

深度学习与生成式 AI——R torch 核心算法与实战

刘思喆

2026-03-22

目录

欢迎	1
序言	2
I. 基石篇—基础概念	4
1. 深度学习与 R 生态	6
1.1. 深度学习简史	6
1.2. R torch 的架构底座	8
1.3. 第一个 torch 程序	9
1.3.1. 安装环境	9
1.3.2. 张量运算	10
2. 张量思维	12
2.1. 从标量到张量	12
2.2. 张量的核心操作	14
2.3. 形状与维度操作	16
2.4. 线性代数运算实战	17
2.4.1. 基础算数	17
2.4.2. 矩阵乘法	18
2.4.3. 广播机制	19
2.4.4. 聚合操作	20
2.4.5. 爱因斯坦求和约定	21
2.5. 计算设备切换	21
3. 自动微分	23
3.1. 从解析解到数值解	23
3.1.1. 殊途同归	23
3.1.2. 解析解	24

3.1.3. 数值解	25
3.2. 理解计算图	28
3.3. 梯度的三部曲	31
3.3.1. 语法和使用细节	31
3.3.2. 上下文管理	33
3.3.3. 张量视角的模型构建	35
3.4. 引入优化器	36
II. 核心篇—神经网络	38
4. 多层感知机	40
4.1. 非线性能力的引入	40
4.1.1. 交互效应	40
4.1.2. 线性坍塌	41
4.1.3. 激活函数三巨头	43
4.1.4. 函数式 vs 模块式	44
4.2. 多层感知机与模块化	45
4.2.1. 什么是多层感知机	45
4.2.2. <code>nn_module</code>	46
4.3. 损失函数	47
4.3.1. 均方误差(MSE)	47
4.3.2. 交叉熵损失	48
4.4. Ames 房价预测	49
5. 优化与训练动力学	55
5.1. 再看反向传播	55
5.2. 梯度下降演进和实现	58
5.2.1. 批量大小的权衡	58
5.2.2. 冲量和自适应算法	59
5.2.3. 三种优化器的对比	62
5.2.4. State Dict (状态字典)	64
5.3. 学习率调度器	65
5.4. 诊断和调试	67
5.5. 混合精度训练 (AMP)	68
5.5.1. 半精度与单精度	68
5.5.2. 自动混排	69

5.5.3. 代码实现	69
6. 工程化实验	71
6.1. 构建高效数据管道	71
6.1.1. R6 与 Dataset 封装	71
6.1.2. 惰性加载	72
6.1.3. 批次与编排	74
6.1.4. 处理变长数据	74
6.1.5. 多进程加速	75
6.2. 偏差和方差的博弈	76
6.2.1. 数据划分原则	76
6.2.2. 偏差-方差权衡	77
6.3. 正则化策略	79
6.3.1. 权重衰减	79
6.3.2. Dropout	81
6.3.3. 批归一化	81
6.4. 拥抱 luz 框架	84
6.4.1. 什么是 luz?	84
6.4.2. 优雅的工作流	85
6.4.3. 引入正则化	89
6.4.4. 回调函数	91
6.5. 超参数调优	93
6.5.1. 完美的学习率	93
6.5.2. 搜索策略	95
6.5.3. 基于 luz 的调优	96
III. 进阶篇—现代网络	100
7. 卷积网络	102
7.1. 应用场景	102
7.2. 卷积	105
7.2.1. 滑动与点积	105
7.2.2. 输出形状的变化	107
7.2.3. 处理多通道	108
7.3. 池化层和归一化	109
7.3.1. 归一化层	110

7.3.2. 激活函数	111
7.4. 从零搭建卷积网络	112
7.4.1. 网络结构实现	112
7.4.2. 参数量剖析	113
7.4.3. 学习到了什么	114
7.5. 模块化的经典架构	117
7.5.1. VGG 块	117
7.5.2. 残差网络 (ResNet)	119
7.5.3. 全局平均池化 (GAP)	121
7.6. 图像处理流	123
7.6.1. 维度转换	123
7.6.2. Torchvision 管道与变换	124
7.7. 迁移学习与混合模型	126
7.7.1. 冻结骨架策略	126
7.7.2. 渐进式微调	128
7.7.3. 混合模型	130
7.8. 一维卷积 (1D-CNN)	131
7.8.1. 架构设计	133
8. 循环网络	135
8.1. 序列建模与 RNN	135
8.1.1. RNN 架构及定义	136
8.1.2. 按时间展开	136
8.1.3. 代码实现	137
8.1.4. 糟糕的梯度问题	139
8.2. LSTM 与 GRU	141
8.2.1. LSTM	142
8.2.2. GRU	145
8.2.3. 变体结构	146
8.2.4. <code>nn_lstm</code> 详解	146
8.3. 连续型序列建模	147
8.3.1. 数据预处理	149
8.3.2. 模型定义和训练	151
8.4. 离散型序列处理	153
8.4.1. Embedding 层	154
8.4.2. 文本流水线	155
8.4.3. 变长序列处理	157

8.5. IMDB 影评情感分析	159
8.5.1. 数据预处理	159
8.5.2. 模型定义和训练	162
8.5.3. Fine-tuning	165
8.6. Seq2Seq 雏形	169
9. Transformer	172
9.1. 注意力机制	172
9.1.1. 物理意义	173
9.1.2. 缩放点积注意力	174
9.1.3. 多头注意力	175
9.2. 完全体架构	178
9.2.1. 失败的实验	178
9.2.2. 位置编码	180
9.2.3. Add & Norm 与 FFN	181
9.2.4. 实现完全体	182
9.3. 重构时序预测模型	184
9.3.1. 原生代码实现	185
9.3.2. 注意力在关注什么	187
9.4. 处理 Data frame	192
9.4.1. 特征的序列化与独立投影	193
9.4.2. 引入 [CLS] Token	193
9.4.3. 个体层面的特征归因	195
9.5. 生成式 Transformer	197
9.5.1. 数据预处理	197
9.5.2. 模型架构	199
9.5.3. 训练过程	203
9.5.4. 推理生成	204
9.5.5. 大语言模型	206
9.6. 微调技术 (PEFT/LoRA)	207
9.6.1. LoRA 的数学之美	208
9.6.2. 工程实现	208
9.6.3. 手写 LoRA 注入网络	209
9.6.4. 执行微调	212
10. 深度推荐模型	214
10.1. 数据和预处理	214

10.2. 协同过滤的基线	216
10.3. 矩阵分解	217
10.3.1. 模型的结构	218
10.3.2. 张量切片	219
10.4. 双塔架构	220
10.4.1. 结构解析	220
10.4.2. 引入特征的双塔	221
10.4.3. 部署和表征导出	224
10.5. 序列推荐和自回归	225
10.5.1. 两大主流架构	226
10.5.2. SASRec 代码解析	227
11. 图神经网络	229
11.1. SVD 到图卷积网络	229
11.2. 代码实现	233
11.3. 增加特征矩阵	236
11.3.1. 加工图数据	236
11.3.2. 带特征的 GCN	238
11.3.3. 模型训练	239
11.4. 图注意力网络	241
11.4.1. GAT 的原理	241
11.4.2. 代码实现	243
11.4.3. 异配性可视化	246
11.5. LightGCN	247
11.5.1. 思想和设计哲学	247
11.5.2. 代码实现	248
11.5.3. 工程应用	250
11.6. 分子水溶性预测	251
11.6.1. 分子图的特征工程	251
11.6.2. 模型结构	253
11.6.3. 训练与效果	254

IV. 前沿篇—生成模型	256
12. 变分自编码器	258
12.1. 从压缩到生成	258
12.1.1. AE 的局限	258
12.1.2. 从点到分布	259
12.1.3. 重建损失与 KL 散度	259
12.2. 构建 VAE 模型	261
12.2.1. VAE 框架	262
12.2.2. 训练模型	264
12.2.3. 生成新图片	265
12.2.4. 聚类和压缩	267
12.3. 时序异常检测	271
12.3.1. 心电图信号	271
12.3.2. 建模和训练	272
12.3.3. 推理和业务解读	273
13. GAN	276
13.1. 什么是 GAN?	276
13.1.1. 分布的变换	277
13.1.2. 判别函数和博弈	277
13.2. GAN 的数学基础	280
13.2.1. 极小极大博弈	281
13.2.2. 训练不稳定的根源	282
13.3. 从 GAN 到 WGAN	283
13.3.1. Wasserstein 距离	284
13.3.2. 距离的对偶形式	286
13.3.3. 强制 Lipschitz 约束	289
13.4. 动漫头像的生成	291
13.4.1. 参数与数据加载	291
13.4.2. 模型定义	293
13.4.3. 训练循环	296
13.5. GAN 演进方向	299
14. 扩散模型	300
14.1. 熵增与时间倒流	300
14.2. 核心公式	301

14.3. 从零构建 Diffusion	302
14.3.1. 调度器和加噪	303
14.3.2. 搭建神经网络	304
14.3.3. 模型训练	305
14.3.4. 逆向采样	307
14.4. U-Net 图像生成	309
14.4.1. 什么是 U-Net	309
14.4.2. 基础块和时间注入	311
14.4.3. 扩散调度器	311
14.4.4. SimpleUNet 主体	312
14.4.5. 训练过程和逆采样	314
14.5. 采样与条件控制	315
14.5.1. 条件生成原理	316
14.5.2. 实现逻辑	318
14.5.3. 从 Class 到 Text	319
V. 工程篇—模型到产品	320
15. 解释性与诊断	322
15.1. 打开黑盒	322
15.2. 错误分析	323
15.3. 宏观机制分析	326
15.3.1. 置换重要性	326
15.3.2. 局部依赖图	327
15.4. 特征贡献	328
15.4.1. Break Down	329
15.4.2. SHAP 值	331
15.5. 视觉解释	332
15.5.1. 实现 Grad-CAM	333
15.6. 文本解释	336
15.6.1. 原理简述	336
15.6.2. 代码实现	337
16. 生产部署	340
16.1. 模型持久化	340
16.1.1. torch_save	340

16.1.2. ONNX 标准	341
16.1.3. TorchScript (JIT)	342
16.2. 脚本到微服务	342
16.3. 容器化 (Docker)	346
16.3.1. 编写 Dockerfile	346
16.3.2. 一键部署	347
16.4. Shiny App	349
16.4.1. 构建 UI	350
16.4.2. 定义 Server 端	351
附录	353
A. 关于数据	353
B. 参考文献	356

欢迎

“统计学是发现真相的科学,而深度学习是逼近真相的艺术。”

如果你曾沉浸在 **Tidyverse** 的优雅中,曾惊叹于 **ggplot2** 的表现力,却在面对 Python 的环境配置和异步生态时感到迟疑,那么这本书就是为你准备的。

在这里,我们不只是在调用 API,而是在用 R 的思维重构智能。我们将拆解复杂的张量流,通过 **torch** 的原生动力,在 R 的环境中复刻那些改变世界的算法。这不只是一次技术的跨越,更是一场关于“创造力”的回归。

在这段充满挑战与乐趣的旅程中,你将:

- 从张量运算、自动微分,到概率论、信息论与最优化理论,我们提供清晰严谨的数学推导,确保你不仅知其然,更知其所以然。
- 学习模型训练的艺术、正则化的智慧、超参数调优的策略,最终走向模型部署、高性能计算与可解释性 AI,完成从实验到产品的跨越。
- 从前馈网络、卷积神经网络、循环神经网络,到变革性的 Transformer、深度推荐模型、图神经网络、到变分自编码器、生成对抗网络、扩散模型,你将亲手用 torch 实现它们,并理解其背后的设计哲学。
- 在 torch 构建的模型之上,你仍可无缝调用 R 强大的数据整理、可视化及统计建模生态,实现端到端、可复现的深度学习 workflow。

无论你是希望将深度学习纳入工具箱的数据科学家和统计学家,还是渴望在 R 生态中探索现代 AI 边界的开发者与研究者,本书都将为你提供一条系统、清晰且坚实的学习路径。

我们相信,深度学习的魅力不仅在于理论与实践的共舞,更在于严谨数学与工程直觉的交融。现在,让我们敲下第一行张量操作的代码,共同开启用 R 驾驭深度学习的新篇章。

祝你旅程愉快,探索无限!

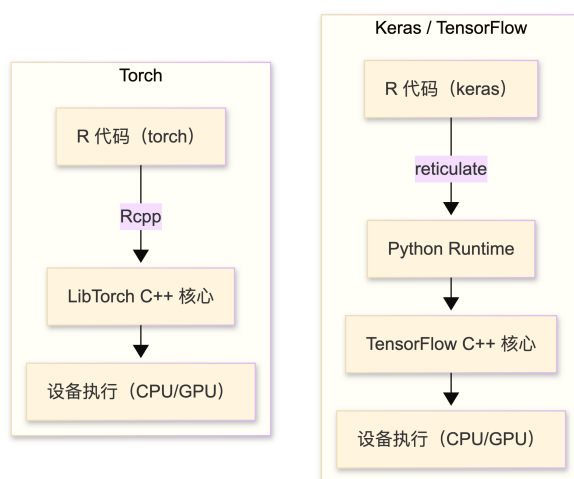
序言

当深度学习的浪潮席卷全球时, R 语言社区曾处于一个微妙而略显尴尬的境地。作为统计计算与数据科学的先驱, R 在数据操纵、可视化及统计建模上拥有得天独厚的优势。其优雅的向量化语法和以 **tidyverse** 为代表的现代生态, 使其成为探索性数据分析的标杆。

然而, 长期以来 R 开发者在构建深度神经网络时, 始终面临着“跨语言”的尴尬。无论是早期的 **mxnet** 还是后来的 **tensorflow** 与 **keras** 接口, 本质上多是 Python 库的 R 语言封装。

为了跑通模型, 首先要跨越“环境地狱”: 配置 Python、管理 Conda 环境、处理 **reticulate** 的调用报错。这种复杂的工具链脱节, 不仅抬高了技术门槛, 更在写下首行核心代码前, 就劝退了许多追求环境纯粹的数据科学家。R 优雅的数据管道与深度学习工作流之间, 始终存在着一层难以消解的隔阂。

真正的转折点源于 **torch** (Falbel 和 Luraschi 2025) 包的横空出世。它不是一个简单平庸的语言绑定, 而是一个为 R 语言习惯量身定制, 直接构建在 C++ 的 LibTorch 之上的原生框架。



本书选择 **torch** 作为核心工具, 正是看重其在“算法透明度”与“工程直觉”之间的完美平衡:

- 运行 **install.packages("torch")** 得到的是一套独立、原生的解决方案。告别 Python 版本冲突与双运行时环境, 让模型从开发到生产的链路变得前所未有的清爽。
- **torch** 坚持使用 R 的 1 起始 (1-based) 索引。这意味着当你面对数学公式中的 x_1 时, 代码里就是 **x[1]** 而非反直觉的 **x[0]**。这种微小的一致性, 极大地降低了将公式转化为算法时的认知负荷

- 与高度封装、隐去细节的高级框架不同, **torch** 鼓励通过 **nn_module** 手动编写 **forward** 函数。配合动态计算图(Dynamic Graph), 你可以像操作原生矩阵一样随时暂停、检查张量维度、观察梯度流动。这让深度学习不再是不可捉摸的黑盒, 而是触手可及的数值演算。

不可否认, 作为一个快速成长的生态, 当前的 R **torch** 在预训练模型库(如 Hugging Face)的生态对接上, 与 Python 仍有差距。但这恰恰使其成为学习深度学习本质的绝佳载体——它迫使我们回归原理, 而非仅仅充当 API 的搬运工。

本书致力于将这种“母语级”的深度学习体验带给每一位 R 用户。你将看到如何用熟悉的 **%>%** 管道连接数据流, 用 **ggplot2** 实时追踪损失曲线, 并在同一个 Quarto 环境中完成从数学推导、代码实现到工程部署的闭环。

深度学习正在重塑现实的边界, 而 **torch** 正在定义 R 语言在人工智能时代的新疆界。愿以此书, 送给那些既追求数学之美、又严谨对待工程实践的同行者。

旅程拉开序幕, 让我们开始。

刘思喆, 2026 年 3 月, 北京