F \text{ is the CDF of } \nu.$$

To give a more complicated example, we define $T_3(\nu)$ to be the expected value of the difference between the greatest and least of 10 independent random variables with common distribution ν. Then T_3 also a statistical functional*.

* Note that we could estimate T_3 by simulation, though it may be difficult to calculate analytically.

Given a statistical functional, our goal will be to use a list of independent samples from ν to estimate $T(\nu)$.

Definition 2.1.2: Estimator

An estimator $\widehat{\theta}$ is a random variable which is a function of n i.i.d. random variables.

Example 2.1.1

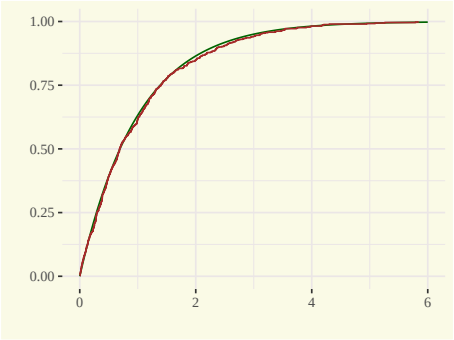
Draw 500 independent samples from an exponential distribution with parameter 1. Plot the function $\widehat{F}$ which maps x to the proportion of samples at or to the left of x on the number line. Compare to the CDF of the exponential distribution with parameter 1.

Solution

We use the step geom to graph $\widehat{F}$:

```
n <- 500
xvals = seq(0,8,length=100)

ggplot() +
  geom_line(aes(x=xvals,y=1-exp(-xvals))) +
  geom_step(aes(x=sort(rexp(n)),y=(1:n)/n))
```



Example 2.1.1 suggests an idea: since the unknown distribution is typically close* to the measure $\widehat{\nu}$ which places mass $\frac{1}{n}$ at each of the observed samples. Then we can build an estimator of $T(\nu)$ by plugging $\widehat{\nu}$ into T.

* In the sense of proximity of CDFs; see Section 2.3.1.

Definition 2.1.3: Plug-in estimator

The plug-in estimator of $\theta = T(\nu)$ is $\widehat{\theta} = T(\widehat{\nu})$.

Example 2.1.2

Find the plug-in estimator of the mean of a distribution. Find the plug-in estimator of the variance.

Solution

The plug-in estimator of the mean is the mean of the empirical distribution, which is the average of the locations of the samples. We call this the **sample mean**:

$$\overline{X} = \frac{X_1 + \cdots + X_n}{n}.$$

Likewise, the plug-in estimator of the variance is **sample variance**

$$S^2 = \frac{1}{n}\left((X_1 - \overline{X})^2 + (X_2 - \overline{X})^2 + \cdots + (X_n - \overline{X})^2\right).$$

Ideally, an estimator $\widehat{\theta}$ is close to θ with high probability. We will see that we can decompose the question of whether $\widehat{\theta}$ is close to θ into two sub-questions: is the *mean* of $\widehat{\theta}$ close to θ, and is $\widehat{\theta}$ close to its mean with high probability?

Definition 2.1.4: Bias

The bias of an estimator $\widehat{\theta}$ is

$$\mathbb{E}[\widehat{\theta}] - \theta.$$

An estimator is said to be **biased** if its bias is nonzero and **unbiased** if its bias is zero.

Example 2.1.3

Consider the estimator

$$\widehat{\theta} = \max(X_1, \ldots, X_n)$$

of the maximum functional. Assuming that the distribution has a density function, show that $\widehat{\theta}$ is biased.

Solution

If ν is a continuous distribution, then the probability of the event $\{X_i < T(\nu)\}$ is 1 for all $i = 1, 2, \ldots, n$. This implies that $\widehat{\theta} < T(\nu)$ with probability 1. Taking expectation of both sides, we find that $\mathbb{E}[\widehat{\theta}] < T(\nu)$. Therefore, this estimator has negative bias.

Zero or small bias is a desirable property of an estimator: it means that the estimator is accurate *on average*. The second desirable property of an estimator is for the probability mass of its distribution to be concentrated near its mean:

Definition 2.1.5: Standard error

The standard error $\mathrm{se}(\widehat{\theta})$ of an estimator $\widehat{\theta}$ is its standard deviation.

Example 2.1.4

Find the standard error of the sample mean if the distribution ν with variance σ^2.

Solution

We have

$$\operatorname{Var}\left(\frac{X_1 + X_2 + \cdots + X_n}{n}\right) = \frac{1}{n^2}(n \operatorname{Var} X_1) = \frac{\sigma^2}{n}.$$

Therefore, the standard error is $\sigma / \sqrt{n}$.

If the expectation of an estimator of θ is close to θ and if the estimator close to its average with high probability, then it makes sense that $\widehat{\theta}$ and θ are close to each other with high probability. We can measure the discrepancy between $\widehat{\theta}$ and θ directly by computing their average squared difference:

Definition 2.1.6: Mean squared error

The mean squared error of an estimator $\widehat{\theta}$ is $\mathbb{E}[(\widehat{\theta} - \theta)^2]$.

As advertised, the mean squared error decomposes as a sum of *squared bias* and *squared standard error*:

Theorem 2.1.1

The mean squared error of an estimator θ is equal to its variance plus its squared bias:

$$\mathbb{E}[(\widehat{\theta} - \mathbb{E}[\widehat{\theta}])^2] + (\mathbb{E}[\widehat{\theta}] - \theta)^2.$$

Proof

The idea is to add and subtract the mean of $\widehat{\theta}$. We find that

$$\begin{aligned}\mathbb{E}[(\widehat{\theta} - \theta)^2] &= \mathbb{E}[(\widehat{\theta} - \mathbb{E}[\widehat{\theta}] + \mathbb{E}[\widehat{\theta}] - \theta)^2] \\ &= \mathbb{E}[(\widehat{\theta} - \mathbb{E}[\widehat{\theta}])^2] + 2\mathbb{E}[(\widehat{\theta} - \mathbb{E}[\widehat{\theta}])(\mathbb{E}[\widehat{\theta}] - \theta)] + (\mathbb{E}[\widehat{\theta}] - \theta)^2.\end{aligned}$$

The middle term is zero by linearity of expectation.

If the bias and standard error of an estimator both converge to 0, then the estimator is *consistent*:

Definition 2.1.7: Consistent

An estimator is **consistent** if $\widehat{\theta}$ converges to θ in probability as $n \to \infty$.

Example 2.1.5

Show that the plug-in maximum estimator $\widehat{\theta}_n = \max(X_1, \ldots, X_n)$ of $\theta = T(\nu) = F^{-1}(1)$ is consistent, assuming that the distribution belongs to the parametric family $\{\text{Unif}([0, b]) : b \in \mathbb{R}\}$.

Solution

The probability that $\widehat{\theta}_n$ is more that ϵ units from θ is equal to the probability that every sample is less than $\theta - \epsilon$, which by independence is equal to

$$\left(\frac{\theta - \epsilon}{\theta}\right)^n.$$

This converges to 0 as $n \to \infty$, since $\frac{\theta - \epsilon}{\theta} < 1$.

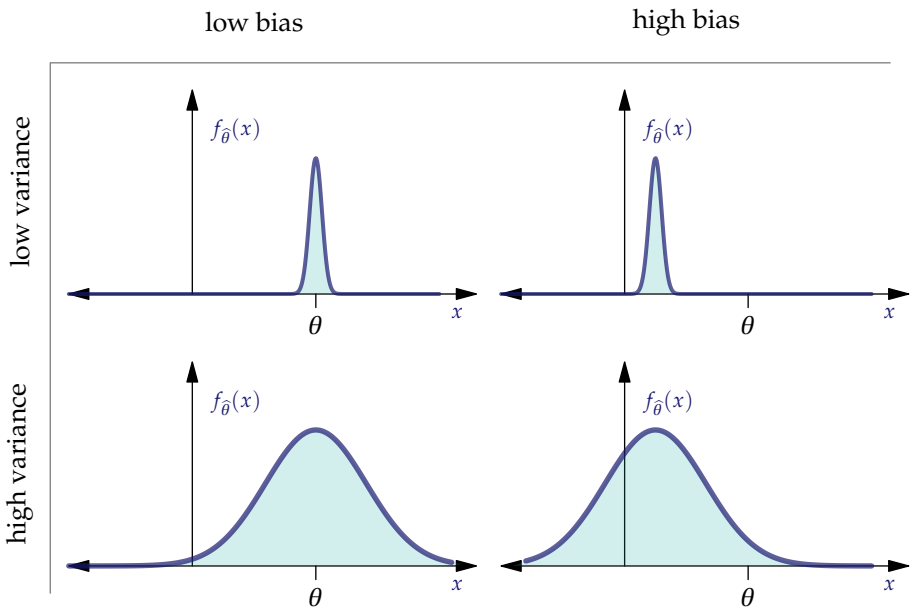


Figure 2.1 An estimator of θ has high or low bias depending on whether its mean is far from or close to θ. It has high or low variance depending on whether its mass is spread out or concentrated.

Example 2.1.6

Show that the sample variance $S^2 = \frac{1}{n}\sum_{i=1}^{n}(X_i - \overline{X})^2$ is biased.

Solution

We will perform the calculation for $n = 3$. It may be generalized to other values of n by replacing 3 with n and 2 with $n-1$. We have

$$\mathbb{E}[S^2] = \frac{1}{3}\mathbb{E}\left[\left(\frac{2}{3}X_1 - \frac{1}{3}X_2 - \frac{1}{3}X_3\right)^2 + \left(\frac{2}{3}X_2 - \frac{1}{3}X_3 - \frac{1}{3}X_1\right)^2 + \left(\frac{2}{3}X_3 - \frac{1}{3}X_1 - \frac{1}{3}X_2\right)\right]^2$$

Squaring out each trinomial, we get $\frac{4}{9}X_1^2$ from the first term and $\frac{1}{9}X_1^2$ from each of the other two. So altogether the X_1^2 term is $\frac{6}{9}X_1^2$. By symmetry, the same is true of X_2^2 and X_3^2. For cross-terms, we get $-\frac{4}{9}X_1X_2$ from the first squared expression, $-\frac{4}{9}X_1X_2$ from the second, and $\frac{2}{9}X_1X_2$ from the third. Altogether, we get $-\frac{6}{9}X_1X_2$. By symmetry, the remaining two terms are $-\frac{6}{9}X_1X_3 - \frac{6}{9}X_2X_3$.

Recalling that $\mathrm{Var}(X) = \mathbb{E}[X^2] - \mathbb{E}[X]^2$ for any random variable X, we have $\mathbb{E}[X_1^2] = \mu^2 + \sigma^2$, where μ and σ are the mean and standard deviation of the distribution of X_1 (and similarly for X_2 and X_3. So we have

$$\mathbb{E}[S^2] = \frac{1}{3}\left(\frac{6}{9}(X_1^2 + X_2^2 + X_3^2) - \frac{6}{9}(X_1X_2 + X_1X_3 + X_2X_3)\right) \quad = \frac{1}{3}\cdot\frac{6}{9}(3(\sigma^2+\mu^2) - 3\mu^2) = \frac{2}{3}\sigma^2.$$

If we repeat the above calculation with n in place of 3, we find that the resulting expectation is $\frac{n}{n-1}\sigma^2$.

Motivated by Example 2.1.6, we define the **unbiased sample variance**

$$\frac{1}{n-1}\sum_{i=1}^{n}(X_i-\overline{X})^2.$$

2.2 Confidence intervals

It is often of limited use to know the value of an estimator given an observed collection of samples, since the single value does not indicate how close we should expect θ to be to $\widehat{\theta}$. For example, if a poll estimates that a randomly selected voter has 46% probability of being a supporter of candidate A and a 42% probability of being a supporter of candidate B, then knowing more information about the distributions of the estimators is essential if we want to know the probability of winning for each candidate. Thus we introduce the idea of a *confidence interval*.

Definition 2.2.1: Confidence interval

Consider an unknown probability distribution ν from which we get n independent samples $X_1,\dots,X_n$, and suppose that θ is the value of some statistical functional of ν. A **confidence interval** for θ is an interval-valued function of the sample data $X_1,\dots,X_n$. A confidence interval has **confidence level** $1-\alpha$ if it contains θ with probability at least $1-\alpha$.

Exercise 2.2.1

Use Chebyshev's inequality to show that if $\widehat{\theta}$ is unbiased, then $(\widehat{\theta}-k\,\mathrm{se}(\widehat{\theta}),\widehat{\theta}+k\,\mathrm{se}(\widehat{\theta}))$ is a $1-\frac{1}{k^2}$ confidence interval.

If $\widehat{\theta}$ is approximately normally distributed, then we can give much tighter confidence intervals using the normal approximation:

Exercise 2.2.2

Show that if $\widehat{\theta}$ is approximately normally distributed, then $(\widehat{\theta}-k\,\mathrm{se}(\widehat{\theta}),\widehat{\theta}+k\,\mathrm{se}(\widehat{\theta}))$ is a $1-\frac{1}{2}(1-\Phi(k))$ confidence interval, where Φ is the CDF of the standard normal distribution.

If we are estimating a *function*-valued feature of ν rather than a single number (for example, a regression function), then we might want to provide a confidence *band* which traps the whole graph of the function with specified probability.

Definition 2.2.2: Confidence band

Let $I\subset\mathbb{R}$, and suppose that T is a function from the set of distributions to the set of real-valued functions on I. A $1-\alpha$ **confidence band** for $T(\nu)$ is pair of random functions $y_{\min}$ and $y_{\max}$ from I to $\mathbb{R}$ defined in terms of n independent samples from ν and having $y_{\min}\le T(\nu)\le y_{\max}$ everywhere on I with probability at least $1-\alpha$.

2.3 Empirical CDFs and bootstrapping

2.3.1 Empirical CDF convergence

Let's begin this section by revisiting Example 2.1.1, where we observed that the CDF of the empirical distribution of an independent list of samples from a distribution tends to be close to the CDF of the distribution itself. The **Glivenko-Cantelli theorem** is a mathematical formulation of the idea that these two functions are indeed close.

Theorem 2.3.1: Glivenko-Cantelli

If F is the CDF of a distribution ν and $\widehat{F}_n$ is the CDF of the empirical distribution $\widehat{\nu}_n$ of n samples from ν, then F_n converges to F along the whole number line:

$$\max_{x\in\mathbb{R}} |F(x) - \widehat{F}_n(x)| \to 0 \quad \text{as } n \to \infty,$$

in probability.

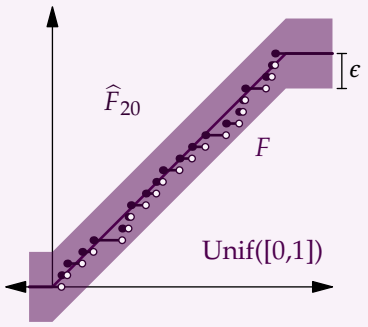
The **Dvoretzky-Kiefer-Wolfowitz inequality** quantifies this result by providing a confidence band.

Theorem 2.3.2: DKW inequality

If $X_1, X_2, \ldots$ are independent random variables with common CDF F, then for all $\epsilon > 0$, we have

$$\mathbb{P}\left(\max_x |F(x) - \widehat{F}_n(x)| \geq \epsilon\right) \leq 2e^{-2n\epsilon^2}.$$

In other words, the probability that the graph of $\widehat{F}_n$ lies in the ϵ-band around F (or vice versa) is at least $1 - 2e^{-2n\epsilon^2}$.



Exercise 2.3.1

Show that the DKW gives a confidence band for F. Specifically, show that if $\epsilon_n = \sqrt{\frac{1}{2n}\log(\frac{2}{\alpha})}$, then with probability at least $1 - \alpha$, we have $|F(x) - F_n(x)| \leq \epsilon$ for all $x \in \mathbb{R}$.

2.3.2 Bootstrapping

Bootstrapping is the use of simulation to approximate the value of the plug-in estimator of a statistical functional which is expressed in terms of independent samples from ν. The key idea is that drawing k samples from $\widehat{\nu}$ is the same as drawing k times with replacement from the list $\{X_1, \ldots, X_n\}$.

Example 2.3.1

Consider the statistical functional $T(\nu)$ = the expected difference between the greatest and least of 10 independent samples from ν. Suppose that 50 samples $X_1, \ldots, X_{50}$ from ν are observed, and that $\widehat{\nu}$ is the associated empirical CDF. Explain how $T(\widehat{\nu})$ may be estimated with arbitrarily small error.

Solution

The value of $T(\widehat{\nu})$ is defined to be the expectation of a distribution that we have instructions for how to sample from. So we sample 10 times with replacement from $X_1, \ldots, X_{50}$, record the largest and smallest of the 10 samples, and record the difference. We repeat B times for some large integer B, and we return the sample mean of these B values.

By the law of large numbers, the result can be made arbitrarily close to $T(\widehat{\nu})$ with arbitrarily high probability by choosing B sufficiently large.

Although Example 2.3.1 might seem a bit contrived, bootstrapping is useful in practice because of a common source of statistical functionals that fit the bootstrap form: *standard errors*.

Example 2.3.2

Suppose that we estimate the median θ of a distribution using the plug-in estimator $\widehat{\theta}$ for 75 observed samples, and we want to produce a confidence interval for θ. Show how to use bootstrapping to estimate the standard error of the estimator.

Solution

By definition, the standard error of $\widehat{\theta}$ is the square root of the variance of the median of 75 independent draws from ν. Therefore, the plug-in estimator of the standard error is the square root of the variance of the median of 75 independent draws from $\widehat{\nu}$. This can be readily simulated. If the samples are stored in a vector `X`, then

```
sd(sapply(1:10^5,function(n) {median(sample(X,75,replace=TRUE))}))
```

returns a very accurate approximation of $T(\widehat{\nu})$.

Exercise 2.3.2

Suppose that ν is the uniform distribution on $[0, 1]$. Generate 75 samples from ν, store them in a vector X, and compute the bootstrap estimate of $T(\widehat{\nu})$. Use Monte Carlo simulation to directly estimate $T(\nu)$. Can the gap between your approximations of $T(\widehat{\nu})$ and $T(\nu)$ be made arbitrarily small by using more bootstrap samples?

2.4 Maximum likelihood estimation

So far we've only had one idea for building an estimator, which is to plug $\widehat{v}$ into the statistical functional. In this section, we'll learn another approach which is quite general and has some compelling properties.

Consider a parametric family $\{f_{\boldsymbol{\theta}}(x) : \boldsymbol{\theta} \in \mathbb{R}^d\}$ of PDFs or PMFs. Given $\mathbf{x} \in \mathbb{R}^n$, the **likelihood** $\mathcal{L}_{\mathbf{x}} : \mathbb{R}^d \to \mathbb{R}$ is defined by

$$\mathcal{L}_{\mathbf{x}}(\boldsymbol{\theta}) = f_{\boldsymbol{\theta}}(x_1) f_{\boldsymbol{\theta}}(x_2) \cdots f_{\boldsymbol{\theta}}(x_n).$$

The idea is that if $\mathbf{X}$ is a vector of n independent samples drawn from $f_{\boldsymbol{\theta}}(x)$, then $\mathcal{L}_{\mathbf{X}}(\boldsymbol{\theta})$ is small or zero when $\boldsymbol{\theta}$ is not in concert with the observed data.

Example 2.4.1

Suppose $x \mapsto f(x;\theta)$ is the density of a uniform random variable on $[0,\theta]$. We observe four samples drawn from this distribution: 1.41, 2.45, 6.12, and 4.9. Find $\mathcal{L}(5)$ and $\mathcal{L}(10^6)$.

Solution

The likelihood at 5 is zero, since $f_5(x_3) = 0$. The likelihood at 10^6 is very small, since $\mathcal{L}(10^6) = (1/10^6)^4 = 10^{-24}$.

We can see from Example 2.4.1 that likelihood has the property that it is zero or small at implausible values of $\boldsymbol{\theta}$, and larger at more reasonable values. Thus we propose the **maximum likelihood estimator**

$$\widehat{\boldsymbol{\theta}}_{\text{MLE}} = \text{argmax}_{\boldsymbol{\theta} \in \mathbb{R}^d} \mathcal{L}_{\mathbf{X}}(\boldsymbol{\theta}).$$

Example 2.4.2

Suppose that $x \mapsto f(x;\mu,\sigma^2)$ is a normal density with mean μ and variance σ^2. Find the maximum likelihood estimator for μ and σ^2.

Solution

The maximum likelihood estimator is the minimizer of the logarithm of the likelihood function, which is

$$-\frac{n}{2}\log 2\pi - n\log\sigma - \frac{nS^2}{2\sigma^2} - \frac{n(\overline{X}-\mu)^2}{2\sigma^2},$$

where $\overline{X}$ and S^2 are the plug-in estimators of the mean and variance, respectively. Differentiating with respect to μ and σ, we find $\mu = \overline{X}$ and $\sigma^2 = S^2$. So the maximum likelihood estimator agrees with the plug-in estimator for μ and σ^2.

MLE enjoys several nice properties: under certain regularity conditions, we have

1. **Consistency**: $\mathbb{E}[(\widehat{\theta}_{\text{MLE}} - \theta)^2] \to 0$ as the number of samples goes to ∞.

2. **Asymptotic normality**: $(\widehat{\theta}_{\text{MLE}} - \theta)/\sqrt{\operatorname{Var}\widehat{\theta}_{\text{MLE}}}$ converges to $\mathcal{N}(0,1)$ as the number of samples goes to ∞.
3. **Asymptotic optimality**: the MSE of the MLE converges to 0 approximately as fast as the MSE of any other consistent estimator.

Potential difficulties with MLE:

1. **Computational difficulties**. It might be difficult to work out where the maximum of the likelihood occurs, either analytically or numerically.
2. **Misspecification**. The MLE may be inaccurate if the distribution of the samples is not in the specified parametric family.
3. **Unbounded likelihood**. If the likelihood function is not bounded, then $\widehat{\theta}_{\text{MLE}}$ is not well-defined.

2.5 Hypothesis testing

Hypothesis testing is a disciplined framework for adjudicating whether observed data do not support a given hypothesis.

Consider an unknown distribution from which we will observe n samples $X_1, \dots X_n$.

1. We state a hypothesis H_0–called the **null hypothesis**–about the distribution.
2. We come up with a **test statistic** T, which is a function of the data $X_1, \dots X_n$, for which we can evaluate the distribution of T assuming the null hypothesis.
3. We give an **alternative hypothesis** H_a under which T is expected to be significantly different from its value under H_0.
4. We give a significance level α (like 5% or 1%), and based on H_a we determine a set of values for T—called the *critical region*—which T would be in with probability at most α under the null hypothesis.
5. **After setting H_0, H_a, α, T, and the critical region**, we run the experiment, evaluate T on the samples we get, and record the result as t_{obs}.
6. If t_{obs} falls in the critical region, we reject the null hypothesis. The corresponding ***p*-value** is defined to be the minimum α-value which would have resulted in rejecting the null hypothesis, with the critical region chosen in the same way*.

Example 2.5.1

Muriel Bristol claims that she can tell by taste whether the tea or the milk was poured into the cup first. She is given eight cups of tea, four poured milk-first and four poured tea-first.

We posit a null hypothesis that she isn't able to discern the pouring method, and an alternative hypothesis that she can tell the difference. How many cups does she have to identify correctly to reject the null hypothesis with 95% confidence?

Solution

Under the null hypothesis, the number of cups identified correctly is 4 with probability $1/\binom{8}{4} \approx 1.4\%$ and at least 3 with probability $17/70 \approx 24\%$. Therefore, at the 5% significance level, only a correct identification of all the cups would give us grounds to reject the null hypothesis. The p-value in that case would be 1.4%.

Failure to reject the null hypothesis is not necessarily evidence *for* the null hypothesis. The **power** of a hypothesis test is the conditional probability of rejecting the null hypothesis given that the alternative hypothesis is true. A p-value may be low either because the null hypothesis is true or because the test has low power.

Definition 2.5.1

The **Wald test** is based on the normal approximation. Consider a null hypothesis $\theta = 0$ and the alternative hypothesis $\theta \neq 0$, and suppose that $\widehat{\theta}$ is approximately normally distributed. The Wald test rejects the null hypothesis at the 5% significance level if $|\widehat{\theta}| > 1.96\,\mathrm{se}(\widehat{\theta})$.

Example 2.5.2

Consider the alternative hypothesis that 8-cylinder engines have lower fuel economy than 6-cylinder engines (with null hypothesis that they are the same). Apply the Wald test, using the `mtcars` dataset available in R.

Solution

We frame the problem as a question about whether the *difference in means* between the distribution of 8-cylinder `mpg` values and the distribution of 6-cylinder `mpg` values is zero. We use the difference between the sample means $\overline{X}$ and $\overline{Y}$ of the two populations as an estimator of the difference in means. If we think of the records in the data frame as independent, then $\overline{X}$ and $\overline{Y}$ are independent. Since each is approximately normally distributed by the central limit theorem, their difference is therefore also approximately normal.*

*...because linear combinations of independent normal random variables are normal.

So, let's calculate the sample mean and sample variance for the 8-cylinder cars and for the 6-cylinder cars.

```
library(tidyverse)

stats <- mtcars %>%
  group_by(cyl) %>%
  filter(cyl %in% c(6,8)) %>%
  summarise(m = mean(mpg), S2 = var(mpg), n = n(), se = S2/sqrt(n))
```

Given that the distribution of 8-cylinder `mpg` values has variance σ^2_{eight}, the variance of the sample mean $\overline{X}$ is $\sigma^2_{\text{eight}}/n_{\text{eight}}$, where n_{eight} is the number of 8-cylinder vehicles (and similarly for $\overline{Y}$). Therefore, we estimate the variance of the difference in sample means as

$$\mathrm{Var}(\overline{X} - \overline{Y}) = \mathrm{Var}(\overline{X}) + \mathrm{Var}(\overline{Y}) = \sigma^2_{\text{eight}}/n_{\text{eight}} + \sigma^2_{\text{six}}/n_{\text{six}}.$$

Under the null hypothesis, therefore, $\overline{X} - \overline{Y}$ has mean zero and standard error

$\sqrt{\sigma^2_{\text{eight}}/n_{\text{eight}} + \sigma^2_{\text{six}}/n_{\text{six}}}$. We therefore reject the null hypothesis with 95% confidence if the value of $\overline{X} - \overline{Y}$ divided by its estimated standard error exceeds 1.96. We find that

```
z <- (stats$m[1] - stats$m[2]) / sqrt(sum(stats$se^2))
```

returns 2.41, so we do reject the null hypothesis at the 95% confidence level. The p-value of this test is `1 - pnorm(z)` $= 0.79\%$.

The following test is more flexible than the Wald test, since it doesn't rely on the normal approximation. It's based on a simple idea: if there's no difference in labels, the data shouldn't look very different if we shuffle them around.

Definition 2.5.2

The **random permutation test** is applicable when the null hypothesis is that two distributions are the same.

1. We compute the difference between the sample means for the two groups.
2. We randomly re-assign the group labels and compute the resulting sample mean differences. Repeat many times.
3. We check where the original difference falls in the sorted list of re-sampled differences.

Example 2.5.3

Suppose the heights of the Romero sons are 72, 69, 68, and 66 inches, and the heights of the Larsen sons are 70, 65, and 64 inches. Consider the null hypothesis that the height distributions for the two families are the same, with the alternative hypothesis that they are not. Determine whether a random permutation test applied to the absolute sample mean difference rejects the null hypothesis at significance level $\alpha = 5\%$.

Solution

We find that the absolute sample mean difference of about 2.4 inches is larger than only about 68% of the mean differences obtained by resampling many times.

```
set.seed(123)
romero <- c(72, 69, 68, 66)
larsen <- c(70, 65, 64)
actual.diff <- abs(mean(romero) - mean(larsen))

resample.diff <- function(n) {
  shuffled <- sample(c(romero,larsen))
  abs(mean(shuffled[1:4]) - mean(shuffled[5:7]))
}

first(which(sort(sapply(1:10000,resample.diff)) >= actual.diff))
```

Since 68% < 95%, we retain the null hypothesis.

2.5.1 Multiple hypothesis testing

* xkcd.com/882

If we conduct many hypothesis tests, then the probability of obtaining some false rejections is high*. This is called the **multiple testing problem**.

The **Bonferroni method** is to reject the null hypothesis only for those tests whose p-values are less than α divided by the number of hypothesis tests being run. This ensures that the probability of having even one false rejection is less than α, so it is very conservative.

Example 2.5.4

Suppose that 10 different genes are tested to determine whether they have an affect on heart disease. The 10 p-values resulting from these hypothesis tests are (rounded to the nearest hundredth of a percent):

$$0.89\%, 2.71\%, 9.11\%, 2.18\%, 9.17\%, 7.48\%, 5.0\%, 2.02\%, 5.22\%, 9.46\%$$

Which results are reported as significant at the 5% level, according to the Bonferroni method?

Solution

At the 5% level, only p values less than $5\%/10 = 0.5\%$ are reported as significant (since we ran ten hypothesis tests). Since none of the p values are below 0.5%, none of the genes will be considered significant.