机器学习系统中的技术负债

沈祥壮

应用统计学

数学学院
中山大学

12th September 2020

目 录

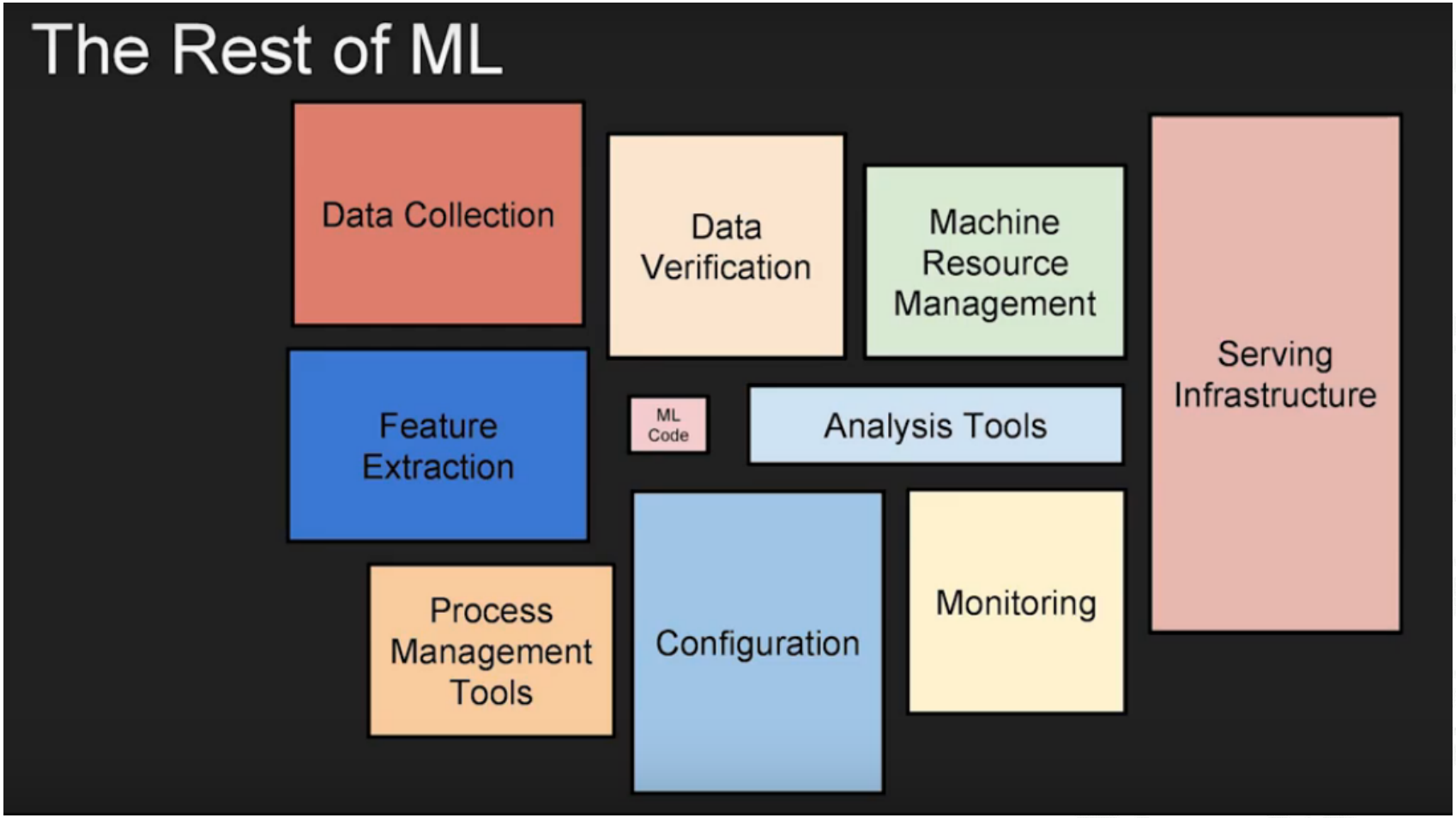

图 1: ML System([SHG+15])

Main Roles?

Only a small fraction of real-world ML systems is composed of the ML code, as shown by the small black box in the middle. The required surrounding infrastructure is vast and complex.

Why: Motivation

Uncomfortable trend

Developing and deploying ML systems is relatively fast and cheap, but maintaining them over time is difficult and expensive. (technical debt that must be paid!)

Compounds silently

Deferring such payments results in compounding costs. Hidden debt is dangerous because it compounds silently.

What: The Definition of Technical Debt

Technical debt(Wikipedia)

Technical debt (also known as design debt or code debt, but can be also related to other technical endeavors) is a concept in software development that reflects the implied cost of additional rework caused by choosing an easy (limited) solution now instead of using a better approach that would take longer.

Technical debt([SHG+15])

The long term costs incurred by moving quickly in software engineering.

What: Technical Debt in ML System

1. All the problems of regular code
2. Plus ML-specific issues at system level

How: Many ways

[SHG+15]

Technical debt may be paid down by refactoring code, improving unit tests, deleting dead code, reducing dependencies, tightening APIs, and improving documentation.

The goal is not to add new functionality, but to enable future improvements, reduce errors, and improve maintainability.

Unstable Data Dependencies

不稳定的数据依赖

当前系统运行时会依赖上游的数据，当上游数据生产不稳定时，可能会导致系统出现难以预期的问题，而当我们想定位这些问题时，可能会十分困难.

数据解析模块的版本更新

在上游数据发生变化的时候，更新系统当前的数据解析模块。使得每个上游数据对应一个数据解析模块；每次数据变动更新对应数据解析模块的版本。

这是一种较为通用的方法，实际上我们应尽量选用更加稳定的数据源，或尽量控制上游数据的稳定性 (有控制权限的前提下).

Underutilized Data Dependencies

未充分利用的数据依赖分为以下几种：

1. Legacy Features: 旧系统遗留下来的特征，如旧的产品编号
2. Bundleed Features: 为提升效果加入一系列特征，而其中有些特征其实对模型并无提升或提升很小
3. ϵ Features: 为了在现有模型效果中更进一步 (即使只有很小的提升) 而加入的一些复杂度比较高的特征
4. Correlated Features: 相关特征

定期检测并删除

在系统不断迭代的过程中，应当定期对这些特征进行检测，删除冗余特征，尽可能较少数据的依赖，提升系统的稳健性.

编程语言: 必须作出选择

在选择机器学习系统的主要的编程语言时要考量的东西比较多，总是需要作出取舍。

1. Python: 语法简单，学习成本低，机器学习生态好；动态类型不适合大型项目，运行效率过低
2. Java: 使用范围广，生态较好，静态类型，效率尚可，学习成本较低；效率不如 C/C++, FP 支持不友好
3. Scala: 大数据生态的核心，分布式的原生支持，静态类型利于大型项目开发与维护，FP 友好，效率较高；学习成本较高
4. C/C++: 静态类型，效率高，底层优化；大数据生态不如 Scala, 学习成本也稍高

编程语言: 必须作出选择

1. Hadoop+Spark+SQL+Java(Or Scala): 各大互联网公司普遍在使用的 (腾讯，字节，快手，Bilibili, Tubi, ...). Scala 很多公司也在用，但是使用范围可能比较小.
2. Python+PySpark: 一些业务算法的开发可能会选择的方案，PySpark 可以很好地与企业现有的大数据生态系统交互，同时开发成本相对较低，利于算法前期的快速迭代. 一般来说性能也不会比 Scala 调用 Spark 慢太多，只是维护成本相对较高
3. C/C++: 现有的很多主流的机器学习框架的底层都还是用的 C/C++ 来实现, 如 Tensorflow, XGBoost 等

第三方库：双刃剑

优点

第三方库的调用提供了很多便利

缺点

库的**安全性**，**稳定性**和**兼容性**等问题都可能破坏机器学习系统的稳健性

废弃代码分支：去与留

在开发算法的过程中，经常会建立很多的测试分支，而这些分支中仅有少部分会合并到 Master 当中，其余的代码要做好处理：**确定无用的分支直接删除，仅仅保留少量有借鉴价值的实验分支**

管中窥豹：抽样数据不能完全反映全量数据

对数据的深刻认识

开发算法时基于抽样数据来进行快速的测试，在全量数据下可能出现很多解析上的问题. (某特征在抽样数据中无缺失，但是在全量数据上存在缺失值；全量数据中某些值的编码问题)

测试 + 验证

先多次抽样汇总问题，再全量数据上验证；在系统中添加特殊的容错处理，处理未来可能出现的数据格式的问题.

管中窥豹：抽样数据不能完全反映全量数据

对模型复杂度的深刻认识

基于小数据测试发现模型效果好，效率也能接受. 但是在别的场景下会失效，如更高维的数据下模型复杂度指数上升.

最坏的打算

提前评估整个算法系统的理论计算复杂度

模型参数的配置

如何设计配置文件，使得算法本身利于迭代开发；使得算法在新场景的调用更加简单.

CACE principle

Changing Anything Changes Everything.

沟通的代价

1. 协同开发中的代码规范问题
2. 新人能否快速掌握整个系统的运行
3. 如何将算法服务推广使用

有效的提问

1. MWE: Minimal Working Example
2. XY Problems
3. 问题抽象：业务背景，算法背景的剥离；数据脱敏

有效的提问: MWE

最小工作示例

对于编程语言的语法问题和一些工具库的使用问题，务必将核心的错误逻辑抽取出来，提供一个 MWE 来复现我们遇到的问题

提问的态度

MWE 是经过检验的，十分高效的解决问题的方式，作为寻求帮助的一方，我们需要端正自己的态度，将问题抽取出来并提供 MWE 至少是表明了自己虚心请教的态度——直接扔出一大堆报错代码然后让别人帮忙解答，是应当极力避免的行为.

有效的提问: XY Problems

- 某人想要解决 X[1]
- 他不知道怎么样才能解决 X，但感觉 Y 可以解决
- 他同样不知道怎么样才能解决 Y 他寻求 Y 的解决办法
- 其他人想要帮助他解决 Y，但是摸不着头脑为什么要使用 Y
- 经过漫长的交流，终于明白了原来他是想要解决 X，而 Y 并不是解决 X 的合适的方案

[1] https://zhuanlan.zhihu.com/p/130335668

有效的提问: XY Problems

- Q: 我怎么用 Shell 取得一个字符串的后 3 位字符？[2]
- A: 如果这个字符的变量是 *$foo*，你可以这样做：*echo ${foo:-3}*。为什么你要获取字符串的后 3 位？你到底想要什么？
- Q: 我想要获取文件的扩展名
- A: 文件的扩展名并不一定是 3 位啊! 那你试试这样做：*echo ${foo##*.}*

[2] https://zhuanlan.zhihu.com/p/130335668

有效的提问: 背景剥离与数据脱敏

清晰的定义并表述自己的问题是相当困难的一件事，尤其在寻求非项目组成员的帮助时。

提出问题时已经解决了问题的一半

我们必须将从一个涉及业务背景或算法背景的问题 X **正确的表示**为更容易理解的 Y(保证不出现 XY Problem)；同时不要涉及敏感的业务数据.


David Sculley, Gary Holt, Daniel Golovin, Eugene Davydov, Todd Phillips, Dietmar Ebner, Vinay Chaudhary, Michael Young, Jean-Francois Crespo, and Dan Dennison.
Hidden technical debt in machine learning systems.
In *Advances in neural information processing systems*, pages 2503–2511, 2015.