

极客大学机器学习训练营

机器学习动手实战

王然

众微科技 AI Lab 负责人

二〇二一年二月一日

- 1 概览
- 2 逻辑回归的实现
- 3 Proximal Methods 和实现
- 4 补充习题
- 5 参考文献

- 1 概览
- 2 逻辑回归的实现
- 3 Proximal Methods 和实现
- 4 补充习题
- 5 参考文献

- ▶ 上次课程里我们讲了如何使用概率模型推导损失函数，并解释了如何对损失函数进行求导
- ▶ 在这一讲中，我们将会讲如何将之前的内容转化为真实的生产力，及推导之前说过的模型

- ▶ 这一章是我们的所有章节中，对于能力培养最核心的章节
- ▶ 在这一章当中，我们将会将第二章到第四章的所有核心知识点进行串联
- ▶ 这一章的难度，大概相当于很多学员口中野鸡大学的阿姆斯特丹大学的非常水的课的一个随堂作业。事实上，这门课如果作准备，大部分既没有编程基础也没有数学基础的文科生也很容易过
- ▶ 这门课程 4 周 12 节课，共考试 13 次，8 次理论，4 次实操，最后一次大考；一次不过全盘否决
- ▶ 实操课内容为给定一篇 paper，从早上九点到晚上 12 点之前，必须实现并作出 Monte Carlo 模拟结果，并写成报告
- ▶ 难点：从 paper 到实现；如何在几核的电脑上跑几百万次模拟
- ▶ 在本章最后，我补充了简化版的习题

- 非常精细的写出模型当中的每一步
- 检查是否有标记的错误
- 使用推导当中的写法，忽略 pep8 进行开发
- 使用最笨的方式进行开发，不要考虑效率
- 使用 Monte Carlo 检查简单的模型是否正常

在开始本课前，注意复习...

- ▶ 极大似然的概念
- ▶ 矩阵求导的基本法则

- 1 概览
- 2 逻辑回归的实现
- 3 Proximal Methods 和实现
- 4 补充习题
- 5 参考文献

- ▶ 定义 $\sigma : x \mapsto \frac{1}{1+\exp(-x)}$
- ▶ 逻辑回归的概率密度函数为 $p_{\beta}(x_i) = \sigma(x_i^t \beta)$, 其中 β 为未知参数, x_i 为解释变量
- ▶ 负的对数似然函数为 $-\sum_i y_i \log(p_{\beta}(x_i)) + (1 - y_i) \log(1 - p_{\beta}(x_i))$
- ▶ 我们现在需要做的是求他的导数

- ▶ 由于矩阵形式非常简单，所以最麻烦的部分其实是对一堆非线性的函数的推导
- ▶ 我们当然可以手推，但是问题在于，手推很容易出错
- ▶ 所以这时候 sympy 就可以出场了
- ▶ 见 notebook

我们可以写出对数似然函数的导数为

$$-\sum_i (y_i \exp(-x_i^t \beta) / (1 + \exp(-x_i^t \beta)) - (1 - y_i) \exp(x_i^t \beta) / (1 + \exp(x_i^t \beta))) x_i$$

括号里面的东西还是有点复杂，所以我们不妨再试试看 sympy 是否能帮我们化简

化简结果如下

$$-\sum_i (y_i - \sigma(x_i^t \beta)) x_i$$

使用 `jax` 实现自动求导的过程并测试整体的正确性（见 colab notebook）。

- ▶ 请注意在这里，我们尽可能用接近于 Numpy 的形式进行了实现，并且我们通过 Jax 的 Autograd 机制判断了我们是否求导准确，这一点是非常重要的
- ▶ 如果没有 Jax，我们可能只能用 PyTorch 或者 TF 计算 Autograd（会相对来说更麻烦一些）
- ▶ 如果这些 Autograd 的机制都没有这时候我们面临的问题就更为麻烦，这时候通常使用 Finite Difference 进行调试
- ▶ 请注意这种调试往往会造成很大的误差，所以有时候很难进行判断

- ▶ 在我们上一讲当中，我们解释过如果我们得到了导数，接下来我们该使用一些优化方法来一步步进行优化了
- ▶ 不幸的是，就目前对于 Python 来说，不论是 `jax.scipy.optimize.minimize` 还是 `scipy.optimize.minimize` 都有巨大的问题
- ▶ 就 Jax 来说，大部分优化算法都没有实现，实现的部分也有 bug
- ▶ 就 scipy 来说，问题更大一些

- ▶ Scipy 优化的最大的问题在于，scipy 包装的是 fortran 77 的优化路径
- ▶ 在 fortran 77 的优化路径当中，其整体只使用了双精度，并对于各种存在的精度问题没有做任何优化
- ▶ 这使得在实际使用中，scipy 几乎永远不会得到正确的结论，因为各种 numerical issue 都会出现，并且不能修复
- ▶ 这也是为什么在科学计算中，大部分人还是使用 matlab 的原因

但是...

- ▶ 从 scipy 的问题回来，其实在之前的讲解中，我们还有一个问题没有解答，那就是可识别性的问题
- ▶ 什么叫可识别性呢？考虑以下问题

思考题：请问以下模型是否可以正常优化求解

- ▶ 假设我们的目标是 y ，我们有 x_1, x_2, x_3 三个变量，并且 $x_3 = 2x_1 + x_2$
- ▶ 我们是否能找到 $\beta_1, \beta_2, \beta_3$ 使得 $\sum_i (y_i - \beta_1 x_{i1} - \beta_2 x_{i2} - \beta_3 x_{i3})^2$ 最小
- ▶ 如果可能，我们能找到多少个？

以上问题称之为不可识别性的问题

- ▶ 简单来说，对于一个模型来说，存在（潜在）无穷多个解使得该模型对应的损失函数最小
- ▶ 对于存在线性表达式的模型来说，这种情况是极其麻烦的，这里面一种很常见的情况，称之为多重共线性，指的是一些变量可以用其他变量的线性组合表达出来
- ▶ 对于 R 来说，这些情况一般可以自动处理，即找到最大线性无关组，很不幸的是，在 python 中，scipy 的实现极烂（大约比正常 C++ 实现慢 100 万倍）
- ▶ 我们不会对具体算法进行讲解，具体算法已经在第二章当中的 cython 例子当中给出，大家可以直接使用

思考题：one-hot 编码输入逻辑回归之后是否可以正常求解？

- ▶ 如果有常数项的话，那么 one-hot 是不可以加入的，原因在于 one-hot 编码加起来等于 1
- ▶ 如果没有任何常数项的话（以及其他输入是可以的）
- ▶ 为什么要加常数项：假设我们用“受教育年限”对“工资”做回归，如果我们我们不加常数项，则等于我们认为未受过的教育的人的工资应该是 0，这显然是不符合实际的

见Cameron and Trivedi (2005) 16.3 的推导和实现

- 1 概览
- 2 逻辑回归的实现
- 3 Proximal Methods 和实现
 - Proximal Methods 的原理（选学）
 - Proximal Methods 的实现
- 4 补充习题
- 5 参考文献

- 1 概览
- 2 逻辑回归的实现
- 3 Proximal Methods 和实现
 - Proximal Methods 的原理（选学）
 - Proximal Methods 的实现
- 4 补充习题
- 5 参考文献

见附件。

1 概览

2 逻辑回归的实现

3 Proximal Methods 和实现

- Proximal Methods 的原理（选学）
- Proximal Methods 的实现

4 补充习题

5 参考文献

- ▶ 这一章，我们将会模仿真实的学习过程，及我们根据知名大学写出来的讲义，直接尝试实现
- ▶ 我们的目标是实现对数似然函数加上 l_1 损失的情况
- ▶ 在这里，假设对数似然函数为 $l_\beta(X, y)$ ，其中 β 为待估参数，而 X, y 为数据，我们的目标是最小化 $-l_\beta(X, y) + \lambda \|\beta\|_1$ ，其中 $\lambda \geq 0$ 为惩罚参数
- ▶ 具体实现过程首先由我们进行练习，然后再进行实现

见 Colab Notebook

思考题（进阶）：如何提升模型的效果

- ▶ 在上面的学习中，我们使用的 step size 都是固定的；在这种情况下，我们的结果类似于梯度下降；
- ▶ 那么是否有办法采用不同的 step size 呢？

- ▶ 首先务必检查数学推导是否正确：一般来说，最好的方法是和其他材料做交叉验证
- ▶ 其次务必检查每一步是否都有合适的结果
- ▶ 最后运行整个算法的时候，需要注意：
 - ▶ 算法是否真的收敛了？
 - ▶ 是否有 overflow 和 underflow？
 - ▶ 在多大情况下，算法会运行到一个局部最优？
 - ▶ 是否可以通过调整初始值的方法加速收敛？
 - ▶ 是否可以改变 line_search 的方向？

- ▶ Proximal methods 主要应用在 l_1 正则化的函数估计上；
- ▶ 这类方法还有很多，例如Efron et al. (2004) 和Garrigues and Ghaoui (2008) 等。目前在深度学习上也开始出现应用 (Yun, Lozano, and Yang 2020)。

- 1 概览
- 2 逻辑回归的实现
- 3 Proximal Methods 和实现
- 4 补充习题
- 5 参考文献

- ▶ 实现时间为 24 小时
- ▶ 可通过任何一种优化方法（BFGS 或 Proximal Methods）实现
- ▶ 根据模型内容，选择任何一种编程语言实现 100 万次以上模拟，并且根据该模拟研究该算法在不同情况下的可靠程度

第一题：非参数 kernel 回归

- ▶ 请选择至少两种不同的和 y 存在非线性关系的 X 进行实验
- ▶ 请实现逻辑回归中的 Kernel Regression 方法，见Cameron and Trivedi (2005) 第 9.5。并实现 Monte Carlo 估计
- ▶ 请回答：
 - 不同的 bandwidth 对于问题的影响有多大
 - 当 X 之间的相关性增加时，估计量效果如何？

第二题：Bayesian MCMC 估计

- ▶ 请复现Cameron and Trivedi (2005) 的 11.36 的内容
- ▶ 请研究 Prior 在样本增加时对于 Posterior 的影响大小

第三题: Nested Logic

- ▶ 阅读Cameron and Trivedi (2005) 的第 15.6 节, 并实现 Nested Logic 模型的估计
- ▶ 研究如果 Nested Structure 有问题时候, 上一层估计量的影响

- ▶ 阅读Cameron and Trivedi (2005) 的 15.9.1 节，并实现该模型
- ▶ 研究如果 ϵ 来自于和 log-likelihood 不同的分布时，估计量的性质

第五题：Tobit 模型

- ▶ 阅读Cameron and Trivedi (2005) 的 16.3 节，并实现该模型
- ▶ 检查当 ϵ 为柯西分布时对整个估计的影响

- ▶ 阅读Cameron and Trivedi (2005) 的 16.7 并实现 Roy Model
- ▶ 检查当 16.47 式子中，当 σ 假定有错误的情况下，对于 Roy Model 的估计有什么影响

第七题：Survival Analysis

- ▶ 阅读Cameron and Trivedi (2005) 的 17.6 节并且实现
- ▶ 检查在 Hazard Function 指定错误的情况下模型的表现

第八题：Finite Mixture of Count Regress

- ▶ 阅读Cameron and Trivedi (2005) 的 24.3 节，并实现模拟
- ▶ 请检查当 latent class 数量指定错误时候，模型的结果

第九题：Censored Count Regression

- ▶ 阅读Cameron and Trivedi (2005) 的 24.4 节，并实现 truncation 和 censored 中任选一种模型
- ▶ 请检查当 truncation 或者 censoring 错误时候，其估计结果的正确性

- 1 概览
- 2 逻辑回归的实现
- 3 Proximal Methods 和实现
- 4 补充习题
- 5 参考文献

-  Cameron, A Colin and Pravin K Trivedi (2005). *Microeconometrics: methods and applications*. Cambridge university press.
-  Efron, Bradley et al. (2004). “Least angle regression”. In: *Annals of statistics* 32.2, pp. 407–499.
-  Garrigues, Pierre and Laurent Ghaoui (2008). “An homotopy algorithm for the Lasso with online observations”. In: *Advances in neural information processing systems* 21, pp. 489–496.
-  Yun, Jihun, Aurelie C Lozano, and Eunho Yang (2020). “A general family of stochastic proximal gradient methods for deep learning”. In: *arXiv preprint arXiv:2007.07484*.