Hidden Technical Debt in Machine Learning Systems

* Massive ongoing maintenance costs in real-world ML systems (technical debt)
* ML specific risk factors:
  + Boundary erosion
  + Entanglement
  + Hidden feedback loops
  + Undeclared consumers
  + Data dependencies
  + Configuration issues
  + External world changes
  + System-level anti-patterns 反面模式（软工定义：实践中反复出现但却低效有待优化的设计模式）

## General Introduction:

**Key: Deploy and develop ML system is fast, while maintaining them is difficult and expensive. ML systems have all traditional maintenance problems plus an additional set of ML-specific issues.** (focus on system-level interactions and interfaces – design, develop, glue code or calibration layers)

**Technical debts are caused by fast development and they are dangerous** (not to add new functionality, but enable futureimprovements, anti-reduce cost and improve maintainability).

## Complex Models Erode Boundaries

Traditional software engineering shown strong abstraction boundaries using encapsulation and modular design help create maintainable code for isolated changes and improvements. Strict abstraction boundaries help express the invariants and logical consistency of the information inputs and outputs from a given component. [8]

**Abstraction:** hide the internal implementation but providing interface for function access with input output information

Problem 1: it is difficult to enforce strict abstraction boundaries by prescribing specific intended behavior (很难找到一些特定的有范式的行为来进行抽象). 正好相反，ML的软件逻辑与外部数据输入是十分相关的 – the desired behavior cannot be effectively expressed in software logic without dependency on external data.

1. **Entanglement:** signals are mixed. (Change a distribution of one dimension -> the weights of other dimensions are changed)
   1. **Possible solution 1:** isolate models and serve ensembles (combination entanglement)
   2. **Possible solution 2:** detect changes in prediction behavior (cross dimension and slicing visualization to see effects)
2. **Correction Cascades:** model reuse leads system dependency issues
3. **Undeclared Consumers:** prediction from a machine learning model may later be consumed by other systems and those consumers might be undeclared, leading silently usage without access control -> **visibility debt (hidden feedback loops, tight coupling)**

## Data Dependency Cost More than Code Dependencies

**Dependency debt** – data dependency in ML systems (code dependency can be analyzed by compilers and linker, while large data dependency chains can be difficult to untangle)

1. **Unstable Data Dependencies:** with unstable signals used as inputs, it is costly to diagnose and address the possible detrimental effects in the consuming system.
   1. **Possible solution:** maintain versioned copy of signal until fully verified.
2. **Underutilized Data Dependencies:** signal providing little incremental modeling benefit could make an ML system vulnerable to change (dependencies issues)
   1. **Legacy features:** included in early development but becoming redundant overtime
   2. **Bundled features:** brutally include features with little values
   3. **High Overhead features:** with little accuracy increase but high overhead
   4. **Correlated features:** miscredit correlated features

## Feedback Loops

ML system often end up influencing their own behavior if they update over time, leading analysis debt – it is difficult to predict the behavior of a given model before it is released.

**Direct feedback loops:** influence its own future training data – isolation or randomization to fix

**Hidden feedback loops:** interactions between systems

## ML-System Anti-Patterns

High-debt design patterns (little code devoting to learning or prediction)

**Glue code:** glue code is the code supporting the data input and output for the generic packages and tend to freeze the system to specific package – maybe less costly to not use the generic package and create a clean native solution

**Pipeline jungles**: tedious code used for preparing ML preferred data for new added signals or features

**Dead Experimental Code paths:** abandoned dead experimental conditional branch might cost problems, requiring periodically to be ripped out

**Abstraction Debt:** lack of widely accepted abstraction

**Common Smells:** data type, multiple-language, prototype

## Configuration Debt

Feature inclusion and several principles are concluded, including smooth transition, avoid manual errors, visually difference comparison, redundancy detection

## Dealing with Changes in the External World

Interaction with external world, leading ongoing maintenance cost

**Fix Thresholds in Dynamic Systems:** updating thresholds across many models is time-consuming and brittle

**Monitoring and Testing:** comprehensive live monitoring of system behavior in real time combined with automated response is critical for long-term system reliability. Several basic monitoring points are offered, including prediction bias, certain alerts, up-stream data source monitoring.

## Other Areas of ML-related Debt

**Data Testing Debt:** test code with data replacement

**Reproducibility Debt:** randomized algorithms make reproduction difficult

**Process Management Debt**

**Cultural Debt:** awarding complexity reduction and monitoring accuracy improvements