R 语言模型部署实战

徐静

2018-08-06

Contents

序言

我们的模型不能只停留在线下的分析报告中，训练好的 R 模型如何应用到生产环境？目前针对于 R 语言的模型生产环境应用的方式有很多，比如用其他语言去调用，Java，Python 等语言均可方便的调用 R 脚本；生成 PMML 文件，目前 R 中主流的一些 R 模型均支持 PMML 比如 xgboost,lightGBM 等，其他语言不需要调用 R 脚本只需调用统一的 PMML 文件就可以；还有就是 Web 端的部署，比如可以做成 REST API 供其他语言调用，或直接做成 web 应用供其他用户访问，本书主要针对于 R 语言模型的 Web 端的部署。过程中，我们会先后介绍 httpuv,opencpu,plumber, jug,fiery,Rserve,RestRserve, 等一些和模型线上化部署相关的 R 包 (当然 shiny 也可以，但他不是我们本书的重点)，最后会介绍 mailR 和 Rweixin 两个 R 和邮件与微信通信的 R 包，用于线上化部署的监测。当然会有其他的线上化部署方式。

欢迎进入 R 模型线上化部署的海洋！

关于我

徐静：

硕士研究生, 目前的研究兴趣主要包括：数理统计，统计机器学习，深度学习，网络爬虫，前端可视化，R 语言和 Python 语言的超级粉丝，多个 R 包和 Python 模块的作者，现在正逐步向 Java 迁移。

Graduate students,the current research interests include: mathematical statistics, statistical machine learning, deep learning, web crawler, front-end visualization. He is a super fan of R and Python, and the author of several R packages and Python modules, and now gradually migrating to Java.

Chapter 1

httpuv

在 httpuv 的官网中, 有这么一段描述：

> Allows R code to listen for and interact with HTTP and WebSocket clients, so you can serve web traffic directly out of your R process. Implementation is based on libuv and http-parser.
>
> This is a low-level library that provides little more than network I/O and implementations of the HTTP and WebSocket protocols. For an easy way to create web applications, try Shiny instead.

我们可以通过 httpuv 搭建一个访问 R 模型的 web API, 但可能这不是最好的。

本部分我们首先介绍官方提供的一些方法，然后解析官方提供的演示 Demo, 从而达到熟练使用 httpuv 的目的。

1.1 方法介绍

下面我们解析一下 httpuv 官方提供的一些调用方法，并演示一些调用方法对应的实例

1. 使用 URI 编码/解码以与 Web 浏览器相同的方式对字符串进行编码/解码。

```
encodeURI(value)
encodeURIComponent(value)
decodeURI(value)
decodeURIComponent(value)
```

参数列表

- value 用于编码和解码的字符向量，UTF-8 字符编码

```
library(httpuv)
value <- "https://baidu.com/中国;?/"
encodeURI(value)
```

```
## [1] "https://baidu.com/%D6%D0%B9%FA;?/"
```

```
encodeURIComponent(value)
```

```
## [1] "https%3A%2F%2Fbaidu.com%2F%D6%D0%B9%FA%3B%3F%2F"
```

```
decodeURI(value)
```

```
## [1] "https://baidu.com/中国;?/"
```

```
decodeURIComponent(value)
```

```
## [1] "https://baidu.com/中国;?/"
```

注意：encodeURI 与 encodeURIComponent 是不一样的因为前者不对特殊字符: ;,/?:@&=+$ 等进行 encode

2. 中断 httpuv 运行的环路

```
interrupt()
```

3. 检查 ip 地址的类型是 ipv4 还是 ipv6

```
ipFamily(ip)
```

参数列表

- ip 一个代表 IP 地址的字符串
- 返回值的意义：如果是 IPv4 返回 4，如果是 IPv6 返回 6，如果不是 IP 地址返回 -1

```
ipFamily("127.0.0.1") # 4
```

```
## [1] 4
```

```
ipFamily("500.0.0.500") # -1
```

```
## [1] -1
```

```
ipFamily("500.0.0.500") # -1
```

```
## [1] -1
```

```
ipFamily("::") # 6
```

```
## [1] 6
```

```
ipFamily("::1") # 6
```

```
## [1] 6
```

```
ipFamily("fe80::1ff:fe23:4567:890a") # 6
```

```
## [1] 6
```

3. 将原始向量转换为 BASE64 编码字符串

```
rawToBase64(x)
```

参数列表

- x 原始向量

```
set.seed(100)
result <- rawToBase64(as.raw(runif(19, min=0, max=256)))
#stopifnot(identical(result, "TkGNDnd7z16LK5/hR2bDqzRbXA=="))
result
```

```
## [1] "TkGNDnd7z16LK5/hR2bDqzRbXA=="
```

4. 运行一个 server

```
runServer(host, port, app, interruptIntervalMs = NULL)
```

参数列表

- host IPv4 地址, 或是 "0.0.0.0" 监听所有的 IP
- port 端口号
- app 一个定义应用的函数集合
- interruptIntervalMs 该参数不提倡使用，1.3.5 版本后废除

```
app <- list(call = function(req){
  list(status=200L,
       headers = list(
         'Content-Type' = 'text/html'
       ),
       body = "HelloWorld!")
})

runServer("0.0.0.0", 5000,app)
```

5. 过程请求

处理 HTTP 请求和 WebSocket 消息。如果 R 的调用堆栈上没有任何东西，如果 R 是在命令提示符下闲置，不必调用此函数，因为请求将自动处理。但是，如果 R 正在执行代码，则请求将不被处理。要么调用栈是空的，要么调用这个函数 (或者，调用 run_now())。

```
service(timeoutMs = ifelse(interactive(), 100, 1000))
```

参数列表

- timeoutMs 返回之前运行的毫秒数。

6. 创建 HTTP/WebSocket 后台服务器（弃用）

```
startDaemonizedServer(host, port, app)
```

7. 创建 HTTP/WebSocket 服务器

```
startServer(host, port, app)
startPipeServer(name, mask, app)
```

参数列表

- host ip 地址
- port 端口号
- app 一个定义应用的函数集

```
app <- list(
  call = function(req) {
    list(
      status = 200L,
      headers = list(
        'Content-Type' = 'text/html'
        ),
      body = "Hello world!"
    )
    }
  )
handle <- startServer("0.0.0.0", 5000,app)

# 此服务器的句柄，可以传递给StestServer以关闭服务器。
stopServer(handle)
```

8. 停止所有应用

```
stopAllServers()
```

9. 在 UNIX 环境中停止运行的后台服务器（弃用）

```
stopDaemonizedServer(handle)
```

10. 停止一个服务

```
stopServer(handle)
```

1.2 例子演示

1. json-server

```
# Connect to this using websockets on port 9454
# Client sends to server in the format of {"data":[1,2,3]}
# The websocket server returns the standard deviation of the sent array
library(jsonlite)
library(httpuv)

# Server
app <- list(
  onWSOpen = function(ws) {
    ws$onMessage(function(binary, message) {
      # Decodes message from client
      message <- fromJSON(message)
      # Sends message to client
      ws$send(
        # JSON encode the message
        toJSON(
          # Returns standard deviation for message
          sd(message$data)
        )
      )
    })
  }
)
runServer("0.0.0.0", 9454, app, 250)
```

2.echo

```
library(httpuv)

app <- list(
  call = function(req) {
    wsUrl = paste(sep='',
                  '"',
                  "ws://",
                  ifelse(is.null(req$HTTP_HOST), req$SERVER_NAME, req$HTTP_HOST),
                  '"')

    list(
      status = 200L,
      headers = list(
        'Content-Type' = 'text/html'
      ),
      body = paste(
        sep = "\r\n",
```

```
        "<!DOCTYPE html>",
        "<html>",
        "<head>",
        '<style type="text/css">',
        'body { font-family: Helvetica; }',
        'pre { margin: 0 }',
        '</style>',
        "<script>",
        sprintf("var ws = new WebSocket(%s);", wsUrl),
        "ws.onmessage = function(msg) {",
        '  var msgDiv = document.createElement("pre");',
        '  msgDiv.innerHTML = msg.data.replace(/&/g, "&amp;").replace(/\\</g, "&lt;");',
        '  document.getElementById("output").appendChild(msgDiv);',
        "}",
        "function sendInput() {",
        "  var input = document.getElementById('input');",
        "  ws.send(input.value);",
        "  input.value = '';",
        "}",
        "</script>",
        "</head>",
        "<body>",
        '<h3>Send Message</h3>',
        '<form action="" onsubmit="sendInput(); return false">',
        '<input type="text" id="input"/>',
        '<h3>Received</h3>',
        '<div id="output"/>',
        '</form>',
        "</body>",
        "</html>"
      )
    )
  },
  onWSOpen = function(ws) {
    ws$onMessage(function(binary, message) {
      ws$send(message)
    })
  }
)

browseURL("http://localhost:9454/")
runServer("0.0.0.0", 9454, app, 250)
```

3.deamon-echo

```
library(httpuv)

.lastMessage <- NULL

app <- list(
  call = function(req) {
    wsUrl = paste(sep='',
                  '"',
```

```
                  "ws://",
                  ifelse(is.null(req$HTTP_HOST), req$SERVER_NAME, req$HTTP_HOST),
                  '"')

    list(
      status = 200L,
      headers = list(
        'Content-Type' = 'text/html'
      ),
      body = paste(
        sep = "\r\n",
        "<!DOCTYPE html>",
        "<html>",
        "<head>",
        '<style type="text/css">',
        'body { font-family: Helvetica; }',
        'pre { margin: 0 }',
        '</style>',
        "<script>",
        sprintf("var ws = new WebSocket(%s);", wsUrl),
        "ws.onmessage = function(msg) {",
        '  var msgDiv = document.createElement("pre");',
        '  msgDiv.innerHTML = msg.data.replace(/&/g, "&amp;").replace(/\\</g, "&lt;");',
        '  document.getElementById("output").appendChild(msgDiv);',
        "}",
        "function sendInput() {",
        "  var input = document.getElementById('input');",
        "  ws.send(input.value);",
        "  input.value = '';",
        "}",
        "</script>",
        "</head>",
        "<body>",
        '<h3>Send Message</h3>',
        '<form action="" onsubmit="sendInput(); return false">',
        '<input type="text" id="input"/>',
        '<h3>Received</h3>',
        '<div id="output"/>',
        '</form>',
        "</body>",
        "</html>"
      )
    )
  },
  onWSOpen = function(ws) {
    ws$onMessage(function(binary, message) {
      .lastMessage <<- message
      ws$send(message)
    })
  }
)

server <- startDaemonizedServer("0.0.0.0", 9454, app)
```

```
# check the value of .lastMessage after echoing to check it is being updated

# call this after done
#stopDaemonizedServer(server)
```

4.

```
library(httpuv)
app = list(call = function(req){
  # 获取POST的参数
  postdata = req$rook.input$read_lines()
  qs = httr:::parse_query(gsub("^\\?", "", postdata))
  dat = jsonlite::fromJSON(qs$jsonDat)
  print(dat)
  # 计算返回结果
  r = 0.3 + 0.1 * dat$v1 - 0.2 * dat$v2 + 0.1 * dat$v3
  output = jsonlite::toJSON(list(message = 'suceess', result = r), auto_unbox = T)
  res = list(status = 200L, headers = list('Content-Type' = 'application/json'), body = o
                                                    return(res)
})
                                                    # 启动服务
                                                    server = startServer("0.0.0.0", 1124L, app = a
                                                    while(TRUE) {
                                                    service()
                                                    Sys.sleep(0.001)
                                                    }
                                                    # stopServer(server)
```

```
RCurl::postForm('127.0.0.1:1124',
style = 'post',
.params = list(jsonDat = '{"v1":1,"v2":2,"v3":3}')
)
```

httpuv 是相对比较底层的包，熟练使用需要掌握前端知识，并且需要用到 RCurl，httr 相关爬虫包的一些知识去处理。本人不推荐这种方式进行模型的部署。

Chapter 2

opencpu

Chapter 3

plumber

Chapter 4

jug

Chapter 5

fiery

Chapter 6

Rserve

Chapter 7

RestRserve

Chapter 8

mailR

mailR 是一个比较小的包，主要解决的问题是 R 与邮件发送的问题, 该包就一个方法：send.mail() 方法调用方式为：

```
send.mail(from, to, subject = "", body = "", encoding = "iso-8859-1",
html = FALSE, inline = FALSE, smtp = list(), authenticate = FALSE,
send = TRUE, attach.files = NULL, debug = FALSE, ...)
```

参数列表：

- from 有效的发送者的邮箱
- to 目标接收的邮箱
- subject 邮箱主题
- body 邮件体
- encoding 邮件内容字符编码支持包括 iso-8859-1 (default), utf-8, us-ascii, and koi8-r
- html bool 值，是否把邮箱体解析成 html
- inline 布尔值，HTML 文件中的图像是否应该嵌入内联。
- smtp lsit 类型，链接邮箱的 smtp
- authenticate 一个布尔变量，用于指示是否需要授权连接到 SMTP 服务器。如果设置为 true，请参阅 SMTP 参数所需参数的详细信息。发送一个布尔值，指示电子邮件是否应该在函数的末尾发送。（默认行为）。如果设置为 false，函数将电子邮件对象返回给父环境。
- attach.files 链接到文件的文件系统中路径的字符向量或有效 URL 到附加到电子邮件（详见更多信息附加 URL）
- debug bool 值，是否查看 debug 的真实细节
- ... Optional arguments to be passed related to file attachments. See details for more

Example1:

```
mailR::send.mail(
  from = 'sender@tuandai.com', # 发送人
  to = 'sendee@tuandai.com', # 接收人
  cc = 'carboncopy@tuandai.com', # 抄送人
  subject = ' 邮件标题',
  body = as.character(
    '<div style = "color:red">邮件正文，可以为HTML格式</div>'
  ),
  attach.files = NULL, # 附件的路径
```

```
  encoding = "utf-8",
  smtp = list(
    host.name = 'smtp.exmail.qq.com', # 邮件服务器IP地址
    port = 465, # 邮件服务器端口
    user.name = 'senderName', # 发送人名称
    passwd = 'yourpassword', # 密码
    ssl = T),
  html = T, inline = T, authenticate = T, send = T, debug = F
)
```

Example2:

```
send.mail(from = "sender@gmail.com",
          to = c("Recipient 1 <recipient1@gmail.com>", "recipient2@gmail.com"),
          cc = c("CC Recipient <cc.recipient@gmail.com>"),
          bcc = c("BCC Recipient <bcc.recipient@gmail.com>"),
          subject="Subject of the email",
          body = "Body of the email",
          smtp = list(host.name = "aspmx.l.google.com", port = 25),
          authenticate = FALSE,
          send = TRUE)
```

Example3:

```
send.mail(from = "sender@gmail.com",
          to = c("recipient1@gmail.com", "recipient2@gmail.com"),
          subject = "Subject of the email",
          body = "Body of the email",
          smtp = list(host.name = "smtp.gmail.com", port = 465, user.name = "gmail_userna
          authenticate = TRUE,
          send = TRUE)
```

Example4:

```
email <- send.mail(from = "Sender Name <sender@gmail.com>",
                   to = "recipient@gmail.com",
                   subject = "A quote from Gandhi",
                   body = "In Hindi :  □□□□ □□ □□□□□ □□□□ □□□□ □□□□□□□ □□ □□□□□ □□□
                   English translation: An ounce of practice is worth more than tons of p
                   encoding = "utf-8",
                   smtp = list(host.name = "smtp.gmail.com", port = 465, user.name = "gma
                 authenticate = TRUE,
                   send = TRUE)
```

Example5:

```
send.mail(from = "sender@gmail.com",
          to = c("recipient1@gmail.com", "recipient2@gmail.com"),
          subject = "Subject of the email",
          body = "Body of the email",
          smtp = list(host.name = "smtp.gmail.com", port = 465, user.name = "gmail_userna
          authenticate = TRUE,
          send = TRUE,
          attach.files = c("./download.log", "upload.log"),
          file.names = c("Download log", "Upload log"), # optional parameter
          file.descriptions = c("Description for download log", "Description for upload l
```

Example6:

```
send.mail(from = "sender@gmail.com",
          to = c("recipient1@gmail.com", "recipient2@gmail.com"),
          subject = "Subject of the email",
          body = "<html>The apache logo - <img src=\"http://www.apache.org/images/asf_log
          html = TRUE,
          smtp = list(host.name = "smtp.gmail.com", port = 465, user.name = "gmail_userna
          authenticate = TRUE,
          send = TRUE)
```

Example7:

```
send.mail(from = "sender@gmail.com",
          to = c("recipient1@gmail.com", "recipient2@gmail.com"),
          subject = "Subject of the email",
          body = "path.to.local.html.file",
          html = TRUE,
          inline = TRUE,
          smtp = list(host.name = "smtp.gmail.com", port = 465, user.name = "gmail_userna
          authenticate = TRUE,
          send = TRUE)
```

Chapter 9

Rweixin

Chapter 10

参考文献