# 在做算法工程师的道路上，你掌握了什么概念或技术使你感觉自我提升突飞猛进?

CV开发者都爱看的 极市平台 2022-03-07 22:00:00 手机阅读





作者丨金瀛若愚、桔了个仔、DLing（已授权）

来源丨https://www.zhihu.com/question/436874654

编辑丨极市平台



极市导读

算法工程师作为近几年非常火热的岗位，近几年校招也开放了大量的算法岗位。作为想要在这个岗位上不断前进不断深入的人，有哪些tips或者经验可以传授给这条赛道上的各位呢？本文汇总了知乎上三个优质的回答，希望能给各位一点启发。 >>加入极市CV技术交流群，走在计算机视觉的最前沿

## # 回答一

作者：金瀛若愚

来源链接：https://www.zhihu.com/question/436874654/answer/1808192248

在科研中训练到的思维方式，是我最宝贵的成长。

## 一. 反馈的闭环

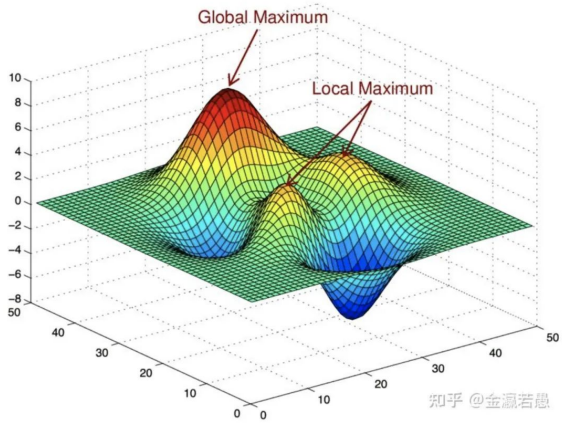
在一个研究任务中，我首先尝试了方法A，没搞定，于是改用方法B。组里大哥问：为什么改方法B。我说A没效果，或许B能work。大哥继续问：为什么A不work。我说或许A不适合这个问题？大哥说：当你有一个尝试，你一定要知道它为什么work以及为什么不work。每次不work了你就换另一个方法，那另一个方法就能work吗？这不是科研，是赌博，是瞎试。你只有知道为什么一个方法有效或不有效，何时有效何时无效，你才能增进对这个问题的理解，然后基于此提出有价值的策略。

我的思考是，既然做了一件事，就一定要得到反馈，要搞清楚哪里做得好哪里不好，这样这个尝试所投入的时间才是有效的。不然就是在碰运气，如同做题不对答案，如同训练模型不算loss不做backprop。

## 二. 把炼丹技术推广到生活中

机器学习的很多技术都与现实世界的概念相互呼应。

在深度学习里，模型掉入局部最优，就是生活中的内卷，就是在狭窄的赛道上追求极致。摆脱内卷就是跳出局部寻求全局最优的过程。破局之道是尝试新事物，或增加训练数据。




与人交流、观察和学习他人可以避免闭门造车。与他人交流就是深度学习里的增加训练数据。进入好的学校好的公司就是提高训练样本的质量：在label准确时，学得轻松。

前面讲的“构建闭环”，就是关注backprop时的梯度。你不能攒了特别多事情后再去反思或寻求反馈，这个反馈链太长，要么梯度消失，要么梯度爆炸，无法有效学习。类似对loss的求导过程：你必须清晰地看到因果链条，才能做到从结果倒推原因，进而优化自己。

已经做得很好的事情没必要重复。此时应该挑战新的项目，保持自己在学习的状态。这就是hard data mining. 遇到坏人后就认为全人类没救了，是overfitting。对应在机器学习里，在unbalanced dataset上学习，要想到用weighted loss。

挫折易使人变得复杂、内心冲突多。如果能用简单的形态存在，我们或许应警惕过分复杂。机器学习里，解决小问题硬上大模型是一种粗暴且没技术含量的办法。用不必要大的模型是对探求事物本质的逃避，科研如此，生活亦然。你以为模型练成了，其实它学的是shortcut，因此有泛化能力差的问题：variance大，不robust（内心不稳）。同时，模型太复杂就不容易理解，遇到bug不容易诊断病因（内心不易平稳和愉悦）。但避免复杂不是要当巨婴，不是抗拒成长。当任务复杂、训练数据也大的时候，就必须要上大模型。核心是模型复杂度（心智复杂度）要与任务复杂度和数据量（阅历）匹配，才是健康的。

## 三. 交流，交流，再交流

组里的女神姐姐教导我：“做research要多和人交流，多去听别人的paper reading，也把你读到的论文和想法讲给别人听。因为在讨论的过程里你会意识到未曾发现的问题。尝试给人讲明白的过程里，你的思路也会越发清晰。”道理我都懂，每次我头点得像敲鼓一样，但也没坚持做到。

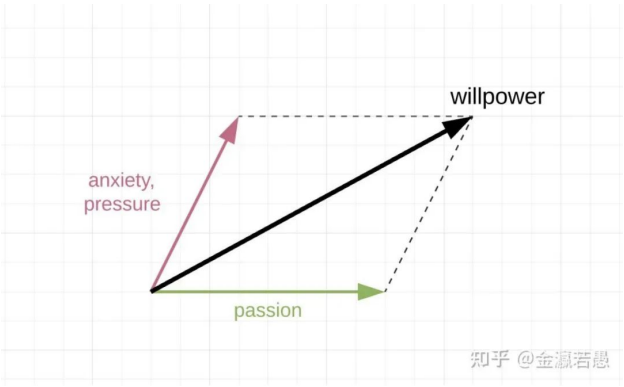
为什么做不到？大概是心理包袱，总想着搞出一个牛X闪闪的东西后再展示给别人，不然觉得丢脸。或者总有一种再试一下就能搞定的错觉。这种错觉和我炒股票的风格如出一辙。

直到后来我看到越来越多平淡无奇的项目都逆袭了，我理解到，事物的发展都有必然的过程，要尊重其发展规律，不要总想直接搞个大新闻。高效的科研要主动的寻求身边的资源，争取让良师益友把时间花在自己身上，如果自己不主动争取，再照顾你的人也不能像亲妈一样耳提面命，他们毕竟不好逼迫你。具体的，应该多把自己读懂的论文讲给别人，在你给他人创造价值的同时，你也使他们帮助你这件事更容易了：他们只有懂了你懂的，才能给你最有效的建议，你才能借用他们的头脑思考。

我发现当我积极主动之后，身边的人都默默支持起我来。稻盛和夫说：“心不唤物，物不至”。

## 四.「焦虑动力」模型

勤奋上进、自我驱动、有行动力，本质是内心动力充沛的精神状态。动力不足时，我们懒惰、拖延、自控力差。用向量分解的思维模型，我们可以看清「动力」的本质。




Willpower=动力; Anxiety=焦虑；Pressure=压力；Passion=热情

- 动力的分量是 1.焦虑/压力 和 2.热情。动力是二者的合力

带着上面这句话，我们可以举出奋斗者的两种极端状况。一种由焦虑主导，被绩效和deadline催促，常处于不安和压力之中。希望获得他人的认可，如果有了过失，会感到煎熬；另一种是热情主导的。专注、投入，不易为外界干扰和诱惑所动。在做事的过程中，能收获成就感和喜悦，认为工作与个人追求比较重合。

警惕"动力"过分依赖"焦虑和压力"的状况。这个动力来源并不持久和可靠。随着年龄增加，体力和心力会下降，且随着阅历增加，阈值变高，很多事会不再觉得重要。那时就会失去动力。

分量之间存在代偿。盲人的耳朵往往很敏锐。如果动力过多依赖焦虑和压力，则留意到热情和乐趣就更难。可能并不是你不热爱，你只是太紧张了而已。

## 五. 独立思考

我对独立思考这件事体会最深的就是决定要不要读PhD时。拿到UW的录取后，因为微软组里就很多博士、教授，我自然就去请教他们问要不要读博，毕竟五年是一笔不小的投入。当时有一部分人说值得一读，另一些人说专心做事业可能有更大回报。

有趣的是，当我签下offer后，所有的人，不论之前给了什么建议，都由衷恭喜和认可我的决定，并和我畅想毕业后Dr. Jin走向人生(996)巅峰的画面。

那之前他们的建议是真心的吗？当然是真心的。签offer后的认可也是真心的。于是我理解到，很多事情，做与不做都能找出道理。他人给建议时，也会考虑到我们的情绪，让我们不论怎样选择都有台阶下。因此，我们要保持独立思考，要对自己的决定负全责。

建议的推导逻辑比结论重要。他人建议的正确用法是让自己看问题多个角度，减少信息差，而不是直接取其结论，让他人代为做选择。

## 六. 把工作当做一个二阶优化过程

不管title是算法工程师还是应用科学家，产出都是代码及其体现的知识产权。我们工作的过程就是优化这个代码及知识产权的过程。但这太basic。

我们不仅要优化产品，也要优化产出产品的过程：一阶优化，是优化代码质量。二阶的优化，是优化工作过程，这个工作过程是代码质量更上一层的原因，是原因的原因，是二阶导数。

类比一下：为了走得远，你可以优化速度（路程的原因是速度），更进一步你可以优化加速度（速度的原因是加速度）。不断向上溯源，能解决根本问题、通用问题。

回到优化工作流程上，我们不断问自己：在我工作的过程中，有哪些是重复工作？有哪些可以更高效的完成？有哪些可以被拎出来整理出可以复用的，于是后面就不再需要花心思重做或者检查其bug？经过一两年我攒出了一套自己的代码库，很多被重复使用的部分（如用matplotlib画各种图，各种数据预处理）就都可以直接copy paste。这就形成了复利效应——时间越长，这些整理出来的代码片段就创造越高的价值。

## 七. 学习些销售意识

销售是一门大学问，值得广大猿类学习。

- 像设计产品外包装一样设计履历

一个畅销的产品必须有个特色，这个特色与其他竞品形成了差异化竞争，才能有自己的市场，卖上个好价格。比如始祖鸟这个牌子的衣服颜值很一般，但它的防水面料好，就拿下了户外市场。一个中庸的什么都不差也什么都不突出的产品就很难被你记住。同时，产品的优秀比不上品牌的优秀，如果产品的价值凝练成了品牌，那这个品牌本身就值高价。比如我们会为了logo付费。

我们的职业发展，说得现实些，也是希望自己在市场上有个好价格。为此，我们不需样样精通，但必须有一个具有代表性的，能拿得出手的技能。比如，我就是要和所有人不同，去学远古技术汇编语言。那只要市场出现了对这个技能的需求，你就能有极大的定价权。当我们深耕一个领域很久，又写了很多优质的博客，那你的名字就成为了品牌，可以帮助公司招贤纳士，等等。

我的老板兼导师也曾说：你做paper要想怎样把你的paper卖出去。你要站在消费者角度想，他们为什么要花时间去读你的文章？你的论文有没有创意，能给他们带来什么价值？写作语言，图表美观程度，就是卖相。这和销售很像。

我们的简历就是商品的成分表——在决定选择什么项目时，不妨想想这会在简历里留下一行什么样的记录，会不会帮你抬高自己的职业价值。

- 用产品思维规划项目

科研项目的立项过程和产品策划非常像。第一步都是要做survey，了解清楚目前技术有哪些分支，是什么历史契机促成了某个技术的出现（比如有了新的数据集），不同的技术优点缺点在哪（A更准确，B速度更快，C不需要很多训练数据，等等）。然后了解这个任务的定义，metric（关键指标）是什么，也就是搞清楚游戏规则，知道大家在比些什么，头部的玩家都是谁等等。不了解清楚，后面的一切都是错的。

导师们经常问我的一个问题就是“这个task的upper bound是什么”。我说我知道state-of-the-art是什么，还要知道upper bound吗？导师说你在开始一个尝试前，要知道还有多少空间留给你去做。如果SOTA（目前最好的）已经比较接近upper bound了，你就很难再往上去攻了，你可能找个别的任务更有成就感。你跳进一个赛道前，要对其发展到了什么阶段有些概念。

## 八. 关于忙碌

在上面各种催人上进的内容之后，我再来分享一个故事来结尾。

一次公司的一位高层前辈见面，临走前我问他能不能给我个过来人的建议。

我以为他会给我讲一些勤奋工作的态度、人生规划的经验、或者给我打打鸡血这种内容。

而前辈只是指了指办公室书架上的一排没拆封的乐高跟我说，我给你一个建议，就是年轻时多花点时间在自己的嗜好上。我年轻时喜欢乐高，但因为一直忙于工作，并没有花很多时间在我

这个小小的嗜好上。现在我时间多了，但对乐高也不再有那种热情了，现在只把他们摆起来，包装都没拆。所以年轻的时候，你可以花点时间在自己的爱好上。不能只有忙碌的工作。

很片面，欢迎指正。

## # 回答二

作者：桔了个仔
来源链接：https://www.zhihu.com/question/436874654/answer/1846044111

很片面，欢迎指正。

谢邀。这道题我犹豫了很久，三个月前就放草稿箱里了，一直没答的原因是，感觉这些年好像没有什么瞬间让我感觉自己水平突飞猛进，感觉这些年的进步都是慢慢进步的，好像没有「突飞猛进」的时候。昨天带我入行的同事离职了，在部门的送别仪式上，我们回忆起我第一天入职时我有么的「小白」，后来怎么独挡一面。于是我在想，如果要总结一个让我的水平提升的关键「概念」或者「技术」，那会是什么。

于是我回顾这些年的技术历程。说实话，在我开始搞算法后，我日均代码量远远低于我之前做游戏时，甚至有的时候，几天都不写一行代码。当然，我并不能代表所有算法工程师，我虽然也是做数据科学工作，但我需要和客户保持联系，理解需求。但无可否认，代码量少的这些日子，我反而做的事情更靠谱，让我更有成就感。

我于是总结出一句话：

**解决问题不要于拘泥于技术。**

当然，仅仅一句话的话，大家可能看得一头雾水，我展开说说吧。

不要拘泥于技术分为三层：

### 1. 不要执着于使用最新的技术。

有的人想走技术专家的路线，那非常好。每天看看arxiv，看看最新的SOTA，那是一个好习惯。但不要太频繁的变更技术方案。你那种想把产品做到极致的思维，我能理解，我也有过，但你真正做产品落地了，会发现，产品是一个系统，你的技术方案变更可能会对其他模块造成影响。

举个例子，我们的模型是XGBoost，不算很复杂吧？由于我们给多个客户部署过系统，我们知道在什么样的硬件条件下运行时间是多少。例如在每日1万条数据的情况下，我们用AWS的t2.xlarge实例，运行时间是1~2小时，符合客户要求。但如果你看到最新的paper提出了一个新模型，决定要采用的话，除了你的技术方案可行性要得到验证，你的技术方案对系统运行时间的影响也要重新评估。如果你的方案确实效果更好，但服务器成本高出几倍，我们该如何说服客户？这都是是一环扣一环的，实际可能遇到的情况可能比我说的要复杂得多。这，就是系统的世界。

> 统计建模并不是为了获得完美的预测能力，而是用最小的必要的模型来实现最大的预测能力。

更何况，很多情况下，使用更复杂的方案未必是最合适的方案。我之前写过一个回答，讲了哪些深度学习效果不如传统方法。

有哪些深度学习效果不如传统方法的经典案例？：https://www.zhihu.com/question/451498156/answer/1802577845

在这个回答里，我引用了那个著名的「电风扇吹香皂盒」的段子。

> 某大企业引进了一条香皂包装生产线，结果发现这条生产线有个缺陷：常常会有盒子里没装入香皂。总不能把空盒子卖给顾客啊，他们只得请了一个学自动化的博士后设计一个方案来分拣空的香皂盒。博士后拉起了一个十几人的科研攻关小组，综合采用了机械、微电子、自动化、X射线探测等技术，花了90万，成功解决了问题。每当生产线上有空香皂盒通过，两旁的探测器会检测到，并且驱动一只机械手把空皂盒推走。中国南方有个乡镇企业也买了同样的生产线，老板发现这个问题后大为发火，找了个小工来说"你他妈给老子把这个搞定，不然你给老子爬走。"小工很快想出了办法他花了190块钱在生产线旁边放了一台大功率电风扇猛吹，于是空皂盒都被吹走了。

但这里需要说明的是，不执着于使用最新技术，不代表不关注最新技术。有的时候，公司的需求是一回事，但我们的技术进步又是一回事，如果我们不care最新技术，那么当我们跳槽时，就会很吃亏。

## 2. 不要只用技术

技术不是职业生涯的全部，尤其是对于做商业产品的算法工程师。作为算法工程师，尤其是做商业产品的，其实有很多时候，技术并不能解决所有问题；有的时候，技术虽然能解决，但是中策，甚至下策，而上策不一定需要使用技术。代替技术的方法有下面四种，一种比一种境界高。

**用金钱代替技术。** 例如你的产品上线了，发现在线上模型训练总是不成功，发现是服务器性能不够，那么你第一件事是优化你的算法还是花钱扩充服务器？如果对于商业价值很高的项目，宕机一天损失可能远远超过服务器开支，这时候，你第一件事买服务器，优化的工作放到日后。

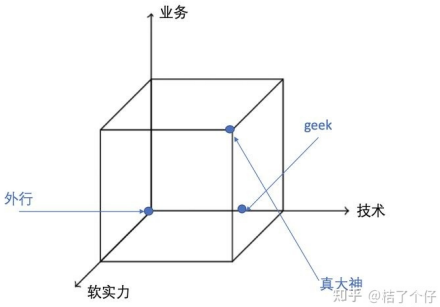
**用沟通代替技术。** 很多人经常方案进展到一半推到重来的情况（设计师尤甚）。对于算法工程师，如果沟通不充分就直接开工，那么很容易做无用功。举个大家都能理解的例子，例如有个银行让你开发个算法识别哪些客户可能逾期，但由于银行数据敏感，不能直接给你，只给了你data schema让你自己模拟数据，你说没问题，回去花了一周写了个模型，准确率（accuracy）是90%，很不错，结果在客户那一上线，准确率99%，你更高兴了，但很快你发现，客户的数据label 0:1的比例是1：99，你准确率99%和瞎猜没区别。事实上，如果你和客户沟通得当，早点理解清楚数据情况，你就不会采用accuracy这个评价指标。

**用管理代替技术。** 这是区别高阶工程师和普通工程师的重要能力。一个项目，如果管理得当，那么可以让一线开发人员轻松点的同时完成更多工作。

- 有的工作是另一个工作的基础，完成A再完成B的总时间可能小于先完成B再完成A。
- 工作是永远做不完的，即使都是技术工作，也有价值不同。好的技术管理人员应该让大家先做价值高的。

**用眼光代替技术。** 做到这个级别，基本就是CTO级别了，充分了解现有技术优缺点，知道什么地方应该投入新技术，什么地方应该采用旧技术，什么地方甚至不需要技术。但毕竟我还没做到这层，这里就没什么经验可以分享了。

更何况，对于算法工程师，除了技术能力，还有很多能力是必须的，如下图，这些能力，组成了算法工程师的「落地能力」




图来自我之前写的一个被知乎日报收录了的回答。这里就不再重复其内容了。

算法工程师的落地能力具体指的是什么？：https://www.zhihu.com/question/304695682/answer/1720475610

## 3.不要只关注自己领域的技术

抛开非技术的能力，来集中讲讲技术。对于技术能力，你是否以为你关注最新论文，会调参，就完事了。没这么简单。我每次讲算法工程师工作内容时，都会贴这张图

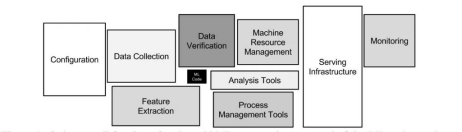



Figure 1: Only a small fraction of real-world ML systems is composed of the ML code, as shown by the small black box in the middle. The required surrounding infrastructure is vast and complex.

中间那个小小的几乎都快看不见的黑块，你放大图片，会发现里面写着ML Code，这就是「算法」的部分。当然，别被这个图吓到，这不一定全是你的工作，这里是一个团队的任务，这个团队可能是两人的团队，也可能是几十人的团队，但可以肯定的是，无论你在哪个公司，一个算法工程师都不太可能只做纯「算法」，不要忘了「工程师」三个字。

我作为一个算法工程师，你以为我只写python，偶尔加点scala？不，你可能想不到，我还写vba。我们客户是银行，他们有内部的模型验证团队，我们需要根据他们提供的模板来生成模型性能报表，而且这个报表模板还不能往外网传，那怎么办？只能用VBA写个Excel宏脚本自动生成报表呗呗。毕竟他们提供的可编程环境就只剩excel了，至于python什么的，由于没联网，pip install啥都不行。

所以我建议，作为算法工程师，多学习下周边技术，不要仅仅只会调参，看论文。周边技术是指能为你快速实现验证与部署的技术，并非所有技术都要学习。你作为一个算法工程师，学习linux，可以方便你部署模型；学习spark，能方便你快速处理大数据；这样你有什么新的idea都可以快速验证而不用再求数据工程师帮你反复做ETL。

但，**吾生也有涯，而知也无涯。以有涯随无涯，殆已！** 但你作为算法工程师，去学习安卓app开发，似乎就有点不必要了，除非你确定用得上，或者你是你们公司唯一的程序员（这种情况建议跳槽）。

这就是我在自我提升路上的一点小总结。

# 回答三

作者：DLing

来源链接：https://www.zhihu.com/question/436874654/answer/1744174731

突飞猛进不敢当，但是从事深度学习图像算法这几年来，却是也有一些感慨。

1. 数据放在第一位，成也数据，败也数据。深刻认识数据的重要性，把数据集维护好，数据量够了，再谈后面的模型优化，数据都不干净，用再好的模型，也不会出好的结果。

2. 启动开发前，多问问自己有没有了解这个业务，目前定的方案还有没有盲点没有考虑到，毕竟启动开发需要准备各种数据集，耗时长且需要一定的人员和经济投入，如果开发过程中或者测试阶段发现方案不合适，这时候推倒重来的话，就DT了。

3. 算法工程师并不只是调包侠，炼丹师，而是一个综合要求很高的岗位。要训的了模型；写的了逻辑；优化的了算法性能，时刻把运行速度，准确率，召回率，显存利用率，显存占用，cpu利用率，内存占用，并发路数等等记在心里；还得深刻了解业务，目前的方案合不合适？产品定的指标，给的需求有没有坑？完成这些需求，要选什么样的硬件最划算，可不可以少几个模型，毕竟看数据也很费眼；

4. 多实验，多记录，多对比，勤讨论，勤汇报，勤迭代。这一行多少还是有点玄学的，很多问题没有很强的理论可以支撑，靠经验的地方很多，这个模型效果好，很多情况也不是推导出来的，而是实验出来的，有时候想破脑袋，也没有动手起几个模型效果来的快。平时多跟同事同行讨论讨论，搞不好费了你好几根头发的问题，就被别人解决过呢?

5. 多看行业顶会论文，多追追大牛的博客，思路打开了，落地也就简单很多。

以上是我从事算法行业几年来的一点点体会，不是具体到看了某一篇论文，学了某一个框架，熟悉了某一个语言给自己带来的提升。但就我而言，这些对岗位认知的更新，做事套路的更新对自己的提升有时候要强于某一项具体的技术。今天把这些体会分享出来，希望能符合题主预期。