

架构师

ARCHITECT

| 特刊 |

范式大学

SPECIAL ISSUE
Oct. 2017

架构师特刊



Geekbang
极客邦科技

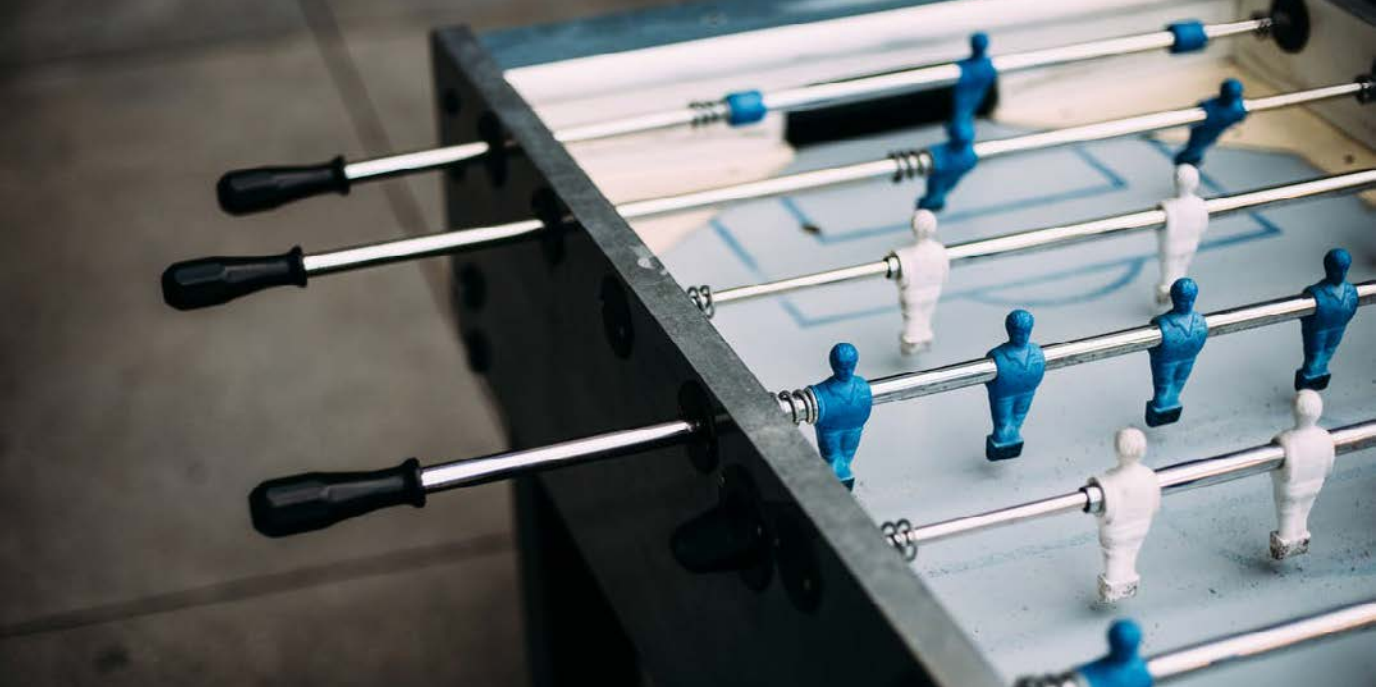
InfoQ

序言

第四范式首席科学家 杨强

凯文·凯利曾说，“AI 会是下一个 20 年里颠覆人类社会的技术。”显然，拥有大数据的互联网公司已经嗅到这一机遇，包括亚马逊、谷歌、BAT 在内的巨头们纷纷开始布局 AI；传统企业也不甘落后，试图在智能时代保持竞争优势，把握机会，成为下一个商业时代的巨擘。尽管入局者们的商业模式、应用场景和技术选择各有不同，但通过 AI 提高企业运营效率、抢夺精细化市场、改善用户体验，成为企业当下的首选。

现如今，通过超高维度的 AI 技术，可以帮助企业进行以往无法想象的精细化运作，从而大幅提升企业运营效率、提升企业核心竞争力。在数据量极高、模型“VC 维”（“VC 维”是统计学指标，可度量机器的智能水平。VC 维越高，机器的学习能力就越强，机器学习模型越准确）极高的情况下，AI 的能力已经可以代替、甚至超越人类专家，处理各类业务问题。换句话说，人类通过总结规则解决实际问题，在具体应用场景中，



业务专家总结的规则数越多，维度就越高。而 AI 技术可以让机器也来总结规律，因为机器精力无限、且能抓住最细粒度的信息，从而总结出大量人类无法发现的规律，拥有远高人类的学习能力。

然而，AI 技术若要在工业界全面落地，人工智能技术及服务提供商第四范式认为：除了超高维度的技术支持，还需满足 Big data(过程数据)、Response（反馈数据）、Algorithm（高维算法）、Infrastructure（计算资源）和 Needs（明确需求）等五个核心要素。第四范式具备国际顶尖的机器学习技术，通过对数据进行精准挖掘与预测，帮助企业提升效率、降低风险。目前，该公司已为金融、互联网等百余个企业成功打造了 AI 解决方案。在企业独立构建 AI 能力门槛较高的当下，第四范式希望通过自主研发的先知平台，降低人工智能的准入门槛。第四范式相信，在“先知”平台的帮助下，每家企业都可以开发属于自家的人工智能技术，享受 AI 技术革命带来的商业能量，立足当下、决胜未来。

目录

- 05 （一）构建商业 AI 能力的五大要素
- 12 （二）AI 不是万能的，判别 AI 改造企业的 70 个指标
- 22 （三）机器学习的 MVP——用最小成本 验证 AI 可行性
- 31 （四）技术人员如何向人工智能靠拢？
- 41 （五）打造基于机器学习的推荐系统
- 52 （六）打造机器学习的基础架构平台
- 63 （七）如何解决特征工程，克服工业界应用 AI 的巨大难关
- 79 （八）人工智能的下一个技术风口与商业风口

（一）第四范式戴文渊：构建商业 AI 能力的五大要素

作者 戴文渊



编者按

过去解决一个 AI 问题需要经历很长的步骤，而现在基于人工智能平台，可以把中间过程简化，只需要有清晰的业务目标 and 数据，直接实现最后的 AI 解决方案。

人类可能要穷其一生，才能从围棋一段升到九段，无论如何，这是个漫长的过程。然而，机器却能一下子到 20 段。机器是在什么时候超过人类的？如果能让机器写出 1000 万条以上的规则，在该领域机器就超越了人。

第四范式创始人、首席执行官戴文渊在《B-R-A-I-N：构建商业 AI 能力的核心要素》的演讲中说了上述观点，他表示，这个数字是理论依据的。

他介绍了相关的理论，介绍了人与机器思考的方式有什么区别，为什么有的地方机器能超过人类？需要哪五大前提，才能利用人工智能技术提升产品或者业绩。

演讲全文

我今天想分享的是 AI 开始火了以后，很多公司都会感兴趣怎么用 AI 的技术能够帮助企业提升业绩，让企业能经营得更好。

首先，让我们来思考一下如何判定机器的智能程度？



对生物而言，脑容量比较大的生物会比较聪明。包括人自己，从一个原始人进化到早期或者晚期的智人，再到现代人是伴随着脑容量的增加。

如何判断机器智力水平？

计算机没有脑容量可言，计算机必须要用数学的模型来建模。那业界是如何判断机器的智力水平呢？

有两位统计学家 Vapnik 和 Chervonenkis，他们提出了一个理论叫“VC 维”。VC 维反映了函数集的学习能力——维度越大则模型或函数越复杂，学习能力就越强。

当然大家可能觉得 VC 维太过数学了，那我们换一种方式理解——人怎么解决问题？



我们总结规则

比如牛顿三大定律，总结出了物理学三条规则。其实各行各业都有专家在总结业务的规律。

机器如果做这件事情，方式可能和人不一样，机器会把所有的物理现象按照速度区间分成 1000 万份或 3000 万份，提取更多的规则出来。

这种情况下，甚至机器会做得比牛顿更好，因为牛顿定律在高速情况下不成立，而机器可以在高速的区间上提取出不一样的规则，可以做得更好。

编者注：由牛顿定律为基础建立的经典力学，只适用于宏观、低速、弱力的场合。在高速（速度接近真空中光速 C ）、微观（粒子角动量接近普朗克常数 h ）和强力（维系原子核的约束力，强度 $\sim 10^4$ 牛顿）的情况下不适用，分别用相对论、量子力学和粒子物理描述。

其实，现在 AI 的商业应用场景就是在做这样的事情——利用机器看数据，提取出比专家规则更多的规则数。

2009 年我加入百度的时候，所有的搜索、广告都是专家规则系统，当时规则数写到了将近一万条，都是资深广告领域的业务专家写出来的。后来，我们用机器分析数据，最后把广告的规则写出了 1000 亿条，比人写的一万条做得更精细，所以最后带来了四年八倍收入的提升。

现在这样的方法已经应用到了各种领域。

比如金融领域的反欺诈场景，我们与一家银行合作，原来规则数大概有 1000 多条，后来利用机器学习，帮其找到 25 亿条规则，提升了预测成功率。此外，在个性化内容推荐领域，现在很多企业学习今日头条做个性化的内容分发，那如何才能做到个性化？本质上其实就是让机器写出来

的规则数足够多就可以了，过去由业务专家来定怎么分发，就没法做到个性化。

机器能可以在短时间内写出来的海量规则，专家可能要 30 年才能写出来。

这也就解释了为什么人在下围棋的时候从一段到九段是漫长的过程，而机器能一下子到 20 段。如果能让机器写出 1000 万条以上的规则，在该领域机器就超越了人。

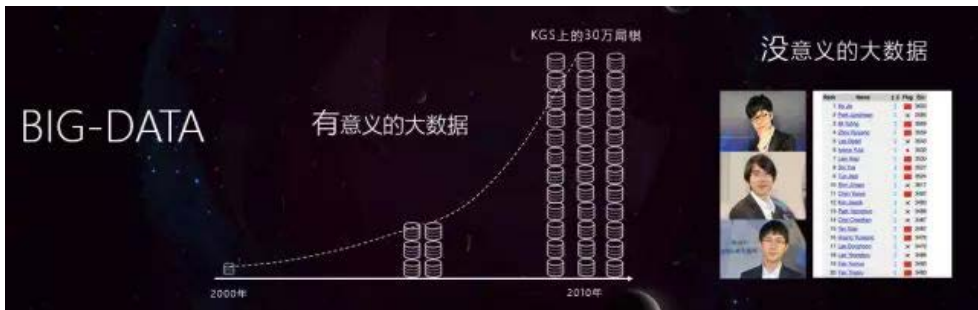
我们会觉得有的时候好像机器比人强很多，有的时候机器比人弱很多，这是为什么呢？其实就是因为有的领域机器能写出比人多得多的规则，但是有的领域不行。

让机器做得好的五个要素：B-R-A-I-N

怎么能让 AI 做好？我们总结了一下，一共有五个前提条件，概括来说叫做 BRAIN。

第一是大数据（Big Data）。

我要强调的并不是说数据量越大越好，而是看你有多少有意义的数据，或者说“过程数据”。



这就像围棋要看很多的棋局才能学会，并不是看每天的等级分排名或者新闻报道。

对于企业经营来说，我们要收集的是过程的数据，总的来说其实是请求、服务，再加上反馈的过程。

以推荐来说，这是一个访问，当用户访问了以后我们进行推荐，用户

可能会接受，可能不接受，这样的话就是一个过程数据。



需要多少个过程数据才能够让机器非常有把握做得比资深的业务专家好呢？1000 万。也就是说，训练出一个优秀的 AI 模型，需要 1000 万以上个样本。

为什么是 1000 万？其实有理论的支持，即获得图灵奖的 VALLIANT 引理，我们可以将它理解为机器模型的规则数量和数据量要相匹配。如果机器写出 1000 万条规则才能超越专家，就要 1000 万个数据。

第二是要有反馈（Response），而且反馈要数字化。

比如过去做内容推荐，目标是推荐“相关”的内容，但计算机无法衡量内容的“相关性”。所以现在，我们会把推荐目标拆解成点击率、阅读时长、转发、收藏、投诉，虽然说这些指标并不完全代表“相关性”，但每一个指标都可以被度量，计算机可以理解，可以优化。

最后实际结果表明，在近似的目标上走得足够远，反倒比执着在准确却无法达成的目标效果会更好。

第三是算法（Algorithm）。

原来我们需要算法专家做很深入的算法研究，但算法专家的数量毕竟是很少的。我们现在在探讨一些新的方式，不仅让算法专家，让普通的业务开发者也能操作算法。现在已经可以实现让一个业务专家经过一到两个月的培训，再加上一个机器学习平台，就能够做出机器学习的模型。

当然平台会封装掉有很多很复杂的工作，比如说要能支持万亿级别特征的算法、比如说实现自动特征工程。过去为什么做机器学习建模会难，就是因为要做特征工程，特征工程需要数据科学家磨炼三到五年的时间，

才能做出非常优秀的模型的效果。

现在，这个问题已经得到解决。年初，我们就在公司内部举行了一场建模大赛，所有参赛选手均为商务、行政、人力等非技术人员，最终超过 70% 的参赛组合 AUC（编者注：AUC 是衡量模型效果的专业指标，取值在 0 到 1 之间）成绩超过 0.8，这个成绩可以与从业多年的资深数据科学家媲美。

第四、基础架构（infrastructure）也是很重要的。

现在 AI 的流派主要是基于大数据和机器学习的，在大数据下其实是没有简单的问题的。

其实管理机器和管理人是一样的，当你管理 20 团队的时候，会出现一些问题，超过 100 人的时候又会出现一些问题，可能每天都会有员工请假等等。同样的道理，超过 100 台机器去跑一个任务，每次跑任务都会有机器宕机。超过 1000 台机器的时候，会出现各种分配任务的不均衡，有的机器可能在跑，做了很多的事，有的机器闲着没事干，这都是很复杂的。当数据量大的时候就有架构问题，这甚至是不亚于算法的难题。

1 万台机器遇到的问题则需要用冠军级的人去写出来的系统，还有支持万亿级的机器学习框架，难度也是很大的。我们也正在设计支持万亿级机器学习的架构，并把分布式计算、流式计算、高性能计算（超算）等能力都封装于产品中，确保在大数据的场景下，计算能力随数据量增加呈线性增长，实现系统时间成本与计算成本的可控。

最后一点很重要的就是需求（Needs），也就是要有清晰的问题定义（单一目标、有边界）。

现在 AI 只能解决单一领域的问题，一个有边界的问题。比如说 AlphaGo 下象棋没有下赢了，大家不会觉得它不行，因为你对它的期望就是下围棋。

但业务问题通常都是多目标优化的。实现多目标优化可以通过拆解的方式，就像做企业的 KPI 也需要拆解成不同的目标，这时候对各个目标进行单一目标优化，最后组合起来达到多目标优化的方式。

总的来说，其实在商业领域应用 AI，其实就是在业务目标清晰的前提下，把模型的维度做高。

说一个更容易理解的方式：过去由业务专家制定出成百上千条规则，今天则要由机器读数据，写出上千万甚至上千亿条规则。要实现这个目标，需要满足五个前提条件，但其实这是很有难度的。

第四范式解决的就是如何降低人工智能商用门槛的问题。过去解决一个 AI 问题需要经历很长的步骤，而现在可以把中间过程简化，只需要有一个业务目标和数据，可以直接做出 AI 解决方案。

五六年前想做个性化推荐，都是很大的工作量，但现在技术带来了开发成本的大幅度下降，以及企业运营效率的快速提升。

（二）AI 不是万能的，判别 AI 改造企业的 70 个指标

作者 王嘉俊



机器学习正在改变越来越多的行业，为了更好的应用机器学习，我们盘点了一些可以被机器学习改造的行业，以及这些行业对应的具体指标。

在看这些指标的时候，我们可以考虑下面的问题：

- 我们是谁解决了什么问题？
- 今天它是怎么解决的？
- 它会如何有效影响业务？
- 数据的输入是什么，这些数据来自哪里？
- 输出是什么？它是如何被使用的？（在线算法、静态报表等）

- 这是一个收入漏斗（省钱）还是收入增长（挣钱）的问题？

我们盘点的行业包括了计算广告、内容推荐、精准营销、金融、医疗健康、服务业、公司运营、制造业 8 个行业，包含了 70 个指标。由于时间和经验所限，我们没能覆盖到太多行业，每个行业的指标也有很多局限，但从中也许能给你一些启发，开发出更多适合机器学习的场景。

计算广告

1、客户细分

如果你能够定性的了解不同的客户群体，就可以给他们不同的市场方案（甚至由公司不同的部门提供）

影响：客户增长

2、预测终身价值（LTV，Lifetime Value）

如果你能够预测出高终身价值客户的特点，就可以进行客户细分，识别追加销售（upsell）的机会

影响：销售增长

3、客户份额估算

识别客户在不同类别上的花费情况，这将增加公司识别追加销售（upsell）和交叉销售的机会

影响：销售增长

4、产品组合

什么样的产品组合会产出最低的客户流失率？例如对于刚办理健身卡的人来说，30 岁以下的男私教 + 30 元的健康餐，是否会降低用户的流失率？

影响：用户维持

5、交叉销售 / 推荐算法

给你客户过去的浏览历史、购买历史和其他特征，他们未来最想购买的是什么？

影响：收入增长

6、追加销售

给你客户的特点，它在未来是否会追加购买？

影响：销售增长

7、渠道优化

给你这些客户的特征，最佳触达客户的方式是什么？

影响：客户增长，支出减少

8、折扣目标

通过折扣诱导消费的概率是多少？

影响：收入增长，客户满意度提高

9、再激活的可能性

对于已经停止使用的客户，再激活的可能性有多少？

影响：客户维持，客户满意度提高

10、搜索引擎优化和广告购买

为不同的关键字、广告位计算合适的价格

影响：优化推广效率

11、销售优先级

潜在客户关闭交易的可能性是多少？

影响：客户维持、收入增长

12、购物篮分析

通过分析用户的购物篮，提升推荐产品的购买率

影响：增加收入

13、最佳报价分析

分析过去的价格、销售数量和总销售额，得出最佳报价

影响：增加收入

内容推荐

14、电商推荐

根据用户和商品情况，推荐最合适的商品列表

15、好友推荐

根据用户的情况，给他推荐最适合的好友

影响：优化产品体验

16、音乐推荐

根据用户数据，给他推荐合适的音乐

影响：优化产品体验

17、主播位置推荐

根据用户数据、主播数据，确定主播页面的排列方法

影响：优化产品体验

18、新闻推荐

根据用户数据、内容数据，确定内容的推送和排列

影响：优化产品体验

19、餐厅推荐

根据用户数据、餐厅数据，确定餐厅的推送和排列

影响：优化产品体验

20、兴趣聚类

按照用户的兴趣，分成群组

影响：优化产品体验

精准营销

21、用户流失分析

识别出流失用户的特征，以支持公司进行产品调整，并通过在线算法对流失的用户提供帮助

影响：用户维持

22、库存管理

对于一件商品，客户需要多少？什么时候需要？通过预测以达到精益库存，同时防止缺货情况的出现

影响：优化管理效率，优化支出

23、价格优化

为每个时间、项目和商店进行优化

影响：提升收入

24、新店选址

根据商店情况、产品情况、地理位置情况等数据进行新店选址

影响：提升收入，风险管理

25、商店中的商品布局

怎样的布局能够提高销售额？

影响：提升收入

26、在商店的购物路线

组合不同的购物路线，得出最佳的方案

影响：增加收入

27、价格敏感度

每增加单位价格，对销售量有什么影响

影响：优化管理，增加收入

28、代理和分支业绩

如何根据历史数据，预测新代理的业绩水平？

影响：优化管理

29、什么产品组合更好？

什么样的产品组合会带来最多的销售数量？

影响：增加收入

30、供应商选择

我们在从最好的供货商进货吗？

影响：减少支出

31、邮件分组

对不同的客户邮件进行分组，选择不同的策略发送邮件

影响：优化客户体验，提升收入

32、地推人员管理（也适合很多垂直行业）

确定每天需要多少劳动力配给

影响：优化管理

金融

33、风险预估

给定借款人和贷款的特点，预测债务是否能得到回收？

影响：管理风险

34、财政或货币风险

我们需要多少的资金来满足这些需求？

影响：风险管理

35、新品种金融产品推广

通过分析相关金融产品的历史数据，一个新品种的金融产品最适合在哪些地方推广？

影响：收入增长

36、催收时间确定

在什么时间点进行催收，会有最好的效果？

影响：风险管理

37、欺诈检测

当系统预测交易可能涉及到欺诈时，决定是否要阻止一笔交易（例如信用卡欺诈）

影响：风险管理，减少支出

38、反洗钱

使用机器学习和模糊匹配来检测和反洗钱法相抵触的交易

影响：风险管理

医疗健康

39、索赔审核的优先次序

根据特征选择，确定哪些索赔应该由审核员手动审核

影响：提升审核效率，提升审核精度

40、医疗保险的欺诈分析

通过用户数据，分析医疗保险中的欺诈行为

影响：风险管理，减少支出

41、医疗资源配置

根据最初病人的访问，优化 / 预测手术室和床位

影响：优化医院管理，提升资源使用率，增加收入

42、实时预警

根据实时的患者数据，为医生提供警报

影响：风险管理

43、处方依从性

预测哪个病人更可能不遵循医生的处方

影响：提高就医效果

44、医生流失

医院希望保留那些多点执业的医生，怎么确定哪些医生更容易流失？

影响：维持组织稳定，防止核心资产流失

45、药物（剂量）有效性

预测不同类型、剂量的药物对治疗疾病的效果

影响：提升就医效果

46、再入院风险

根据患者的属性、病史、诊断和治疗，预测再入院的风险

影响：提升就医效果

47、识别产品包装盒中警告的生物标志物

在药品存储、流通过程中做到更为安全

影响：风险管理

48、药物 / 化学发现和分析

更准确、高效的发现新的药物、化学品的可能性

影响：创新发现

49、识别不良反应

例如在社交网络中监测药物会出现的早期问题

影响：风险控制

50、预测不同地区对不同药物需求

根据药物销量数据、不同地区的疾病数据、药店、医院数据等，确定药物的分发策略

影响：优化管理

51、通过不同的方法预测处方依从性并提醒患者

根据患者数据、药物依从性的历史等，预测哪些患者会不遵循医嘱

影响：优化客户体验

52、患者评价数据

识别患者对药物的看法，哪些是正面反馈、哪些是负面反馈，以及如何通过反馈提高药物的质量

影响：优化客户体验

服务业

53、酒店动态定价

根据酒店历史数据、日期、人流量等各种信息，确定酒店动态定价

影响：提升收入

54、酒店优惠券

分析不同的优惠政策会给酒店带来什么影响

影响：提升收入

55、酒店预订管理

预测一天当中会有多少人预约酒店

影响：优化管理

56、飞机调度

根据客流、天气状况，给出最佳的调度方案

影响：优化管理

57、旅游预测

根据旅游地的情况，分析是否要新增航线

影响：新产品开发

公司运营

58、简历筛选

根据候选人的特征，包括上一份工作、毕业学校、学历、年龄等进行简历筛选

影响：优化招聘效率

59、员工流失

预测哪些员工最有可能离开

影响：维持公司稳定

60、培训推荐

基于绩效考核数据，推荐特定的培训项目

影响：提升员工水平

61、可能性问题预测

尽早预测建设项目中可能会出现的问题

影响：风险管理

62、呼叫接听路径

基于呼叫者 ID 的历史、时间、呼叫的数量、拥有的产品、流失的风险、终身价值的多少确定呼叫的路径，这决定了每一个呼叫者的等待时间

影响：提升用户体验，保证关键用户的体验

63、呼叫中心的消息优化

把最合适的数据放在操作员的屏幕上

影响：提高操作员效率，提升用户体验

64、呼叫量预测

为了更好的确定呼叫人员的排班，进行呼叫量的预测

影响：减少呼叫中心的成本，优化管理

制造业

65、产量管理

通过监测土壤的传感器数据，预测农产品的产量

影响：优化管理

66、灾害预测

通过土壤数据、天气数据、农作物数据等，预测是否会发生农作物灾

害

影响：风险控制

67、故障预测

通过传感器数据来预测故障的发生

影响：优化管理

68、保修预测

预测产品是否需要保修

影响：优化管理

69、电力分配

根据地区、时间的不同，确定需要分配的电力

影响：优化管理

70、可能问题预测

尽早预测建设项目中可能会出现的问题

影响：风险管理

参考文章：<https://www.kaggle.com/wiki/DataScienceUseCases>

（三）机器学习的 MVP——用最小成本 验证 AI 可行性

作者 田枫



机器学习想要转化价值，最关键的一步是什么？

一个业务问题，埋坑无数，该如何巧妙转化，转变为机器学习的问题？

要平衡机器学习开发人力和时间成本，怎样才能找到最优产出比？

在范式大学在 AI 前线社群首节公开课上，针对以上问题，第四范式联合创始人，产品负责人田枫，基于丰富的专业从业经验，系统化梳理了解决之道。本文整合了直播干货，内容略有删减。

大家好，我是第四范式的联合创始人田枫，很高兴在这里和大家分享机器学习的 MVP 模型！

我们曾经在第四范式知乎专栏上发过一篇文章《年薪百万的机器学习专家，为什么不产生价值？》（<https://zhuanlan.zhihu.com/p/26435192>），文中的机器学习专家花了大量的时间搭建平台，做数据的清洗、处理与机器学习建模，却没有带来公司所期望的价值。问题出在哪里了呢？

基于第四范式在机器学习工业应用方面的大量成功案例和经验，我们今天就来分析一下，想用机器学习提升业务价值，在搭建平台、处理数据、训练算法之前，真正要做的第一步应该是什么？

我们今天不谈技术，不谈算法，不谈平台，但是今天聊的东西却是机器学习产生价值过程中，最关键的步骤之一。

这次分享我们会从几个方面分析这个问题：

- 第一，机器学习是不是万能良药？我们首先需要想清楚，机器学习作为特别牛的技术，它能解决什么样的问题。
- 第二，一个业务问题，可能有各种千奇百怪的坑，假设我们初步判定可以通过机器学习来解决他，那么应该通过怎样的转化，避开这些坑，把业务问题变成机器学习的问题。
- 第三，如果有一个好的可以转化成机器学习的问题，我怎么去设计机器学习的开发节奏，估算它的投入产出比，如何分阶段去推动问题的建模和应用。

这就是我今天要介绍的，机器学习的 MVP。

机器学习的最小可用产品

现在的互联网技术，接受的一个概念是最小可用产品，MVP，就是开发团队、设计团队用最小的成本代价，最大程度去验证产品的可行性。这个产品的可行性，是指这个需求是否真实存在，一个产品满足需求的方式是不是对的。

机器学习也是一样的，我们做机器学习的投入是长期的、持续的，带来的收入和回报也是巨大的，在开始之前，我们一定会希望以比较低的成

本知道：现在引入机器学习是否可以影响我们所面对的业务，产生价值的潜力有多大。

那么把一个业务真正用机器学习做之前，我们可以用两步，做一个机器学习的 MVP：

- 第一步：我们要选择正确的业务问题，并不是所有的问题都可以套在机器学习的框架里，有些适合机器学习解决，有些不适合机器学习解决。在任何的技术项目管理中，用差的方法解决好的问题，一定优于用好的方法解决错误的问题。
- 第二步：当我们找到一个机器学习可以解决的问题后，我如何通过最小的时间和人力代价，去证明机器学习可以解决它，带来满意的投入产出比。

选择正确的问题：从分类器开始

首先我们看看机器学习擅长解决什么问题。我举一个例子，就是周志华老师的西瓜书讲的例子，它很经典，也很简单，还很深刻，这个问题是说我要判断一个西瓜是好的还是不好的。

这个问题的业务场景是什么呢，一个西瓜，我怎么在不交易、不打开的情况下，就知道它是好的还是不好的。如果我知道，我就可以用同样的价钱买到更好的西瓜；而如果我是瓜商，有了一套标准之后，我就可以更好的管理我的货品。

回到这个问题，一个西瓜是好的还是不好的，这是典型的机器学习二分类问题。首先我们要找到，判断这个西瓜好不好有哪些可以用到的数据。我们不能把买卖西瓜之后的数据放进去分析，比如买了西瓜之后，我打开就知道好不好了，那么这个就没有价值。

所以我必须在不破坏西瓜的前提下，这时候能用到的数据是西瓜的产地、西瓜的纹路、重量、比重、敲击西瓜的声音是浑浊还是清脆、西瓜皮的质感等等，这些不打开西瓜的情况就知道的数据。

刚刚我们的目标已经讲得很清楚了，好的还是不好的，好的是 1，不

好的是 0，甚至我还可以定义一个评分，0 到 1 之间的一个数，但总体而言我可以设定一个机器学习的目标，我们称之为 Label。

选择正确的问题：真实世界模型

这看起来是一个很简单场景，好像一旦我们具备了这样的数据，就可以尝试建立机器学习模型了。然而在现实中，当我们想用机器学习来解决实际问题时，也会这么简单么？真实世界中往往是有很多陷阱的。这些陷阱可能有什么呢？

第一，西瓜好不好，是怎么定义的？是大？还是甜？皮厚不厚？瓢脆不脆？如果建立这个模型是为了西瓜的售卖，这些可能都是评价因素，模型学习的样本也都需要基于这个标准来建立。如果我们仅仅是基于西瓜大不大来定义样本，而实际的应用场景是综合判断西瓜好不好，那么可能会得不到想要的好的结果。

第二，西瓜好不好，是以什么为标准的？是用科学方法和仪器测量的？还是专家评测？如果是后者，评测者是同一个人么？如果是不同的人，大家对好西瓜的判断标准一样么？现实情况中，很可能是不一样的，那就要想办法消除 Label 的偏差。

第三，互联网的场景下，往往是需要满足所有人个性化的需求的，有些人喜欢甜的西瓜，有些人喜欢脆的西瓜，那将问题定义为分辨好的西瓜是否还合适？因为每个人对好西瓜的定义不一样，这个问题可能就转化为了推荐一个西瓜给一个用户，他（她）会不会喜欢。

第四，真实的应用环境是怎样的？假设我们需要一个在线实时的西瓜分类器，拿到西瓜那一刻马上判断它好不好，那是不是有些当时不能马上拿到的特征就不能用了？如果好瓜的判断标准在不断发生变化，或者瓜本身的特性在不断变化，模型还需要能够跟得上这个变化，基于新的数据和反馈做自我更新迭代，这就是我们搭建模型更新的方法。

可见，即便是简单的问题，我们都需要思考一下业务的方方面面，理清哪些因素，边际，个性化要素和基础设施是要考虑进去的。

选择正确的问题：业务问题的本来面貌

我们从西瓜还原到业务，任何一个业务能不能做机器学习，我们要看三个要素。

- 第一，这个业务的目标值是什么，它不一定是唯一的，但一定有主次。这个目标是否可以量化、收集反馈、客观观测的。什么叫客观观测，我说甜和你说甜，这个事情就可能不客观，那有没有一个客观的东西可以反馈。
- 第二，样本应该如何构造，样本不应该违反因果关系， $y=f(x)$ ， x 一定是我们业务场景中所能知道的信息。在西瓜的问题，就是打开西瓜之前我们能知道的信息，才可以作为 x 。同时，样本应该符合业务场景的真实情况，假设我们的业务是摸黑挑西瓜，我们看不见西瓜长什么样，我们只能敲，那西瓜的颜色就不能作为特征。
- 第三，样本的每一行代表什么意思，每一行应该代表西瓜的每次测量，然后才是选择哪些数据作为 x ，这些我们已经讲得很清楚。

当西瓜的问题说完后，我们来看看真实的业务问题是怎样的。

1. 点击率预估

比如说我们看到的推荐系统问题——点击率预估。

一个推荐系统的目标是什么？它的终极目标一定是用户体验，但这个目标很虚幻，我们要把它量化，变成一系列可以测量的数据，比如说点击、观看时长、购买、好评等，这些就是 y 。

然后我们看有哪些 x ，这些 x 代表的是我做出推荐排序的一瞬间，当客户请求时，在那个瞬间我知道的事情。我能知道客户的属性、特征，我能知道内容特征、上下文特征，但不知道最终这个内容是否有被展现和点击。我可以知道内容在这一瞬间之前被点击了多少次，但一定不是这个瞬间之后被点击了多少次，因为这样就穿越了。

有了 y 和 x ，就可以构造样本了。我的样本比如说，我给用户展现了 10 条推荐的内容，这个的反馈可能是点击和观看，那么每一次的样本展

现就是一个样本。

这里我们可以思考一个有趣的问题，当我们思考不同的特征对问题的影响时，比如说我们把展现作为一个样本，一个避免不了的问题是，我怎么知道这个内容是否被用户看到。

一种做法是我不去想这件事情，那么模型可能就是有偏的，比如说你认为这个样本没有被点击，但也有可能是没有被看到，但最理想的是把推荐到用户手机屏幕上的作为一条样本。

退一步，还有一个办法，就是把展现的位置补充回来，作为一个特征。然后请求的时候虽然没有这个特征，但是这个特征吸收了位置对于展现和反馈的偏差。

2. 简历匹配

再举一个场景的例子——简历匹配。简历匹配是什么意思？它其实想预测的是，我给企业推荐了一个简历，这个人有没有被企业聘用，这看起来是个简单的机器学习问题。但是回到业务场景思考，这个问题有没有这么简单？对于内容推荐来说，用户有没有点击这个内容，点击后看多久，都是用户单方面的选择。

但是简历有两个选择，第一个选择是企业通过面试、简历的选择，判断这个人是否适合企业。第二个选择是应聘者，他会不会去企业面试，而即便拿到了企业的 offer，会不会被打动加入企业。

所以这就变成了多点、双向的问题，在这样的情况下，就需要对问题进行拆解。我们可以不直接做个人被企业招聘的事情，而是分开来做，比如说企业会不会邀请这个人去面试，以及这个人会不会接受企业的面试邀请，这样就能把问题做的更好。

选择正确的问题：小结

总结一下我们刚刚所介绍的 MVP 第一步：做机器学习，首先不是着急去建机器学习的模型，而是认真思考这件事情的业务场景到底是怎么样的。

总结下来一个机器学习能解决的业务问题，有这么几个点：

- 第一它是否能转化成分类/回归的问题。
- 第二目标是否是容易获取、客观无偏差的数据。
- 第三是问题的预测目标，因果关系是什么，因果关系越简单越好，如果是多因多果，或者说描述“因”的相关信息不方便获取，那是否可以拆分成多个模型。特征往往是因的数据，或者是一些不是直接原因的数据，只要它不破坏这个因果关系。
- 第四是我们刚刚没具体去描述的，就是这个问题是不是一个真的业务需求。

一个真的业务需求是指，在我们用机器学习做出预测后，业务能否可以根据这个预测结果而受到影响？这个影响点是否足够清晰、有效？因为业务人员会用对业务影响的结果来评估我们项目的效果，如果我们预测的结果并没有有效影响业务，即使这个模型再好，也不会发挥作用。

比如说推荐系统，我预测了新的点击率后，可以按照点击率倒排来影响业务结果。但如果是游戏呢？如果我们预测这个人明天有 30% 的几率付费，我该如何影响到他，我能不能影响他？

所以你一定要思考，你的预测结果会怎么在业务中使用，这个使用不会对业务产生提升。如果你发现提升本身是很难的，那这本身就是个伪需求。然后你还需要思考，现在没有用机器学习的业务，它是用了什么方法和数据，现在的方法和数据有什么缺陷，哪些是机器学习可以帮到的。

当以上的问题都有清晰的回答后，这时候你就可以提出一个好的问题了。这时候你就成功 80% 了，而剩下的问题都相对简单了。

机器学习的投入

这就是我们 MVP 的第二步：在可控的人力、金钱投入下，构建一个有效的机器学习模型。

那什么是可控呢？1-3 人月的投入，更多就会风险太高。我们会期望获得什么提升？Case by case，不同的业务不一样，有些业务比如说广告，

1% 的收入就是好几百万，而有些问题可能要提升好几倍才有商业价值。

在机器学习成本分配中，最大比例在机器学习本身，调参、特征工程、模型评估、模型上线这些工程的事情占了大量的时间，而问题的定义、数据的采集占的时间非常小，我们认为这是有问题的。我们认为一个机器学习的项目，无论通过合作还是使用第三方平台的方式，应该把大钱花在采集好的数据，定义好的问题上去，甚至这要超过一半的时间。而另一半的时间，才是真正做机器学习模型的时间。

降低数据的成本

那我们怎么降低数据的成本呢？我给大家一些思考。

- 第一，除非必要，只使用采集好的数据。因为数据采集是一个有成本的事情，当一个公司的体系越复杂，它采集数据的成本就越高，所以除非这个数据采集起来很轻松，或者已经有了，你才会去考虑。
- 第二，如果你要开发新的数据，首先要考虑的是成本。开发新的数据源是有风险的。机器学习最怕的是说不清楚这是算法的问题，还是数据问题，还是问题定义的问题，所以让 MVP 环节中能出问题的环节越少越好。

前面我们介绍了问题定义的问题如何避免，而算法一般是不太容易出问题的，除非用错，而数据其实是很容易出问题的，所以我们尽量用简单、可靠、成熟的数据。

第三，我们讲到在建模的过程中，尽量使用成熟的工具。真正在数据处理，特征计算，和算法训练的这些过程中，大量的工作是可标准化，甚至可以用算法自动优化的，大量的坑其实也是可总结，或者说可以在产品引导中避免的。我们一直在研发的第四范式先知建模平台，就是在努力将建模过程中的 know-how 封装到产品中，让用户操作更简单，而且少踩坑，更有效的获得好模型。

总结一下，这一步总的思想是，能不制造新的风险点，就不制造风险

点，能降低不确定性就降低不确定性。

如何 Review 机器学习的模型？

好了，做好了前面介绍的两步，我们已经有了机器学习的 MVP，机器学习对业务的影响已经初见结论，如果业务有明显提升，那么祝贺你，找到了新的价值增长点，优化后一定还会有更大的提升潜力；而如果效果不明显，我们这里再给大家一些关于如何 review，如何检查 MVP 的建议：

首先要 Review 问题的方向是不是对的，模型的效果是否符合预期，模型的优化目标是否有明显的变化，比如优化的目标是西瓜好不好，优化之后是不是买到的西瓜好的变多了。

如果不是，那就是这个问题没有解决。那还会有什么原因？是不是指定了错误的目标，用在了错误的环境，或者数据有问题。其实说白了，要么是目标有错，要么是模型用错，要么是数据有问题，基于这 3 点来检查。

在现实业务中，解决了一个问题，有时也会带来新的问题。比如说新闻推荐的系统，现在点击的人多了，那么是不是由于推荐，新闻变得更加娱乐化了，是不是新闻的点击变得更集中化了，这可能并不是业务上非常希望的，需要继续想办法来优化。

第二步是 Review 数据，这些数据里面哪些起了关键作用，哪些数据是经验上认为会有作用的，但实际上没有的。那么重新检查这些数据，看是不是数据质量的问题，使得没有发挥应该发挥的作用。还可以看下一步我们可以引入哪些新的数据，数据最好一批一批引入，我加入一批，一次性开发结束。

第三步，当我 Review 上面的事情后，我要制定下一步的方案，往往是我会有新的、更多的数据。我也可能会调整目标，有可能是目标错了要改，也可能是增加目标，原来一个目标不够了，我要加入好几个新的指标，使模型变得更平衡。还有就是在工程上，看性能能不能优化等。

这就是我们这次分享的内容，我们怎么去推动一个机器学习的项目，问题如何定义，风险如何管理，等等。

（四）企业技术人员如何向人工智能靠拢？

作者 王嘉俊



相信看到这篇文章的朋友，几乎都想成为机器学习科学家。

事实上，绝大多数的付费课程，基本上都有完全免费的课程放在另一个地方。我们只是把这些信息整理好，告诉你在哪儿可以找到他们，以及通过什么样的顺序进行学习。

这样，哪怕你是还没毕业的大学生，或者是初入职场的工程师，都可以通过自学的方式掌握机器学习科学家的基础技能，并在论文、工作甚至日常生活中快速应用。

在这里我们推荐一份用户友好型的机器学习教程，你可以通过几个月

的学习成为机器学习科学家，完全免费。

一份用户友好型的机器学习教程

当你学习机器学习课程时，有没有被信息过载所淹没？

大部分的学习者都遇到了这个问题，这不是他们的错，因为绝大多数的机器学习课程都过于关注个别算法了。

没错，虽然算法很重要，但他们还是把太多时间花在了算法上。

以至于 你几乎很难在短时间内走完一遍机器学习的流程，从而感受到通过它解决具体数据问题的巨大兴奋。

这些机器学习课程关注于算法是因为它容易教。相比之下，如果机器学习老师要带你走一遍机器学习的流程，那么他需要搭建计算环境，完成数据采集、清洗、拆分，特征处理，模型调参和模型预测，甚至他还需要一个面向学习者的交互界面。老师哪有这么多的工具，与其手把手带着学生走一遭，还不如学习机器学习算法。

但这样的问题是，很难有人能坚持通过自学，成为一个卓越的机器学习科学家。哪怕他是数学博士，或者技术高超的程序员，都很容易陷在细节中而难以有具体项目实现的成就感。

这份教程将会带来完全不同的思路。它非常适合自学者，即便完全没有编程的基础，也能通过恰当的工具快速实现机器学习模型，解决工作、生活中遇到的具体问题。

值得注意的是，我们享用了世界顶级的机器学习资源，而不需要花费 1 分钱。

自我学习的方式

我们推荐通过 Doing Shit（不是技术术语）完成你的学习。

在这之前你也许已经学习过机器学习了，但从我和朋友们的经验来看，往往会被各种神秘的符号、公式、大量的教科书和论文整的晕头转向，然后再也不想碰这恼人的玩意了。

我们的方法会更加友好，它的学习过程就像小朋友学习一样，你会了解一些基础的知识（但不一定要完全弄懂），然后通过好用的工具快速实现出来就好了。而当你被建模出来的结果吸引，那时候我们才谈算法背后的数学逻辑和计算逻辑。

所以我们会在学习中做很多机器学习项目，这样的好处是当你面对一个工作机会时，你就是一个经验丰富的机器学习科学家了！

当然自学本身是需要自律的，这本教程将一直陪伴着你，以下是 4 个步骤。

1. 前提条件（不需要完全弄懂）

统计学、编程和数学（也可以不需要编程）

2. 海绵模式

把自己浸泡在机器学习的各种理论中

3. 目标实践

通过机器学习包实践 9 个有意思的题目

4. 机器学习项目

深度参与到感兴趣的项目和领域中

步骤 1：前提条件

机器学习之所以看起来很吓人，是因为总伴随着那些晦涩难懂的术语。实际上，即便你是中文系毕业的，也可以学好机器学习。不过，我们需要你在一些领域有基础的理解。

好消息是，一旦你满足了前提条件，其余的将会非常容易。事实上，几乎所有的机器学习都是把统计学和计算机科学的概念应用于数据领域。

任务：确保你了解基础的统计学、编程和数学

统计学：理解统计学、特别是贝叶斯概率对许多机器学习算法来说都是至关重要的。

免费的指南：[How to Learn Statistics for Data Science, The Self-Starter Way](#)

编程：懂得编程将会更灵活的应用机器学习。

免费的指南：[How to Learn Python for Data Science, The Self-Starter Way](#)

数学：对原始算法的研究需要线性代数、多变量计算的基础。

免费的指南：[How to Learn Math for Data Science, The Self-Starter Way](#)

你可以先看看这些教程，给你的机器学习道路打下知识基础。

步骤 2：海绵模式

海绵模式是尽可能吸收足够多的机器学习理论知识。

现在有些人可能会想：“如果我不打算进行原创性研究，为什么在可以使用现有机学习包的时候，还需要学习理论？”

这是一个合理的问题！

然而，如果你想把机器学习更灵活的应用于日常工作，学习一些基础理论还是很有好处的，而且你并不需要完全弄懂。下面我们会剧透学习机器学习理论的 5 个理由。

（1）规划和数据采集

数据采集真是一个昂贵和耗时的过程！那么我需要采集哪些类型的数据？根据模型的不同，我需要多少数据？这个挑战是否可行？

（2）数据假设和预处理

不同的算法对数据输入有不同的假设，那我应该如何预处理我的数据？我应该正则化吗？假如我的模型缺少一些数据，它还稳定吗？离群值怎么处理？

（3）解释模型结果

简单的认为机器学习是一个“黑盒子”的概念是错误的。是的，并不是所有的结果都直接可以解释，但你需要诊断自己的模型然后改善它们。我要怎么评估模型是过拟合还是欠拟合？我要向业务利益相关者怎么解释这些结果？以及模型还有多少的改善空间？

（4）改进和调整模型

你的第一次训练很少会达到最佳模式，你需要了解不同的调参和正则化方法的细微差别。如果我的模型是过拟合了，我该如何补救？我应该花更多时间在特征工程上，还是数据采集上？我可以组合我的模型吗？

（5）驱动商业价值

机器学习从来不会在真空中完成。如果你不了解武器库中的工具，就无法最大化发挥它们的效能。在这么多结果指标中，哪些是优化的参考指标？哪个更为重要？或者还有其他的算法会表现更好吗？

好消息是，你不需要一开始就知道所有问题的答案。所以我们推荐你从学习足够的理论开始，然后快速进入到实践。这样的话，你比较能够坚持下来，并在一段时间后真正精通机器学习。

以下是一些免费的机器学习资料。

机器学习视频课程

这是来自哈佛大学和耶鲁大学的世界级课程。

任务：完成至少一门课程

哈佛大学数据科学课程

端到端的数据科学[课程](#)。相比吴恩达的课程，它对机器学习的重视程度较低，但是从数据收集到分析，你可以在这里学到整个数据科学的工作流程。

课程主页：[斯坦福大学机器学习课程](#)

这是吴恩达的著名课程，这些视频说清楚了机器学习背后的核心理念。如果你的时间只能上一节课，我们建议这个。

课程主页：[吴恩达机器学习课程](#)

机器学习参考资料

接下来我们推荐行业中两本经典的教材。

任务：看这些 PDF 作为教科书

[An Introduction to Statistical Learning](#)

Gentler 在书里介绍了统计学习的基本要素，适合所有机器学习的学习者。

[Elements of Statistical Learning](#)

严格的介绍了机器学习理论和数学，推荐给机器学习的研究员。

PDF 地址：

成功的关键

以下是每个步骤成功的关键。

A：注重大局，总是问为什么

每当你被介绍一个新概念时，问一句“为什么”。为什么在某些情况下要使用决策树而不是回归？为什么要规范参数？为什么要拆分数据集？当你了解为什么使用每个工具时，你将成为真正的机器学习从业者。

B：接受你不会记得所有学过的东西

不要疯狂的做笔记，也不要每个课程都复习 3 次。在自己的实际工作中，你会经常需要回过头查看。

C：继续前进，不要气馁

尽量避免在一个话题上拖太久的时间。即便是对于机器学习教授来说，有些概念也很不好解释。但是当你在实践中开始应用时，你会很快就懂得概念的真实含义。

D：视频比教科书更有效

从我们的经验来看，教科书是很好的参考工具，但它很难坚持。我们强烈推荐视频讲座的形式。

步骤 3：有目的实践

在海绵模式之后，我们会通过刻意练习的方式磨练技能，把机器学习能力提高到一个新水平。目标包括三个方面：

1. 实践完整的机器学习流程：包括数据收集、清洗、预处理，建立模型，调整参数和模型评估。

2. 在真实的数据集中练习，逐渐建立哪种模型适合哪种挑战的直觉。
3. 深入到一个具体主题中，例如在数据集中应用不同类型的聚类算法，看哪些效果最好。

在完成这些步骤后，当你开始解决大型项目时就不会不知所措了。

机器学习的工具

为了快速实现机器学习模型，我们推荐使用现成的建模工具。这样的话，你会在短时间内练习整个机器学习的工作流程，而无需在任何一个步骤花费太多时间。这会给你非常有价值的“大局直觉”（Big Picture Intuition）。

Python: Scikit-Learn

Scikit-learn 和 Sklearn 是通用机器学习中 Python 的黄金标准库，它具有常规算法的实现。

R: Caret

Caret 为 R 语言中的模型包提供一个统一的界面。它还包括了预处理、数据拆分、模型评估的功能，使其成为一个完整的端到端解决方案。

实践数据集

学习了工具后，你还需要一些数据集。数据科学和机器学习的艺术，很多都在于解决问题时的几十个微观决定。我们会在不同的数据集中看到建模的结果。

任务：从以下选项中选择 5 到 10 个数据集。我们建议从 UCI 的机器学习库开始，例如你可以选择 3 个数据集，分别用于回归、分类和聚类。

- 在进行机器学习工程的时候，想想以下问题：
- 你需要为每个数据集执行哪些类型的预处理？
- 你需要进行降维操作吗？你可以使用什么方法？
- 你可以如何拆分数据集？
- 你怎么知道模型是否出现“过拟合”？
- 你应该使用哪些类型的性能指标？

- 不同的参数调整会如何影响模型的结果？
- 你能够进行模型组合以得到更好的结果吗？
- 你的聚类结果和直观的相符么？

UCI 机器学习报告

UCI 机器学习报告采集了超过 350 个不同的数据集，专门为机器学习提供训练数据。你可以按照任务搜索（回归、分类或聚类），也可以按照行业、数据集大小搜索。

地址：<http://archive.ics.uci.edu/ml/>

Kaggle

Kaggle.com 以举办数据科学比赛闻名，但是该网站还拥有超过 180 个社区数据集，它们包含了有趣的话题，从用户宠物小精灵到欧洲足球比赛的数据应有尽有。

地址：<https://www.kaggle.com/datasets>

Data.gov

如果你正在寻找社会科学或者与政府有关的数据集，请查看 Data.gov。这是美国政府开放数据集合，你可以搜索超过 190,000 个数据集。

地址：<https://www.data.gov/>

步骤 4：机器学习项目

好了，现在到了真正有趣的部分了。到目前为止，我们已经涵盖了前提条件、基本理论和有目的实践。现在我们准备好进入更大的项目。

这一步骤的目标是将机器学习技术整合到完整的、端到端的分析中。

完成一个机器学习项目

任务：完成泰坦尼克幸存者挑战。

泰坦尼克号幸存者预测挑战是一个非常受欢迎的机器学习实践项目，事实上，这是 Kaggle.com 上最受欢迎的比赛。

我们喜欢以这个项目作为起点，因为它有很多伟大的教程。你可以从中了解到这些有经验的数据科学家们是怎么处理数据探索、特征工程和模

型调参的。

Python 教程

我们真的非常喜欢这个教程，因为它教会你如何进行数据预处理和纠正数据。教程由 Pycon UK 提供。

教程地址：<https://github.com/savarin/pyconuk-introtutorial>。

R 教程

在 R 中使用 Caret 包来处理几个不同的模型。本教程很好总结了端到端的预测建模过程。

教程地址：<http://amunategui.github.io/binary-outcome-modeling/>

这是一个“不负责任”的快速教程：仅仅是个教程，跳过了理论讲解。不过这也很有用，而且它显示了如何进行随机森林操作。

教程地址：<http://will-stanton.com/machine-learning-with-r-an-irresponsibly-fast-tutorial/>。

从头写个算法

为了对机器学习有更深入的理解，没有什么比从头写个算法有帮助了，因为魔鬼总是在细节里。

我们建议从一些简单的开始，例如逻辑回归、决策树或者 KNN 算法。这个项目也为你提供了一个将数据语言翻译成程序语言的实践。当你想把最新的学术界研究应用于工作时，这个技能将会十分方便。

而如果你卡住了，这里有一些提示：

- 维基百科有很多好资源，它有很多常见算法的伪代码。
- 为了培养你的灵感，请尝试查看现有机器学习软件包的源代码。
- 将你的算法分解，为采样、梯度下降等编写单独的功能
- 从简单开始，在尝试编写随机森林前，先执行一个决策树。

选择一个有趣的项目或领域

如果你没有好奇心，你是很难学好的。但目前为止，也许你已经找到了想坚持下去的领域，那么开始建模吧！

老实说这是机器学习最好的部分了。这是一个强大的工具，而一旦你开始理解，很多想法都会主动找上门。

好消息是，如果你一直在跟踪，也准备好从事这份工作，那么你的收获会远超你的想象！

我们也推荐了 6 个有趣的机器学习项目。

地 址：<https://elitedatascience.com/machine-learning-projects-for-beginners>。

恭喜你到达了自学指南的终点

这里有一个好消息，如果你已经遵循并完成了所有任务，那么你在应用机器学习上将会比 90% 自称是数据科学家的人更好。

而更好的消息是，你还有很多东西要学习。例如深度学习、强化学习、迁移学习、对抗生成模型等等。

成为最好的机器学习科学家的关键是永远不要停止学习。在这个充满活力、激动人心的领域，开始你的旅程吧！

该教程由 EliteDataScience 提供，我们翻译了这份教程，略有改动。这是原文链接：<https://elitedatascience.com/learn-machine-learning>。

（五）用户画像、协同过滤过时了？打造基于机器学习的推荐系统

作者 周开拓



今天想和大家分享，如何使用大规模机器学习解决真实的业务问题。我们今天会以机器学习中的一个典型场景为例来讲解，即基于大规模机器学习模型的推荐系统。

推荐系统的本质是什么？

比如说我们看到手机淘宝首页，往下一拉，就能看到各种各样推荐的商品；比如说百度，它会给我们推荐广告，在某种程度上他的工作方式也很像推荐系统；再比如说今日头条，今日头条从数十万的新闻中选出会被我们看到的数十个新闻，这也是推荐系统。

尽管我们在生活中会已经见过非常多的推荐系统，但是在用机器学习搭建推荐系统之前，我们还是应当先思考一下，推荐系统要解决的到底是个什么样的问题？

推荐系统在本质上是一个信息检索的系统。它和搜索最大的区别是，搜索是主动式的，根据关键词和引擎参数、搜索引擎召回、机器学习排序，决定给你看到的是哪些内容。而我们看到的推荐系统，在大多数情况下是没有主动输入的（有时会有一些简单的反馈动作），是被动出现的。

推荐系统是利用上下文，根据当前用户所处的环境，根据信息的特点来决定给你推荐什么内容和商品。而在我们进一步去想之前，我们要问自己一个问题，就像上节课田老师讲的一样，推荐系统的目标是什么，什么才是一个好的推荐系统，要优化的指标是什么。

推荐系统的指标是什么？

推荐系统是个产品，产品当然是想方设法让用户去喜欢的，或者至少是不讨厌的。因而，我们需要把喜欢和讨厌这两件事情定义出来。同时我们毕竟不是用户肚子里的蛔虫，我们只能用我们可以测量到的数据来描述喜欢和讨厌两件事情。并用这些数据来决定我们做什么和不做什么。

比如说：我是个电商，用户表达对一个推荐商品喜欢的方式是：点击、收藏、加购物车、甚至购买下单、分享到社交平台上等等。用户讨厌一个推荐商品的方式，就是会投诉、会提意见。因而我们要预防一些很可能会让用户讨厌的推荐结果：比如说推荐成人用品和内衣，尤其是在上班时间；比如推荐用户刚刚买过的商品，等等。

我们一定能为一个推荐系统去定义指标，我们可以给这些指标分轻重缓急，看能用什么顺序实现。现在我用的指标可能有点投机取巧，我用的是点击率。而真实的指标考虑的是很多的，仅仅考虑点击率的模型，可能会出现标题党，如果是电商就可能会出现一堆 9 块 9 包邮，这可能不是我们业务想要的。

另外即使只考虑点击率，我们也知道其实我们推荐的是一个列表，列

表的质量不完全是由单一的商品决定的，而是整个列表的组合、顺序、多样性所决定的。所以真实的业务中，我们会考虑用更复杂的目标，比如 MAP 来评价一个推荐列表的质量。

但没关系，今天我们就用点击率作为试点，介绍如何用机器学习来搭建推荐系统的完整过程。

选择推荐系统的 y 和 x

第一步，我们已经知道机器学习模型需要预测的就是优化目标，点击率；那我们把用户的点击行为需要记录下来。这样一来，对于机器学习来说，我们已经有 y 了。

第二步，我们需要定义好 x ，也就是特征。

一般来说推荐系统的特征体系由 3 个部分组成：用户特征、内容特征、上下文特征。

- 用户特征：包括但不限于用户姓名、性别、年龄、注册时间、收货地址、常用区域等用户特征。
- 内容特征：包括但不限于以及商品、内容的标题分词、内容的 TF-IDF、内容来源、内容渠道、内容生产者等等。
- 上下文特征：是代表用户当前时空状态、最近一段时间的行为抽象的特征。比如说用户当前的 GPS 坐标，大家可能觉得奇怪，GPS 坐标怎么用来推荐呢？其实很简单，地球一圈是 4 万公里，GPS 一圈是 360° ，一度大概是 100 公里。如果我们把 GPS 坐标保存到小数点后一位，组合起来，这样的特征就是 10×10 公里的格子，这就代表了一个有泛化能力的用户的位置。

位置是一个非常强的特征，如果我们更进一步，做到了 1 公里，显然我们可以相信，在中关村地区，大家的偏好是有共性的，而在金融街，大家的喜好也是有共性的。当大家的数据足够多的时候，落在同一个格子里的人会非常多，GPS 就会成为非常重要的特征。

另外是 IP 地址，比如最近浏览的内容、最近购买的商品，这些都会

构成上下文特征。所以我们就是在用户特征、内容特征和上下文特征的基础上，预测用户对当前内容的点击率。

推荐系统的样本构造和数据拼接

一个成熟的推荐系统，它可能有非常复杂的样本构造方法，今天用了比较简单的方法，让问题变得简单。另外，一个成熟的推荐系统，它可能会有多个指标和业务边界条件。

那么接下来：基于已知的 x 和 y ，我们要为机器学习构造样本。什么是一条样本？一个样本代表机器学习预测的一个最小粒度的事件。当你把一条内容展现给用户，用户点击或不点击，这就代表了一个最小粒度的事件，就是一条样本。再比如说我们给用户展示了 10 条新闻，用户对应每个新闻点击或者不点击，就是 10 条样本。

在样本采集后，就要考虑数据怎么收集和拼接了。在拼接的时候要注意的是，假如是为了优化点击率，我不光要把用户特征、上下文特征收集起来，我还要把点击率拼回到当时那一条样本请求上去。所以系统一定要有这样的考虑，记录下时间和拼接的 ID，同时还要考虑刚才说的三类特征是处于实时变化当中的，日志也是实时产生的，而不是后面去拿的，因为这样做很可能会出现数据问题。数据一旦出问题，是非常难以 debug 的。给大家举个栗子，有一家公司，BAT 之一，他的推荐系统过去几年 85% 的效果提升来源是把之前有问题的数据给修复了，做对了。

推荐系统的场景思考

样本构造还需要考虑场景的问题，比如说我们会遇到一个问题，屏幕的大小是不一样的，同样展示 10 条新闻，我怎么知道用户有没有看到它。如果没有看到就不应该作为一条样本。这时候就有两种解决方案，第一种解决方案是把用户真正看到的纳入进来，因为前端是你设计的，所以你会知道哪些内容是用户的可见范围内。当然这会让客户端变得更重一些。

第二种是一个比较简单的方法，把内容的位置作为一个特征。因为我

们知道，同样是一屏幕展现 10 行内容，即便是一样的内容，用户也会选择一个他舒服的位置去点，这个可能是偏中上的位置。所以当新闻在第三个位置被点击的时候，这可能是一个容易被点击的位置，但不一定代表这个新闻比其他新闻要好。那我们怎么办？我们就要通过某些手段，把这些偏置吸收掉，所以我们会把位置、屏幕大小等作为特征，通过特征工程的方法来吸收这个偏差，变成无偏的模型。

这时候有些人可能会问，这不是穿越吗？因为在给出预测的时候，是不知道内容最终的位置信息的。但这相当于把偏差的锅由位置来背了，这是机器学习推荐系统中的一个策略。

刚才我们已经构造好特征了，现在给大家讲怎么建模。大家可能会认为，前面的部分是快的，真正做机器学习，做特征工程、模型调参等，这些是慢的。但是今天我们会看到，在成熟的工业界里面，其实前面要花的时间会多很多，后面的内容在成熟的工具下会变得简单。

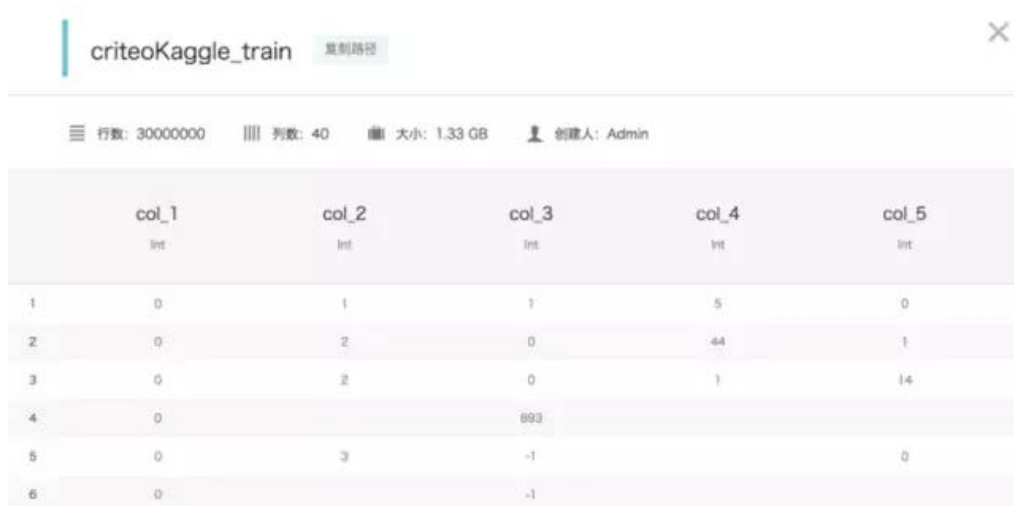
在先知上完成推荐系统的建模历程

我们会从一个真实的案例出发，虽然我们做了很多的推荐系统的案例，但毕竟不能把客户的数据给大家看，所以我们用了一份公开的数据，这份数据和我们之前讲的场景是相似的。这份数据来自于 Kaggle，叫做 Criteo 点击率预估比赛。

数据预览

首先我们看数据的样子，第 1 列 col_1 代表的是广告有没有被点击，1 代表被点击，0 代表没有被点击。然后我们看第 2 列到第 14 列，都是数值型的特征，因为这份数据已经被匿名化了，所以我们可能也不知道这些数值代表什么意思，也许是这个用户的 PV，或者标签的权重，不过我们也不需要知道。然后看第 15 列到 40 多列，这些都是离散的特征，这些特征都做了哈希化，都做了匿名处理。

这个数据有 3000 万行 40 多列，按照我们传统的做法，进行特征工程



	col_1	col_2	col_3	col_4	col_5
	int	int	int	int	int
1	0	1	1	5	0
2	0	2	0	44	1
3	0	2	0	1	14
4	0	993	0	0	0
5	0	3	-1	0	0
6	0	-1	0	0	0

以及 one-hot 编码后，会有 4000 多万个特征。真实的业务数据中，训练数据体积会更大，往往达到上亿，同时原始特征数量会达到上百，因为为了更好的个性化效果，我们会使用诸如 GPS 坐标、手机型号、ip 地址、最近浏览内容等等精细化的特征，并进行非常极致的特征工程，这样的模型在特征工程之后的特征数会达到数亿甚至几百亿。这样规模的机器学习训练，挑战的不仅是算法，更是如何在成本可承受的计算资源上进行训练和实时预估。

开发这样一个规模的可以并行运行的系统的挑战更加大，即使 BAT 这样的大公司也会养一个百人的团队，只为了做好机器学习模型训练和预估的工程实现。下面我们会看到利用第四范式的先知平台去做这件事情，会大大降低我们开发和运维一个在线机器学习系统的成本，让我们更加聚焦在业务本身。下面我们会看到在先知平台上对这份数据的建模会非常简单。

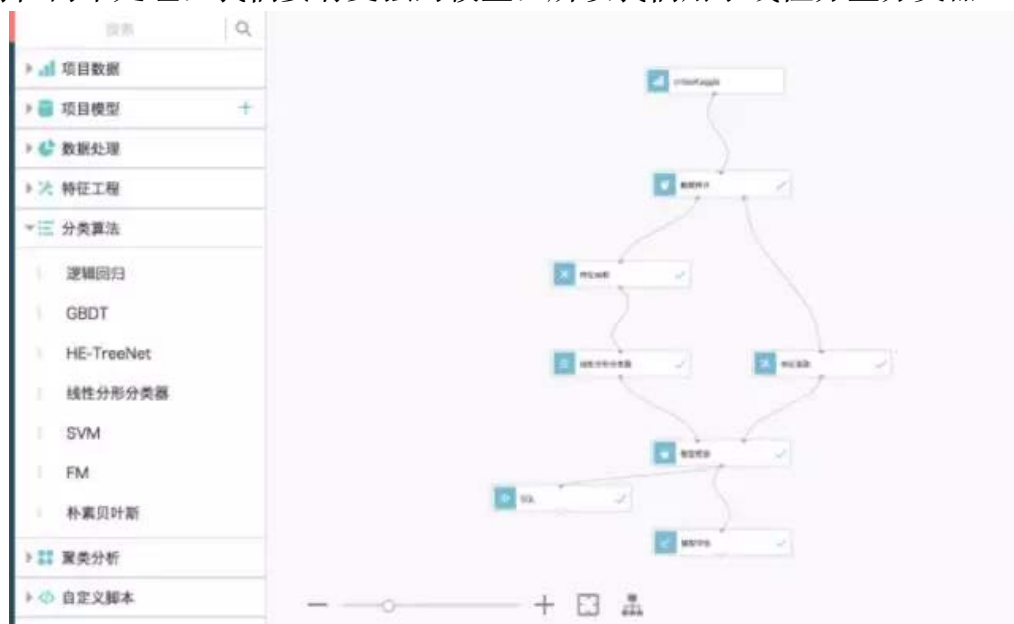
数据拆分

首先我们把这份数据拆成了训练集和测试集，以 9:1 的方式。当然这可能是不太合适的拆分方法，因为真正训练一个机器学习模型，它的拆分是按照时间排序后再拆的，就是用前面的时间来训练模型，然后用这个模型来预测点击。这样训练和测试在时间上是正交的，那么模型如果在这

种实验设计下有好的效果，这个效果就会有时间平移性，上线后就会有好的效果。当然，为什么我们在这里没有使用按照时间排序拆分的方式，是因为参考了一篇论文的做法（<https://arxiv.org/abs/1703.04247>），这样同样的做法结果可比。在真实的业务中，不建议大家按照这种方式来处理数据。

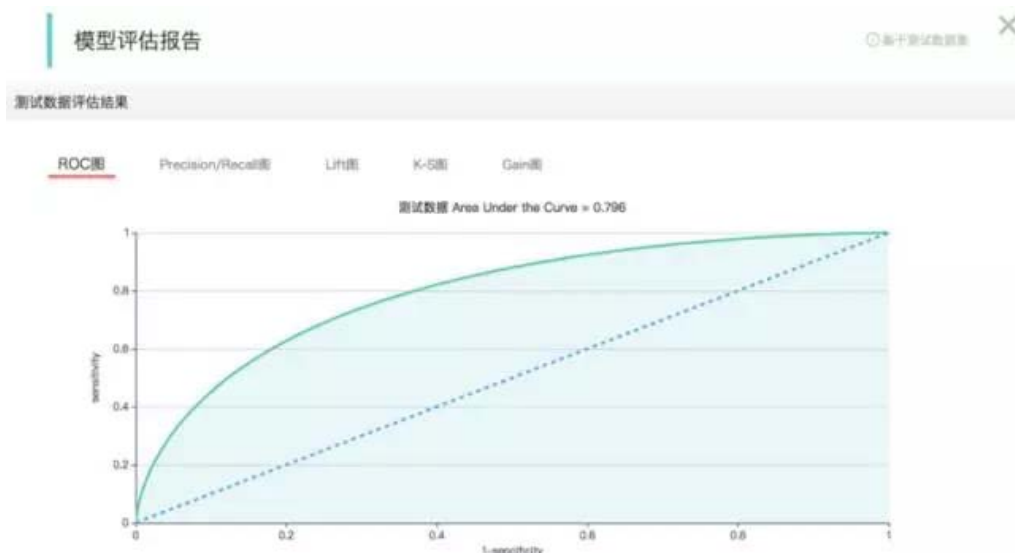
特征工程、模型选择和评估

然后就是做特征工程了，一般来说大家可能觉得很难，但只需要告诉电脑哪个是 Label，其他的直接默认配置就好了。但天下没有免费的午餐，特征简单处理，我们要有更强的模型，所以我们用了线性分型分类器。



一般来说，推荐系统中会有离散特征和连续特征，当我们用逻辑回归这种宽的离散线性模型的时候，我就会遇到一个很大的问题，就是我对特征进行分段，这样才能学到连续特征的非线性结构。比如说用户年龄对点击率的影响，它并不是线性上升的，它可能会上升到 30 岁，然后就下降了，甚至有更复杂的模式，这些是非线性的特性。所以我们可以用机器对连续的特征做自动搜索分段，这样的预处理效率就大大提高了。线

性分形分类器就是做的这样的事情，基本上我花了不到 5 分钟，就跑出了 0.796 的 AUC。



AUC 代表了一个模型对样本的排序能力。如果这是一个完美的排序，点击率高的 100% 排到了点击率低的前面，那 AUC 就是 1。如果是随机瞎排，那 AUC 就是 0.5。所以 AUC 越接近于 1，就代表了整体的排序能力越强。现在的 AUC 是 0.796，比论文的 0.801 稍微差一点，我只做了这么点就到了这个成绩，我觉得还是可以的。

当然这里进一步说一下，评估上线的时候，不能只是看 AUC。例如说，我要看的是 User AUC，或者每一刷的 AUC，因为模型的排序能力，一部分来自于对每个用户个性化的识别能力，另一部分来自于对用户本身的识别能力。比如说有些用户就是什么都爱点，什么都点，而有些用户基本只看标题不点开。我们把这样的人分开，对个性化推荐是有帮助的。所以我最终应该看的是，在每一个用户身上的 AUC，甚至是每一次展现的 AUC，当然先知也提供了工具，就不多赘述了。

但我还是想让这个模型更好，怎么办呢？在做推荐系统的时候，特征工程最重要的处理方法是特征组合。比如说有两个特征，一个是性别，另一个是新闻的色情等级。我们可能会注意到，男同志比较喜欢火爆的新闻，女同志可能正好相反。我把色情等级作为一个特征，可能两边的喜好不同

导致最终这个特征整体对点击率的影响和一个随机数一样，它就不是好的特征，没有预测能力。

但如果我把这些特征组合起来，我就对这些空间有更细致的分割能力，我就会做出很好的效果。性别、用户 ID 和新闻色情等级组合起来，是新闻推荐非常有用的特征。进一步说，假如我们有 100 个特征，那么特征两两组合的空间，就是一万个，这是个很大的工作量。

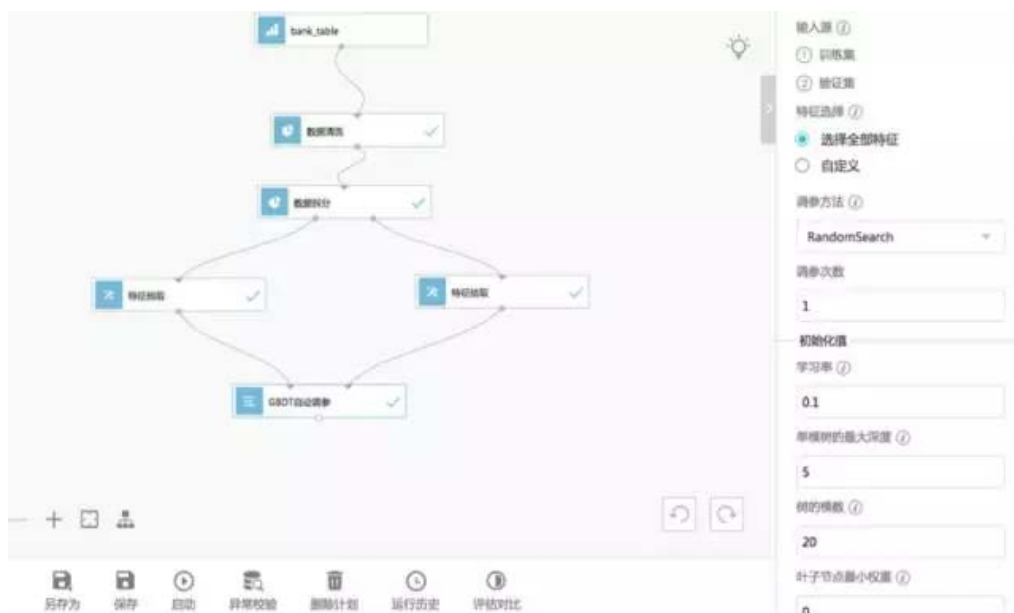
传统的手段是，通过业务经验和实验去筛选好的组合。但是工具可以帮助我们做得更快更好，先知的 FeatureGo 功能，可以通过一系列的配置和优化目标设定，自动搜索出好的特征组合，大大节省我们的工作时间，让人的工作由机器来完成。然后我就跑了 FeatureGo，找到了 18 和 28、33 和 34 这些组合特征就不错。而如果计算资源足够，那你还可以搜索 3 阶的特征，这是由人来做非常难的事情。

```
57 f_combine_18_28=discrete(combine(col_18,col_28))
58 f_combine_20_37=discrete(combine(col_20,col_37))
59 f_combine_18_28_34=discrete(combine(col_18,col_28,col_34))
60 f_combine_20_31=discrete(combine(col_20,col_31))
61 f_combine_29_40=discrete(combine(col_29,col_40))
62 f_combine_18_20_28=discrete(combine(col_18,col_20,col_28))
```



在用机器完成自动特征组合后，调参其实也可以自动化的。我们知道

机器学习就像炼丹，炼丹就是调参，调参就是在参数空间里，根据我们的经验去搜索一下，看什么参数是好的。我们也有好的工具，可以自动搜索到好的参数。当把这些设定好之后，我就去睡觉了。第二天醒来后发现，这个模型的 AUC 达到了 0.802，已经超过了论文的效果。由于这篇论文还是今年比较有名的论文，我还是很满意的。



模型上线

设计好模型后，一般是用 Restful API 的模式服务的。比方说根据这个 API，就可以给不同的内容进行打分，打分后根据倒排后的结果，响应推荐的内容。

推荐系统相关组件

当然，一个完整的推荐系统，不仅仅需要机器学习排序，还需要相应的组件。比方说我要有物料库，把商品的特征维护起来，这需要一个高性能的缓存和数据库，能够增删查改，能够进行特征的初步生成。我还需要一个日志系统，日志系统通过唯一性的标识，把实时的请求和后续的反馈 Label 记录并拼接起来。我还需要一个自学习的系统，机器学习的模型每



天都是要更新的，如果拿 7 天前学到的模型去推今天的商品，那岂不是用前朝的剑斩本朝的官，效果一定不好，所以我们还需要自学习，或者是增量的在线学习，来保证模型捕捉到最新的用户偏好和市场情况。

然后我们还需要后续的预测，比方说我的内容有好几十万，我并不是拿好几十万给机器学习去预估，我会用启发式的方法，比如说 CF、热度、Tag 匹配等召回策略先生成候选集，然后才进入到机器学习排序。召回策略同样使得我们有更大的能力去影响机器学习排序的结果，比如我们可以过滤掉一些我们明知道不好的内容或者增加我们认为好的内容的权重或出现比例。在机器学习排序后，我们也需要对结果进行去重、多样化和随机化，最后才做成一个好的推荐系统。

今天和大家介绍的是，我们如何利用机器学习去搭建一个推荐系统的排序环节。总结起来是这么几个点：第一个是如何使用机器学习来剖析一个问题，我们用了推荐系统的例子。第二个是我们如何构造一个推荐系统的样本、数据并进行建模，当我们有一个非常好的机器学习工具的时候，我们可以把精力聚焦在业务上，在怎么找到好的数据上，以及在怎么定义好的目标和规划上。第三我们描述了机器学习系统是如何和其他系统发挥作用的，机器学习就像发动机，汽车当然需要发动机，但只有发动机是跑不起来的，你还需要周边的配件，这是系统化的工程。在这方面我们已经做了一些工作，我们既有发动机，也就是先知平台，大家可以在这里试用 <https://prophet.4paradigm.com>，我们也有整车，就是整个推荐系统的解决方案。我们很高兴和大家分享这样的技术和能力，谢谢。

（六）打造机器学习的基础架构平台

作者 陈迪豪



基础架构（Infrastructure）相比于大数据、云计算、深度学习，并不是一个很火的概念，甚至很多程序员就业开始就在用 MySQL、Django、Spring、Hadoop 来开发业务逻辑，而没有真正参与过基础架构项目的开发。在机器学习领域也是类似的，借助开源的 Caff e、TensorFlow 或者 AWS、Google CloudML 就可以实现诸多业务应用，但框架或平台可能因行业的发展而流行或者衰退，而追求高可用、高性能、灵活易用的基础架构却几乎是永恒不变的。

Google 的王咏刚老师在《[为什么 AI 工程师要懂一点架构](#)》提到，研

究院并不能只懂算法，算法实现不等于问题解决，问题解决不等于现场问题解决，架构知识是工程师进行高效团队协作的共同语言。Google 依靠强大的基础架构能力让 AI 研究领先于业界，工业界的发展也让深度学习、Auto Machine Learning 成为可能，未来将有更多人关注底层的架构与设计。

因此，今天的主题就是介绍机器学习的基础架构，包括以下几个方面：

- 基础架构的分层设计；
- 机器学习的数值计算；
- TensorFlow 的重新实现；
- 分布式机器学习平台的设计。

第一部分，基础架构的分层设计

大家想象一下，如果我们在 AWS 上使用编写一个 TensorFlow 应用，究竟经过了多少层应用抽象？首先，物理服务器和网络宽带就不必说了，通过 TCP/IP 等协议的抽象，我们直接在 AWS 虚拟机上操作就和本地操作没有区别。其次，操作系统和编程语言的抽象，让我们可以不感知底层内存物理地址和读写磁盘的 System call，而只需要遵循 Python 规范编写代码即可。然后，我们使用了 TensorFlow 计算库，实际上我们只需调用最上层的 Python API，底层是经过 Protobuf 序列化和 swig 进行跨语言调研，然后通过 gRPC 或者 RDMA 进行通信，而最底层这是调用 Eigen 或者 CUDA 库进行矩阵运算。

因此，为了实现软件间的解耦和抽象，系统架构常常采用分层架构，通过分层来屏蔽底层实现细节，而每一个底层都相当于上层应用的基础架构。

那么我们如何在一个分层的世界中夹缝生存？

有人可能认为，既然有人实现了操作系统和编程语言，那么我们还需要关注底层的实现细节吗？这个问题没有标准答案，不同的人在不同的时期会有不同的感受，下面我举两个例子。

在《为了 1% 情形，牺牲 99% 情形下的性能：蜗牛般的 Python 深拷贝》这篇文章中，作者介绍了 Python 标准库中 `copy.deepcopy()` 的实现，1% 的情况是指在深拷贝时对象内部有可能存在引用自身的对象，因此需要在拷贝时记录所有拷贝过的对象信息，而 99% 的场景下对象并不会直接应用自身，为了兼容 100% 的情况这个库损失了 6 倍以上的性能。在深入了解 Python 源码后，我们可以通过实现深拷贝算法来解决上述性能问题，从而优化我们的业务逻辑。

另一个例子是阿里的杨军老师在 Strata Data Conference 分享的《Pluto: 一款分布式异构深度学习框架》，里面介绍到基于 TensorFlow 的 `control_dependencies` 来实现冷热数据在 GPU 显存上的置入置出，从而在用户几乎不感知的情况下极大降低了显存的使用量。了解源码的人可能发现了，TensorFlow 的 `Dynamic computation graph`，也就是 `tensorflow/fold` 项目，也是基于 `control_dependencies` 实现的，能在声明式机器学习框架中实现动态计算图也是不太容易。这两种实现都不存在 TensorFlow 的官方文档中，只有对源码有足够深入的了解才可能在功能和性能上有巨大的突破，因此如果你是企业内 TensorFlow 框架的基础架构维护者，突破 TensorFlow 的 Python API 抽象层是非常有必要的。

大家在应用机器学习时，不知不觉已经使用了很多基础架构的抽象，其中最重要的莫过于机器学习算法本身的实现，接下来我们将突破抽象，深入了解底层的实现原理。

第二部分，机器学习的数值计算

机器学习，本质上是一系列的数值计算，因此 TensorFlow 定位也不是一个深度学习库，而是一个数值计算库。当我们听到了香农熵、贝叶斯、反向传播这些概念时，并不需要担心，这些都是数学，而且可以通过计算机编程实现的。

接触过机器学习的都知道 LR，一般是指逻辑回归 (Logistic regression)，也可以指线性回归 (Linear regression)，而前者属于分类算法，

后者属于回归算法。两种 LR 都有一些可以调优的超参数，例如训练轮数（Epoch number）、学习率（Learning rate）、优化器（Optimizer）等，通过实现这个算法可以帮忙我们理解其原理和调优技巧。

下面是一个最简单的线性回归 Python 实现，模型是简单的 $y = w * x + b$ 。

```
def linear_regression():  
  
    x_array = [1.0, 2.0, 3.0, 4.0, 5.0]  
    y_array = [5.0, 7.0, 9.0, 11.0, 13.0]  
    instance_number = len(x_array)  
    learning_rate = 0.01  
    epoch_number = 100  
    w = 1.0  
    b = 1.0  
  
    for epoch_index in range(epoch_number):  
        w_grad = 0.0  
        b_grad = 0.0  
        loss = 0.0  
  
        for i in range(instance_number):  
            x = x_array[i]  
            y = y_array[i]  
            w_grad += -2.0 * x * (y - w * x - b)  
            b_grad += -2.0 * (y - w * x - b)  
            loss += 1.0 * pow(y - w * x - b, 2)  
  
        w -= learning_rate * w_grad  
        b -= learning_rate * b_grad  
        print("Epoch is: {} w is: {}, w is: {}, loss is: {}".format(  
            epoch_index, w, b, loss))
```

从这个例子大家可以看到，实现一个机器学习算法并不依赖于 Scikit-learn 或者 TensorFlow 等类库，本质上都是数值运算，不同语言实现会有性能差异而已。细心的朋友可能发现，为什么这里 w 的梯度（Gradient）是 $-2 * x * (y - x * x - b)$ ，而 b 的梯度这是 $-2 * (y - w * x - b)$ ，如何保证经过计算后 Loss 下降而准确率上升？这就是数学上保证了，我们定义了 Loss 函数（Mean square error）为 $y - w * x - b$ 的平方，也就是说预测值越接近 y 的话 Loss 越小，目标变成求 Loss 函数在 w 和 b 的任意取值下的最小值，因此对 w 和 b 求偏导后就得到上面两条公式。

逻辑回归与线性回归类似，当由于是分类问题，因此需要对 $w * x + b$ 的预测结果进行归一化（Normalization），一般使用 Sigmoid 方法，在

Python 中可以通过 $1.0 / (1 + \text{numpy.exp}(-x))$ 这种方式实现。由于预测值不同，Loss 函数的定义也不同，求偏导得到的数值计算公式也不同。最终求得的偏导是非常简单的，用任何编程语言都可以轻易实现。但我们自己的实现未必是最高效的，为什么不直接用 Scikit-learn、MXNet 这些开源库已经实现好的算法呢？

我们对这个算法的理解，其实是在工程上使用它的一个很重要的基础。例如在真实的业务场景下，一个样本的特征可能有百亿甚至千亿维，而通过前面的算法我们了解到，LR 模型的大小和样本特征的维度是相同的，也就是说一个接受百亿维特征的模型，本身参数就有百亿个，如果使用标准的双精度浮点数保存模型参数，那么百亿维的模型参数部分至少要超过 40G，那么千亿维的特征更是单机所无法加载的。

因此，虽然 Scikit-learn 通过 native 接口实现了高性能的 LR 算法，但只能满足在单机上训练，而 MXNet 由于原生没有支持 SparseTensor，对于超高维度的稀疏数据训练效率是非常低的，TensorFlow 本身支持 SparseTensor 也支持模型并行，可以支持百亿维特征的模型训练，但没有针对 LR 优化效率也不是很高。在这种场景下，第四范式基于 Parameter server 实现了支持模型并行和数据并行的超高维度、高性能机器学习库，在此基础上大规模 LR、GBDT 等算法训练效率才能满足工程上的需求。

机器学习还有很多有意思的算法，例如决策树、SVM、神经网络、朴素贝叶斯等等，只需要部分数学理论基础就可以轻易在工程上实现，由于篇幅关系这里就不在赘述了。前面我们介绍的其实是机器学习中的命令式（Imperative）编程接口，我们把求偏导的公式提前推导出来，然后像其他编程脚本一样根据代码那顺序执行，而我们知道 TensorFlow 提供的是一种声明式（Declarative）的编程接口，通过描述计算图的方式来延后和优化执行过程，接下来我们就介绍这方面的内容。

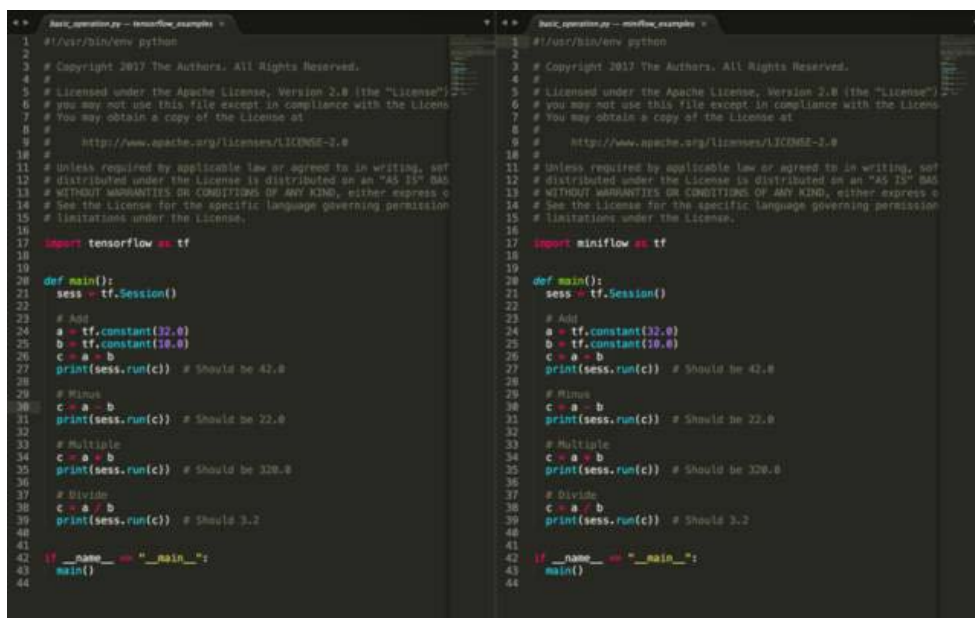
第三部分，TensorFlow 的重新实现

首先大家可能有疑问，我们需要需要重新实现 TensorFlow ？

TensorFlow 灵活的编程接口、基于 Eigen 和 CUDA 的高性能计算、支持分布式和 Hadoop HDFS 集成，这些都是个人甚至企业很难完全追赶实现的，而且即使需要命令式编程接口我们也可以使用 MXNet，并没有强需求需要一个新的 TensorFlow 框架。

事实上，我个人在学习 TensorFlow 过程中，通过实现一个 TensorFlow-like 的项目，不仅惊叹与其源码和接口的设计精巧，也加深了对声明式编程、DAG 实现、自动求偏导、反向传播等概念的理解。甚至在 Benchmark 测试中发现，纯 Python 实现的项目在线性回归模型训练中比 TensorFlow 快 22 倍，当然这是在特定场景下压测得到的结果，主要原因是 TensorFlow 中存在 Python 与 C++ 跨语言的切换开销。

这个项目就是 MiniFlow，一个实现了链式法则、自动求导、支持命令式编程和声明式编程的数值计算库，并且兼容 TensorFlow Python API。感兴趣可以在这个地址参与开发 <https://github.com/tobegit3hub/miniflow>，下面是两者 API 对比图。



```
1 #!/usr/bin/env python
2
3 # Copyright 2017 The Authors. All Rights Reserved.
4
5 # Licensed under the Apache License, Version 2.0 (the "License");
6 # you may not use this file except in compliance with the License.
7 # You may obtain a copy of the License at
8
9 # http://www.apache.org/licenses/LICENSE-2.0
10
11 # Unless required by applicable law or agreed to in writing, software
12 # distributed under the License is distributed on an "AS IS" BASIS,
13 # WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
14 # implied. See the License for the specific language governing
15 # permissions and limitations under the License.
16
17 import tensorflow as tf
18
19 def main():
20     sess = tf.Session()
21
22     # Add
23     a = tf.constant(32.0)
24     b = tf.constant(10.0)
25     c = a + b
26     print(sess.run(c)) # Should be 42.0
27
28     # Minus
29     c = a - b
30     print(sess.run(c)) # Should be 22.0
31
32     # Multiple
33     c = a * b
34     print(sess.run(c)) # Should be 320.0
35
36     # Divide
37     c = a / b
38     print(sess.run(c)) # Should 3.2
39
40 if __name__ == "__main__":
41     main()
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
```

```
1 #!/usr/bin/env python
2
3 # Copyright 2017 The Authors. All Rights Reserved.
4
5 # Licensed under the Apache License, Version 2.0 (the "License");
6 # you may not use this file except in compliance with the License.
7 # You may obtain a copy of the License at
8
9 # http://www.apache.org/licenses/LICENSE-2.0
10
11 # Unless required by applicable law or agreed to in writing, software
12 # distributed under the License is distributed on an "AS IS" BASIS,
13 # WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
14 # implied. See the License for the specific language governing
15 # permissions and limitations under the License.
16
17 import miniflow as tf
18
19 def main():
20     sess = tf.Session()
21
22     # Add
23     a = tf.constant(32.0)
24     b = tf.constant(10.0)
25     c = a + b
26     print(sess.run(c)) # Should be 42.0
27
28     # Minus
29     c = a - b
30     print(sess.run(c)) # Should be 22.0
31
32     # Multiple
33     c = a * b
34     print(sess.run(c)) # Should be 320.0
35
36     # Divide
37     c = a / b
38     print(sess.run(c)) # Should 3.2
39
40 if __name__ == "__main__":
41     main()
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
```

了解 TensorFlow 和 MXNet（或者 NNVM）源码的朋友可能知道，两者都抽象了 Op、Graph、Placeholder、Variable 等概念，通过 DAG 的方式

描述模型的计算流图，因此我们也需要实现类似的功能接口。

与前面的 LR 代码不同，基于 Graph 的模型允许用户自定义 Loss 函数，也就是用户可以使用传统的 Mean square error，也可以自定义一个任意的数学公式作为 Loss 函数，这要求框架本身能够实现自动求导的功能，而不是我们根据 Loss 函数预先实现了导数的计算方式。

那么用户可以定义的最小操作，也就是 Op，需要平台实现基本的算子，例如 ConstantOp、AddOp、MultipleOp 等，而且用户实现自定义算子时可以加入自动求导的流程中，并不影响框架本身的训练流程。参考 TensorFlow 的 Python 源码，下面我们定义了 Op 的基类，所有的 Op 都应该实现 forward() 和 grad() 以便于模型训练时自动求导，而且通过重载 Python 操作符可以为开发者提供更便利的使用接口。

```
class Op(object):
    """The basic class for all operation."""

    def __init__(self, name="Op"):
        self.name = name

    def forward(self):
        raise NotImplementedError

    def grad(self):
        raise NotImplementedError

    def __add__(self, other):
        return AddOp(self, other)

    def __radd__(self, other):
        return self.__add__(other)

    def __sub__(self, other):
        return MinusOp(self, other)

    def __rsub__(self, other):
        return MinusOp(other, self)

    def __mul__(self, other):
        return MultipleOp(self, other)

    def __rmul__(self, other):
        return self.__mul__(other)

    def __div__(self, other):
        return DivideOp(self, other)

    def __rdiv__(self, other):
        return DivideOp(other, self)

    def __pow__(self, power, modulo=None):
        return PowerOp(self, power)

class ConstantOp(Op):
    def __init__(self, value, name="Constant"):
        super(ConstantOp, self).__init__(name)
        self._value = value

    def forward(self):
        return self._value

    def grad(self, partial_derivative_opname=None):
        return 0

class VariableOp(Op):
    def __init__(self, value, is_trainable=True, name="Variable"):
        super(VariableOp, self).__init__(name)
        self._value = value

    def forward(self):
        return self._value

    def grad(self, partial_derivative_opname=None):
        if partial_derivative_opname is None:
            grad = 1
        else:
            if self.name == partial_derivative_opname:
                grad = 1
            else:
                grad = 0
        return grad
```

那么对于常量 (ConstantOp) 和变量 (VariableOp)，他们的正向运算就是得到的是本身的值，而求导时常量的导数为 0，求偏导的变量导数为 1，其他变量也为 0，具体代码如下。

其实更重要的是，我们需要实现加 (AddOp)、减 (MinusOp)、乘 (MultipleOp)、除 (DivideOp)、平方 (PowerOp) 等算子的正向运算和反向运算逻辑，然后根据链式法则，任何复杂的数学公式求导都可以简化成这些基本算子的求导。

例如加法和减法，我们知道两个数加法的导数等于导数的加法，因此根据此数学原理，我们可以很容易实现 AddOp，而 MinusOp 实现类似就不赘述了。

```
class AddOp(Op):
    def __init__(self, input1, input2, name="Add"):
        super(AddOp, self).__init__(name)

        if not isinstance(input1, Op):
            self._op1 = ConstantOp(input1)
        else:
            self._op1 = input1

        if not isinstance(input2, Op):
            self._op2 = ConstantOp(input2)
        else:
            self._op2 = input2

    def forward(self):
        result = self._op1.forward() + self._op2.forward()
        return result

    def grad(self, partial_derivative_opname=None):
        result = self._op1.grad(partial_derivative_opname) + self._op2.grad(
            partial_derivative_opname)
        return result
```

而乘法和除法相对复杂，显然两个数乘法的导数不等于导数的乘法，例如 x 和 x 的平方，先导数后相乘得到 $2x$ ，先相乘后导数得到 3 倍 x 的平方。因此这是需要使用乘数法则，基本公式是，而代码实现如下。

除法和平方的求导方式也是类似的，因为数学上已经证明，所以只需要编码实现基本的正向和反向运算即可。由于篇幅有限，这里不再细

```
class MultipleOp(Op):  
  
    def __init__(self, input1, input2, name="Multiple"):  
        super(MultipleOp, self).__init__(name)  
        self._op1 = input1  
        self._op2 = input2  
  
    def forward(self):  
        result = self._op1.forward() * self._op2.forward()  
        return result  
  
    def grad(self, partial_derivative_opname=None):  
        op1_value = self._op1.forward()  
        op2_value = self._op2.forward()  
        op1_grad = self._op1.grad(partial_derivative_opname)  
        op2_grad = self._op2.grad(partial_derivative_opname)  
        result = op1_grad * op2_value + op1_value * op2_grad  
        return result
```

致介绍 MiniFlow 的源码实现了，感兴趣可以通过上面的 Github 链接找到完整的源码实现，下面再提供使用相同 API 接口实现的模型性能测试结果，对于小批量数据处理、需要频繁切换 Python/C++ 运行环境的场景下 MiniFlow 会有更好的性能表现。

前面介绍了机器学习算法和深度学习类库的实现，并不是所有人都有能力去重写或者优化这部分基础架构的，很多时候我们都只是这些算法的使用者，但从另一个角度，我们就需要维护一个高可用的计算平台来做机器学习的训练和预测，下面将从这方面介绍如何打造分布式机器学习平台。

第四部分，分布式机器学习平台的设计

随着大数据和云计算的发展，实现一个高可用、分布式的机器学习平台成为一个基本需求。无论是 Caffe、TensorFlow，还是我们自研的高性能机器学习库，都只是解决数值计算、算法实现以及模型训练的问题，对于任务的隔离、调度、Failover 都需要上层平台实现。

那么设计一个针对机器学习全流程的基础架构平台，需要涵盖哪些功能呢？

首先，必须实现资源隔离。在一个共享底层计算资源的集群中，用户提交的训练任务不应该受到其他任务的影响，尽可能保证 CPU、内存、GPU 等资源隔离。如果使用 Hadoop 或 Spark 集群，默认就会在任务进程上挂载 cgroups，保证 CPU 和内存的隔离，而随着 Docker 等容器技术的成熟，我们也可以使用 Kubernetes、Mesos 等项目来启动和管理用户实现的模型训练任务。

其次，实现资源调度和共享。随着通用计算的 GPU 流行，目前支持 GPU 调度的编排工具也越来越多，而部分企业内还存在着 GPU 专卡专用的情况，无法实现资源的动态调度和共享，这必然导致计算资源的严重浪费。在设计机器学习平台时，需要尽可能考虑通用的集群共享场景，例如同时支持模型训练、模型存储以及模型服务等功能，可以对标的典例就是 Google Borg 系统。

然后，平台需要有灵活的兼容性。目前机器学习业务发展迅速，针对不同场景的机器学习框架也越来越多，灵活的平台架构可以兼容几乎所有主流的应用框架，避免基础架构因为业务的发展而频繁变化。目前 Docker 是一种非常合适的容器格式规范，通过编写 Dockerfile 就可以描述框架的运行环境和系统依赖，在此基础上我们可以在平台上实现了 TensorFlow、MXNet、Theano、CNTK、Torch、Caffe、Keras、Scikit-learn、XGBoost、PaddlePaddle、Gym、Neon、Chainer、PyTorch、Deeplearning4j、Lasagne、Dssne、H2O、GraphLab 以及 MiniFlow 等框架的集成。

最后，需要实现机器学习场景下的 API 服务。针对机器学习的模型开发、模型训练和模型服务三个主要流程，我们可以定义提交训练任务、创建开发环境、启动模型服务、提交离线预测任务等 API，用熟悉的编程语言来实现 Web service 接口。要实现一个 Google-like 的云深度学习平台，大家可以参考下面这三个步骤。

当然，要实现一个涵盖数据引入、数据处理、特征工程以及模型评估功能的机器学习平台，我们还需要集成 HDFS、Spark、Hive 等大数据处

■ Architecture of Google-like Cloud Machine Learning



理工具，实现类似 Azkaban、Oozie 的工作流管理工具，在易用性、低门槛方面做更多的工作。

总结

最后总结一下，机器学习的基础架构包含了机器学习算法、机器学习类库以及机器学习平台等多个层次的内容。根据业务的需求，我们可以选择特定的领域进行深入研究和二次开发，利用轮子和根据需求改造轮子同样重要。

在机器学习与人工智能非常流行的今天，希望大家也可以重视底层基础架构，算法研究员可以 理解更多工程的设计与实现，而研发工程师可以了解更多的算法原理与优化，在合适的基础架构平台上让机器学习发挥更大的效益，真正应用的实际场景中。

（七）如何解决特征工程，克服工业界应用 AI 的巨大难关

作者 陈雨强



人工智能是一个非常炙手可热的名词，且已经成功应用在语音、图像等诸多领域。但是，现在人工智能有没有达到可以简单落地的状态呢？工业界的人工智能需要什么技术呢？带着这些问题开始我们的思考。

工业大数据需要高 VC 维

首先，我们先探讨一下工业界人工智能需要一个什么样的系统？人工智能的兴起是由于数据量变大、性能提升以及并行计算技术发展共同产生的结果。所以，工业界的问题都是非常复杂的。因此，我们需要一个可扩展系统，不仅在吞吐与计算能力上可扩展，还需要随着数据量与用户的增

多在智能水平上可扩展。怎么实现一个可扩展系统呢？其实很重要的一点是工业界需要高 VC 维的模型，去解决智能可扩展性的问题。怎么获得一个高 VC 维的模型呢？大家都知道，机器学习 = 数据 + 特征 + 模型。如果数据在给定的情况下，我们就需要在特征和模型两个方面进行优化。

特征共分两种，一种叫宏观特征，比方说年龄、收入，或是买过多少本书，看过多少部电影。另外一种微观特征，指的是比拟细粒度的特征，你具体看过哪几本书，或者具体看过哪几部电影。每一部电影，每一本书，每一个人，都是不同的特征。书有几百万本，电影有几百万部，所以这样的特征量非常大。

模型可分为两类，一个是简单模型，比如说线性模型。还有一种是复杂模型，比如非线性模型。



这样就把人工智能分为了四个象限。如上图，左下角是第一象限，使用宏观特征简单模型解决问题。这种模型在工业界应用非常少，因为它特征数少，模型又简单，VC 维就是低的，不能解决非常复杂的问题。右下

角的第二象限是简单模型加上微观特征，最有名的就是大家熟知的谷歌 Adwords，用线性模型加上千亿特征做出了世界顶尖的广告点击率预估系统。左上角的第三象限是复杂模型加宏观特征，也有诸多知名公司做出了非常好的效果，例如 Bing 广告和 Yahoo，经典的 COEC+ 复杂模型在这个象限内是一个惯用手段。最后是第四象限，利用复杂模型加上微观特征，由于模型空间太大，如何计算以及解决过拟合都是研究的热点。

刚才说沿着模型和特征两条路走，那如何沿着模型做更高维度的机器学习呢？

研究模型主要是在学术界，大部分的工作是来自于 ICML、NIPS、ICLR 这样的会议，非线性有三把宝剑分别是 Kernel、Boosting、Neural Network。Kernel 在十年前非常火，给当时风靡世界的算法 SVM 提供了非线性能力。Boosting 中应用最广泛的当属 GBDT，很多问题都能被很好地解决。Neural Network 在很多领域也有非常成功的应用。工业界优化模型的方法总结起来有以下几点。首先，基于过去的数据进行思考得到一个假设，然后将假设的数学建模抽象成参数加入，用数据去拟合新加入的参数，最后用另一部分数据验证模型的准确性。

这里举一个开普勒沿模型这条路发现开普勒三定律的例子。在中世纪的时候，第谷把自己的头绑在望远镜上坚持观察了 30 年夜空，将各个行星的运动轨迹都记录下来。基于这些数据，开普勒不断的进行假设，最后假设行星的运动轨道是椭圆的，用椭圆的方程去拟合他的数据，发现拟合的非常好，便得到了一个新的模型：开普勒第一定律。这就是一个典型的沿着模型走的思路，通过观测数据，科学家获得一个假设，这个假设就是一个模型，然后用数据拟合这个模型的参数，最终在新的数据上验证模型是否正确，这是沿着模型走的一条路。

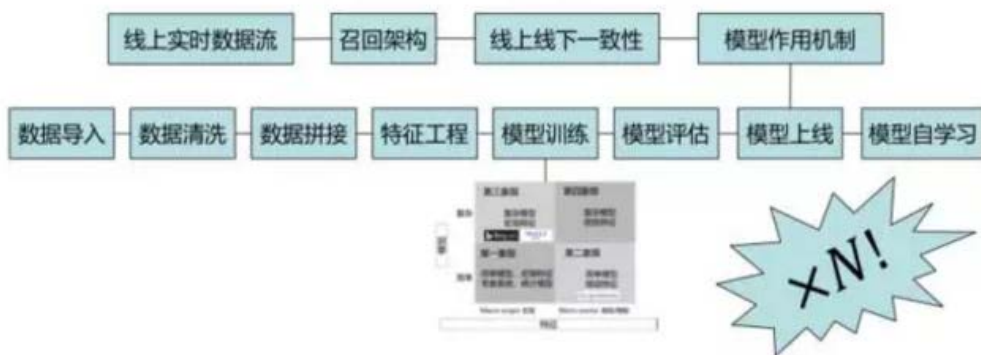
沿着特征走主要是由工业界来主导，比如说谷歌的 Adwords 里面有上千亿的特征，想要描述为什么会产生一次广告点击，这就需要解决高效并行的问题。这些技术大部分进展来自于 KDD 或是 WWW。沿着特征优化机器学习，就是把问题通过特征的方式分的足够细，做非常准确的模型。

到底是深度模型好还是宽度模型好呢？这里有一个没有免费的午餐定理：不存在万能的模型。简单来说，世界上不存在一个优化算法对任何问题上都有效，也就是说我们总能找到一个问题，让这个优化算法表现的并不比随机的更好。更进一步的说，所有的机器学习都是一个偏执，代表了对这个世界的认知。如果数据较少，这个偏执就需要比较强。比如说科学家观测物理现象，数据并不是特别多。这种情况下，你需要大量的理论和猜想，有少量数据做拟合验证就可以了。但如果假设错的话，就可能出现错误的结论。比如用地心论研究天体物理的话，就发现结论都是错的。但是如果数据很多，我们就不需要很强的偏置，将更多的不确定性加入模型，自动的通过数据进行拟合。综合起来，工业界的机器学习里面并没有免费的午餐，不存在哪一个模型是万能的模型。所以说你一定要根据你的业务做出合适的选择，才是最好的一个方式。

人工智能落地的关键：提高 AI 的易用性

人工智能目前还远没有达到可以遍地开花的程度，即使解决了刚才讲的宽与深的问题，我们依然还有很多事情要做。如何训练出好的模型、如何去选择好的参数、如何进行特征组合，都不是一件容易的事情。

工业界应用机器学习的难题



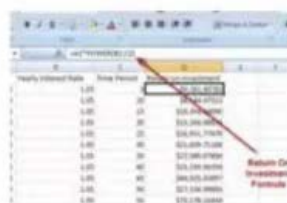
比如，数据需要归集、整理、导入、清洗、拼接、特征工程等之后才能模型训练。之后模型上线还要保证系统的稳定性、时效性和吞吐等等指

标。同时为了提供线上服务，需要重做一套线上的架构，这个架构需要保证数据流实时性、线上线下一致性，以及模型作用的机制。只有完成这些，这样才真正拥有了一个落地的人工智能系统，否则只能称之为在笔记本上做一个 AI 的玩具而已。

刚才说到的还只是一个公司的问题与系统，如果用在各行各业的不同问题上，会发现问题非常多，所以企业一定需要一个集成上述所有能力的人工智能平台。开源社区内好的工具平台和算法能够起到很大作用，这些工具也会日趋成熟，但并不足够。虽然看起来人工智能已经在非常多的领域应用或者开始了突破性的尝试，但是如果对比 Hadoop 这样的分布式存储计算系统来说，还远远未普及。

工业界应用机器学习的难题

- 需要AI应用平台
 - Tensorflow, Mxnet, Caffe等工具日趋丰富
 - 但是，足够了么？
- 为什么人工智能还没有真的大规模应用到每个企业
 - Hadoop为什么用的人多？
 - 先驱知识要求太多
 - 能做AI的还是研究/应用机器学习科学家
 - 核心机器学习算法平台只降低了一部分门槛
 - 更大的应用基础：降门槛 > 算法效果



关于这个问题我们先分析 Hadoop。之所以这么多人用 Hadoop 是因为它虽然是个分布式系统，但对使用它的程序员并不需要掌握很高的分布式系统知识，研发人员并不需要为了使用 Hadoop 针对性的对自己的数据、业务做出改变，也不需要因为 Map-Reduce 框架重新设计自己的线上服务系统。但人工智能不一样，为了使用 AI，所有的上下游组件都会和模型相关：不同的模型不仅意味着不同的训练系统，还意味着不同的实时、非实时的数据流，不同的拼表要求与框架选择、不同的特征抽取、不同的线上服务架构、不同的灾备架构、回滚架构相关。这样你就会发现，为 AI 系统做数据流与线上系统的架构师，必须要懂机器学习才能做好工作。

所以现在能够做 AI 应用的人，主要还是那些研究及应用的机器学习科学家，需要那种既懂机器学习，又了解业务，还精通系统架构的工程师。这就造成了 AI 的高门槛。就如同三四十年前，真正编程的人并不是现在我们这样的人，而是一群科学家们，他们通过纸带来控制程序的运行，自己不仅要编程，还得非常懂计算机体系架构。导致的结果是，根本不是每个人都能接触到这项技术，不是每个企业都能受惠于这个技术。但是现在，甚至在 Excel 中都可以编程，这些程序员可能完全不知道计算机的体系结构、操作系统、编译原理、数据库的概念，将全部心思花在理解与解决业务问题上，达到事半功倍的效果。

所以，如果想让 AI 在工业界中产生更大的影响，真正的落地，我们需要的是一个完整的人工智能应用平台，让人以更低的成本用上人工智能。从这个角度上看，阻碍 AI 普及的并不是现在的算法效果不够好，而是现在算法的门槛太高，研发新的平台以及算法降低门槛的重要性大于优化算法效果的重要性，我们期望用低的门槛获得好的效果。

如何解决特征工程

如何降低这些门槛呢？这里分享一下第四范式的成果。首先特征工程是工业界应用 AI 的巨大的难关。特征工程的目标是针对于某个模型找出与要解决问题相关的关键属性，现在也有一些开源的项目尝试解决特征工程，下图就列出了 Spark 2.2 官方文档中包含的特征工程算法。那么，针对不同的业务、不同的模型，这些算子就足够我们低门槛建模了吗？

如果想要做好特征工程，需要对将要使用的机器学习算法有深入了解才行，随便地将所有的特征全部扔进去，现有的算法并不能很好地处理。有时候，不同的算法为了达到同一个目标，使用特征工程与做法会完全不一样。以新闻推荐为例，我们要做两种特征，来提高推荐新闻的点击率。一种是一阶特征，描述的是那些用户直接喜欢的内容。另一种是二阶特征，描述的的是个性兴趣的扩展。比如说喜欢大数据的人，很有可能对机器学习也感兴趣。

如何解决特征工程

- 特征工程在工业界是巨大的难关
 - 什么是特征工程？现在的平台已经足够了吗？
 - 需要对机器学习与业务都非常理解
 - 不同的算法，要使用不同的特征工程达到同一个目标
- 以新闻推荐为例
 - 一阶特征：每个用户直接喜欢什么
 - 二阶特征：用户的扩展兴趣（喜欢大数据的人，可能对机器学习也感兴趣）
 - 不同模型如何添加？

- Feature Transformers
 - Tokenizer
 - StopWordsRemover
 - N-gram
 - Browser
 - PCA
 - PolynomialExpansion
 - Discrete Cosine Transform (DCT)
 - StringIndexer
 - IndexToString
 - OneHotEncoder
 - VectorIndexer
 - Interaction
 - Normalizer
 - StandardScaler
 - MinMaxScaler
 - MaxAbsScaler
 - Bucketizer
 - ElementwiseProduct
 - SQLTransformer
 - VectorAssembler
 - QuantileDiscretizer
 - Imputer

在下面的示意中，小人代表一个用户（User），小人下面表示通过统计得到的用户画像，也就是用户的历史兴趣点（User_Topic）。右边是 3 篇新闻，每个新闻有一个话题（News_Topic）。

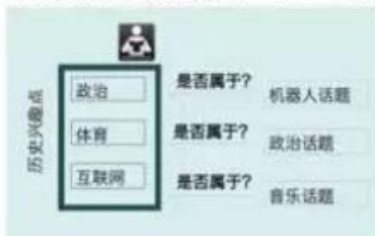
线性模型，学习一阶特征



线性模型，学习二阶特征



非线性宏观特征模型，学习一阶特征



非线性宏观特征模型，学习二阶特征



那么，如何在之前提到的“简单模型（线性模型）+ 微观特征”这条路添加一阶特征呢？如图左上角所示，我们只需要简单的将用户与新闻话题做组合特征（User-New_Topic 笛卡尔积）就可以了。在这条路上，我们并不需要任何的用户画像统计，因为最后每篇新闻点击或者不点击就已经可以训练出“User-News_Topc”组合特征的权重与偏好了。这种方式下，在线上服务的时候，所有的信息都是推荐时能获取的，但是为了用户兴趣更

新及时，我们需要把模型的时效性做到非常高。

回头看，如何在之前提到的“复杂模型（非线性模型）+ 宏观特征”这条路添加一阶特征呢？如图左下角所示，由于是宏观特征，我们需要将不同的话题变成单个特征，一种做法是通过一阶逻辑的判断“本篇新闻的话题是否属于用户历史的兴趣”加入。这种方式下，在线上服务的时候，我们除了需要推荐实时信息之外，还需要实时维护用户历史的兴趣点，但是模型本生的更新频率就不用那么快了。毕竟，为了达到推荐时效性目标，要么特征静态、模型特别实时，要么特征实时、模型静态不变。

那么，如果我们要学习二阶特征呢？对于线性模型（如右上角所示），我们也需要用到用户的历史兴趣点，将用户的历史喜好与文章的话题进行组合（User_Topic-New_Topic），这样模型就可以学到历史上喜欢什么样话题的人还会喜欢什么样的新闻话题，达到二阶迁移的目标。对于非线性模型（如右下角所示），我们要做的将原先的一阶逻辑判断（可以认为是个 Identity 矩阵）变成一个二阶状态转移矩阵，通过历史统计得知不同话题间喜欢转换的情况，推算出一个不在用户现有兴趣点中的文章话题是不是用户喜欢的。

如何解决特征工程

- 特征工程在工业界是巨大的难关
 - 需要对机器学习与业务都非常理解
 - 不同的算法，要使用不同的特征工程达到同一个目标

	一阶特征（用户直接喜欢什么）	二阶特征（用户的扩展兴趣）
线性模型	用户ID -- (不用统计，形成高维特征)	用户历史兴趣点 -- (需要统计，形成高维特征)
非线性模型（宏观特征）	用户历史兴趣点 是否包含待推荐新闻话题 (需要统计，形成单个特征)	用户历史兴趣点，兴趣相关性矩阵，待推荐新闻话题 (需要统计，单特征)
非线性模型（精细特征）	用户ID -- (不用统计，形成高维特征)	用户历史兴趣点 -- (需要统计，形成高维特征)

更进一步的，我们总结对比一下，对于前文提到的机器学习四象限中的第 2, 3, 4 象限的模型，我们做特征工程的方式差距非常大。对于一阶

特征，如果是线性模型加精细特征，直接做组合，不用统计；如果做非线性模型是需要统计的，同时用包含关系来做；如果用非线性模型不需要用包含关系，模型本身会进行特征组合。如果做二阶特征，每种方法都需要使用统计特征，但适用方式也各不相同，比方说非线性模型宏观特征，你需要三个相关的信息和很多统计才可以做到。

这个例子说明了一个道理，如果要做好的特征工程，需要非常多的针对模型的定制化的优化，仅用现在的工具还完全不够，完全需要靠人的经验与判断。因此，研发自动特征工程的算法就变得尤为重要。自动特征工程是一个比较难的问题，在学术界与工业界都在积极地研究这个问题，这里跟大家分享自动工程的三个方向，隐式特征组合（如 NN，FM），半显式特征组合（如 GBDT）与显式特征组合（显式特征叉乘）。

隐式特征组合

隐式特征组合主要特点是对连续值特征非常友好，最成功的应用场景是语音和图像。在这些原始信号是像素或是声波的问题里面，深度学习通过神经网络产生底层的 Filter 以及层次化的特征组合，获得了远超人类手工特征工程的效果。但是深度神经网络并不是万能的，在深度学习中，高维离散特征的变量处理非常复杂，同时缺乏可解释性，过于黑盒化也是神经网络大家关注的焦点。这样会导致深度学习出来的特征组合相对难用

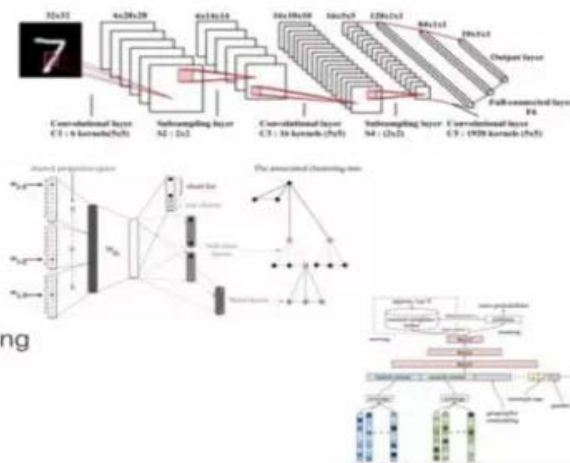
隐式特征组合

主要特点

- 对连续值特征天然友好
- 最大的成功：语音图像
- 高级离散变量处理相对更复杂
- 隐式组合，基本无可解释性

对离散特征需要Large Scale Embedding

- Embedding NN
- FM, FNN, PNN
- DeepFM

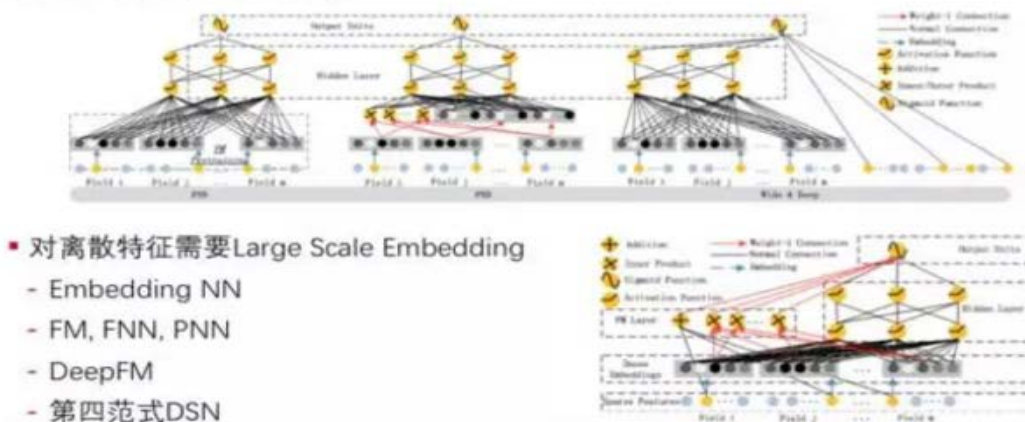


到其他算法之上，也很难给人明确的信息反馈。

针对 NN 难以处理离散特征的问题，我们需要 Large Scale Embedding 的技术进行解决。Embedding 最早在 NN 上的应用是在 NLP 的问题上，当时研究者们使用 Embedding 的技术将每个单词映射到一个低维空间，通过 concat, sum, pooling, convolution 等方式形成等长的最底层输入，然后使用标准的深度神经网络进行训练。随后在越来越多的领域用上了 Embedding 技术，推荐是一个典型的场景，限制波尔兹曼机（RBM）在提出之初就曾使用 Embedding 尝试解决协同过滤问题。

最近，谷歌发表描述如何使用大规模 Embedding 技术在 Youtube 上向数以十亿计的用户推荐数以亿计的视频，在他们的工作中，他们同时对每个用户与视频进行 Embedding，之后把用户观看历史、搜索历史等视频的向量通过求和变成特征，继而进行深度学习，获得了比较大的成功。

隐式特征组合



Large Scale Embedding 在现在依然是一个比较热门的研究领域，其中的成果包括 Discrete Factorization Machine, FNN, PNN, DeepFM 等，上图显示了这些算法的异同，简单来讲这些模型既能找到特征之间的推理关系，同时也能去记忆比较细的特征。在这个领域，第四范式提出了 DSN（Deep Sparse Network）的算法，它是一个非常宽且深的模型，里面同样会做大规模 Embedding，用神经网络做自动学习组合，目标解决高维模型（上万亿 VC 维度）的正则化以及并行计算问题。

半显式特征组合

第二个是半显式的组合,主要基于的是树模型。为什么说是半显式呢?因为大家可能认为树模可解释或者做特征组合是很自然的事情,但其实并不是:叶子节点的每一个分支并不是一种显式、直接的特征组合,而是这些特征在特定取值区间的组合。所以从结果上来说我们做到了特征组合,有一定可解释性,但是同样也没有办法直接看特征相关性或者特征之间组合关系。作为非线性模型,树模型主要的特点是容易理解,效果也是非常好的。但是类似的,它对离散的精细特征非常难处理,传统上训练一棵 m 个特征 n 个训练数据 k 层深 t 棵树的模型需要 $O(mntk)$ 的时间,即使对系数特征进行优化,也很难降低特征分裂中分桶上的空间与传输消耗。在这个方面,第四范式提出了一系列算法,包括 HE-TreeNet 和 GBM 系列算法,通过 Embedding, Ensemble、Stacking, General Boosting 等方式让树模型可以在大规模特征的情况下可以进行特征组合。

半显式特征组合

- 主要是森林类算法
 - 为什么是“半显式”
 - 看起来可以解释,实际上并不可解释
 - 看起来在做特征组合,实际上是层次贪心的副产物
- 主要特点
 - 理解容易,相对鲁棒,效果优秀
 - Off-the-shelf
 - 离散特征非常难解,无现有方案
- 第四范式HE-TreeNet, GBM
 - 解决大规模离散特征的树模型
 - 研发基于Embedding, Ensembling, Stacking的系列树算法

显式特征组合

显式特征组合有一个问题是如何做连续值的组合,比如说一个人的年龄是 30,收入是 10000,应该怎么做组合特征呢?是乘、是加还是平方和?在 NN 中,是通过线性组合加上非线性变化,在 GBDT 中使用过特征值

分裂，但是对限时特征组合，实际上没有一个好的现有方法去直接组合。

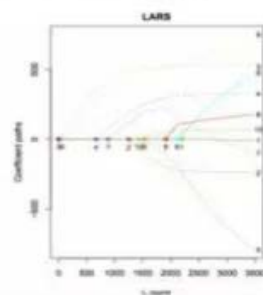
显式特征组合：问题

- 主要基于贪心与搜索
 - 正则化
 - Beam Search, MCTS
 - 遗传算法, 模拟退火
- 问题特别的难
 - 围棋的状态空间 $< 3^{19 \times 19}$ ；而 n 个特征，选 m 个特征，限制最大 k 阶组合，状态空间为 $C_{\sum_{i=2}^m C_n^i}^m$
 - 难以组合连续值特征？
- 显式特征组合优势
 - 可解释性：提供深度业务洞察
 - 可叠加性：增强所有机器学习算法

虽然有着重重困难，显式特征组合的优势在于可解释性，会提供一个非常深入的洞察，可以知道哪些特征是潜在有关系的，应该组合起来；同时这样的方法享受比较好的可叠加性：所有机器学习算法都基于特征，显式特征组合的产出是特征集，这个特征集可以增强所有其他机器学习的算法，成为训练的基础。

显式特征组合：现状

- State-of-art
 - Online Boosting Feature Selection: 单特征Weak Learner基于Adaboost的选择
 - Online Regularization：基于Lasso对梯度、权重截断
- 现有算法的问题
 - 并非为 n 选 m 个 k 阶以下特征设计
 - 多为副产物，对信息损失的比较大
 - 二阶组合为主，基本无法高阶特征组合



目前，显式特征组合主要有几种算法，一些方法基于 Boosting，训练单 Feature 弱分类器，通过 Boosting 的过程寻找组合的启发；或者基于 Regularization 进行权重截断，形成组合候选。这些算法一般不是为了特

征组合而设计，组合也多为训练过程的副产物，很难真正的获得高阶的组合特征。

新一代的显式特征组合：FeatureGO

下面介绍第四范式最新发布的算法—FeatureGO。它是基于 MCTS，对特征与特征组合状态进行建模，训练组合下的收益函数。在搜索的过程中，我们做了非常多的调优技术，使用我们内部线性分型算法 LFC 解决连续值特征组合的问题。最终，我们发现这个特征组合最多能达到十阶以上，且发现即使达到十阶也能提供明显的的效果提升，这个是过去靠人所做不到的事情。即使是在最好的广告或者推荐系统中，人工的特征组合一般也只能到达 5-6 阶。

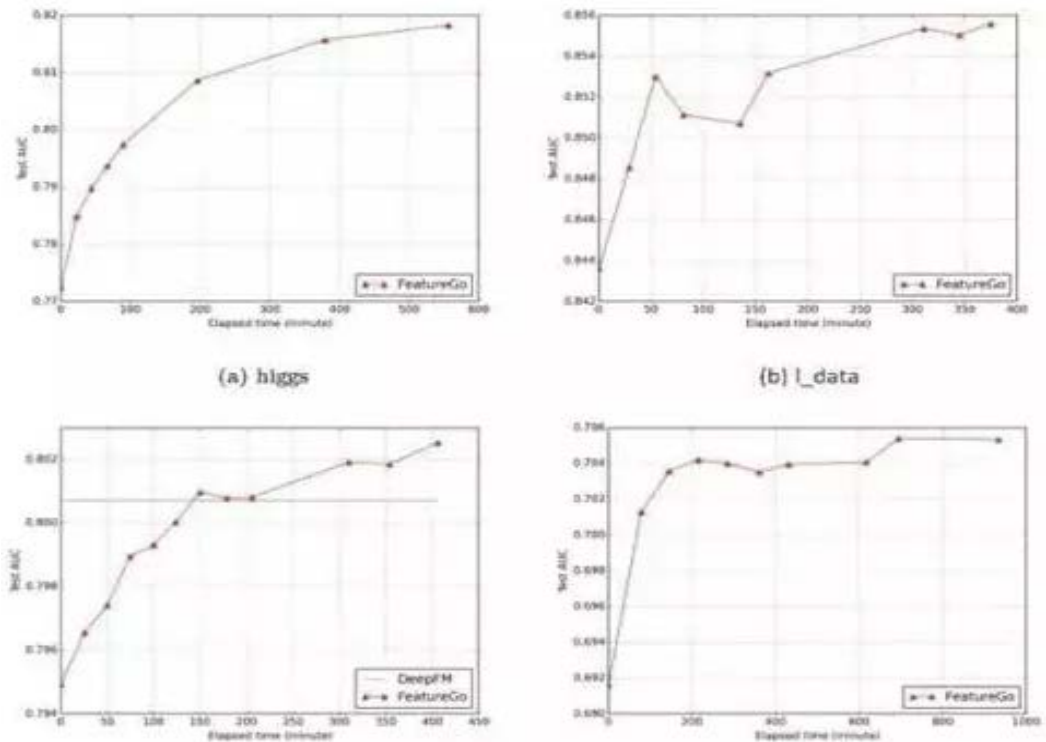
我们在 4 个数据及上实验了 FeatureGO 算法，包括两个公开的数据集（higgs、criteoDeepFM）与两个我们私有的数据集（l_data，m_data），数据集统计信息如下：

Dataset	#instance		#slot	#sign	
	#train	#test		#cont.	#disc.
higgs	10,000,000	1000,000	28	28	0
criteoDeepFM	41,256,555	4,584,062	39	13	33,762,577
l_data	1,888,366	1,119,778	27	8	120,836
m_data	2,340,209	1,059,016	76	55	4,126,417

在这四个数据集上，我们使用 FeatureGO 产生的特征集合加上 LR 模型进行训练，使用 AUC 作为评估标准。实验效果如下。可以看到，使用了 FeatureGO 进行特征工程，在 4 个数据集上的效果都有非常明显的提升，大部分提升在 AUC 2 个百分点左右。

	higgs	criteoDeepFM	l_data	m_data
LR	0.6828	0.7831	0.8387	0.6860
LR(FeatureGo)	0.8183	0.8025	0.8556	0.7053

我们也实验了效果随着时间与新组合特征的加入而发生的变化，如下图所示。可以看到随着时间的变长，特征组合数越多，特征组合效果会越来越好。



基线对比算法除了 LR 之外，我们也比较了一些最新的非线性算法成果，在 cretio 广告数据上进行实验，结果如下。在这个实验里，我们也可以看到，即使是基于最新 NN 或者 FM 的特征组合，也并不能找全所有可以利用的信息，这个对比中显式特征组合依然有非常好的表现。同时要注意的是，FeatureGO 产出的新组合特征还可以进一步提高上述所有模型

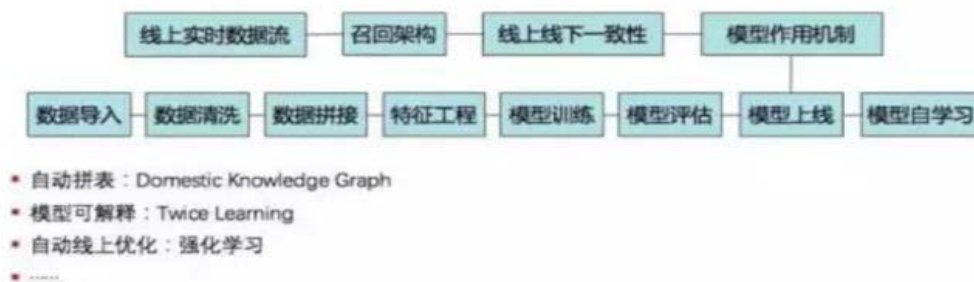
	Criteo	
	AUC	LogLoss
LR	0.7686	0.47762
FM	0.7892	0.46077
FNN	0.7963	0.45738
IPNN	0.7972	0.45323
OPNN	0.7982	0.45256
PNN*	0.7987	0.45214
LR & DNN	0.7981	0.46772
FM & DNN	0.785	0.45382
DeepFM	0.8007	0.45083
FeatureGO	0.8025	0.4501

的效果。

在 FeatureGO 这样的算法背后，其实还有好多隐藏的架构上的黑科技，让基于搜索的算法可以实际变成可行。比方说我们提出了 CPS（Cross Parameter-server Sharing），Dynamic Graph 等技术，将计算过程中公共的数据读取处理、特征请求存储更新等进行共享，达到同时训练 10 个模型的时候使用远小于 10 倍单模型时间的效果。

关于计算能力与架构，我们认为这是人工智能至关重要的一部分。过去将 APP 开机速度由 20ms 优化到 2ms 其实意义并不是特别大，但是在机器学习中，10 倍的速度意味着同时间训练 10 倍多的数据，或者同时能训练 10 个模型，也就意味着更好的效果，这也就代表着以往模型优化、效果优化这种只有科学家做的事情，现在优秀架构师也能做到。在这方面，谷歌是非常好的榜样。在过去，从没有人想过 LR 这么简单的模型也能获得良好的效果，但是通过极致的工程架构与实现优化，谷歌证明了使用上千亿特征，即使简单的 LR 模型也不比任何非线性模型差。第四范式也是一个架构工程优化和算法并重的公司。我们不仅做人工智能的通用平台，也会投入非常多的精力去优化速度与架构，希望能获得更强、更全面的人工智能水平和能力。

AutoML : AI for Everyone



说到计算与架构这一点，也要提到谷歌 Deepmind 团队最近做的另外一件工作 NEURAL ARCHITECTURE SEARCH WITH REINFORCEMENT LEARNING，通过强化学习框架自动进行神经网络结构的学习，有人提到这个工作为了自动获得一个网络结构，使用了 800 块 GPU 同时进行

训练，非常不实用。但我们认为，谈到将来的趋势，不会是通过专家来节省机器，降低机器的使用，而是通过机器节省专家降低专家的参与，因为随着摩尔定律，机器的计算成本会指数级的降低，但是专家的数量并不会指数增加。要让更多的领域用上人工智能，更少人力的 AutoML 是将来的必经之路。

作者介绍

陈雨强，世界级深度学习、迁移学习专家，曾在 NIPS、AAAI、ACL、SIGKDD 等顶会发表论文，并获 APWeb2010 Best Paper Award, KDD Cup 2011 名列第三，其学术工作被全球著名科技杂志 MIT Technology Review 报道。同时，陈雨强也是 AI 工业应用领军人物，在百度凤巢任职期间主持了世界首个商用的深度学习系统、在今日头条期间主持了全新的信息流推荐与广告系统的设计实现，目前担任第四范式首席研究科学家，带领团队研究、转化最领先的机器学习技术，着力打造人工智能平台级产品“先知”。

（八）第四范式首席科学家杨强教授：人工智能的下一个技术风口与商业风口

作者 杨强



作为华人界首个国际人工智能协会 AAAI Fellow、至今为止唯一的 AAAI 华人执委，以及 IEEE Fellow、AAAS Fellow、IAPR Fellow，杨强教授在专注学术研究的同时，也更关注如何让人工智能技术落地转化为生产力的问题。

作为第四范式首席科学家、范式大学的导师，杨强教授近日在第四范式公司内部进行了一场主题为“人工智能的下一个三年”的培训，深入浅出地分享了自己人工智能产业推广上的经验，并预判了人工智能即将爆发的技术风口与商业风口。此前，杨强教授与第四范式曾提出人工智能的五个必要条件，为人工智能行业提供了权威的准入标准。

以下内容根据杨强教授主题演讲编写，略微有所删减。

一、AlphaGo 为我们带来了什么

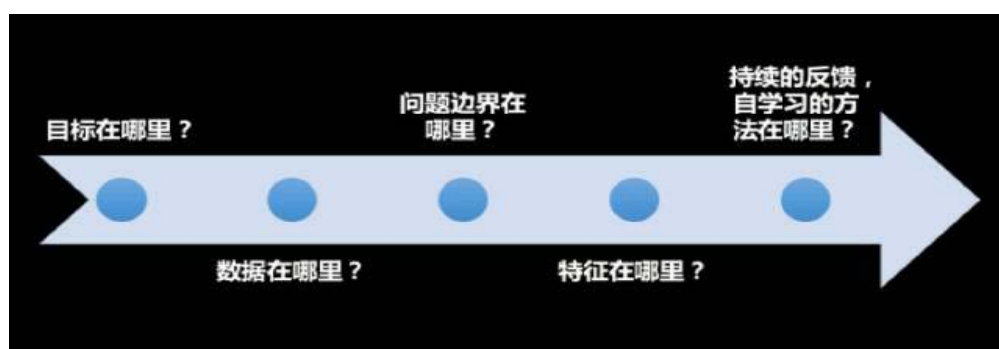
大家记得在 2016 年 3 月，AlphaGo 横空出世对战李世乭，这对于人工智能的社会影响非常大。这里，我们问一下：AlphaGo 到底为我们带来了什么？

在 AlphaGo 的搜索中，Deepmind 团队引入了一个新概念——即用深度学习和强化学习的结合来做两种任务的判别，即来判别现在所在的棋盘是好是坏，同时来预测未来有利的走向。讲到这里大家应该能看出 AlphaGo 的算法和未来商业模式的关联，即：通过对大数据的分析，让我们对“现在状态”有了一个靠谱的理解；这个状态可以是棋盘、可以是足球运动中两队交锋的状态，也可以是当前营销的一个状态。同时，下围棋中的一步，可以理解成对未来走向的预判，在商业活动中，这可以是营销活动中的下一步。这里很重要的一点，是区分我们商业行为中的两个任务，即对现实的判断和对商业未来走向的预估。这两个任务同样重要，也同样都需要大数据的支持。因为围棋是一个封闭式的游戏（即没有外界因素的干扰），为了得到更多的数据，AlphaGo 也引入了自我博弈。所谓自我博弈就是自己玩游戏，你会得到不断的反馈，然后来更新自己的策略，经过无数次这样的比赛，最后会得到一个好的策略，你的最终输出是一个行为的策略。所以 AlphaGo 也告诉我们，在一个封闭场景中，可以用自我博弈的模拟方法得到更多的数据。

从AlphaGo到人工智能的应用流程

我们如果沿着下围棋的步骤走，就要面对这些问题：你的人工智能算法的目标是什么？有没有数据？数据在哪里？问题的边界是否清晰？什么叫合理的走法、什么叫犯规的走法？你的特征在哪里？又如何得到这些特征？是否可以得到一个持续的反馈？这样的流程是 AlphaGo 设计团队所走过的路。不妨把这些步骤记下来，变成一个 workflow，看看其他的

领域是不是可以重复 AlphaGo 的成功。比如，如果用 AlphaGo 治疗癌症，如何治疗呢？治疗癌症一般是用放射性来杀掉癌细胞，而每一个癌症患者需要的剂量、角度、频次可能都不一样，如果能把所有的这些信息都记录下来，再记录治疗结果，因为结果不是马上就知道的，而是经过一段时间才知道，这样就有了数据、有了特征、有了问题持续的反馈，并且有了非常清楚的目标，即在副作用最小的情况下杀死癌细胞。并且这个 workflow 是可以重复的。



AI的发展历史还有前30年，这些年的积累也很有用

刚刚我们说了 AlphaGo 的一路历程，但我们对人工智能的理解不应该片面地认为人工智能就是机器学习。人工智能的发展历史还有前 30 年，前 30 年是从 50 年代中一直发展到 80 年代中。这 30 年 AI 是在干什么呢？是在做人工输入的规则型的知识表达研究，以及基于这些规则的符号空间的推理和搜索。我认为，这个人工规则型的知识表达在 AI 的应用当中也是必不可少的，因为在众多领域当中还会碰到冷启动的问题，以及如何规范一个领域的边界的问题。这就是说，逻辑推理，逻辑知识表达，以及在符号空间的搜索的人工智能这个分支，在今后几年会和统计学习相结合，会大有发展。这种发展会也涉及技术和商业两个层面。

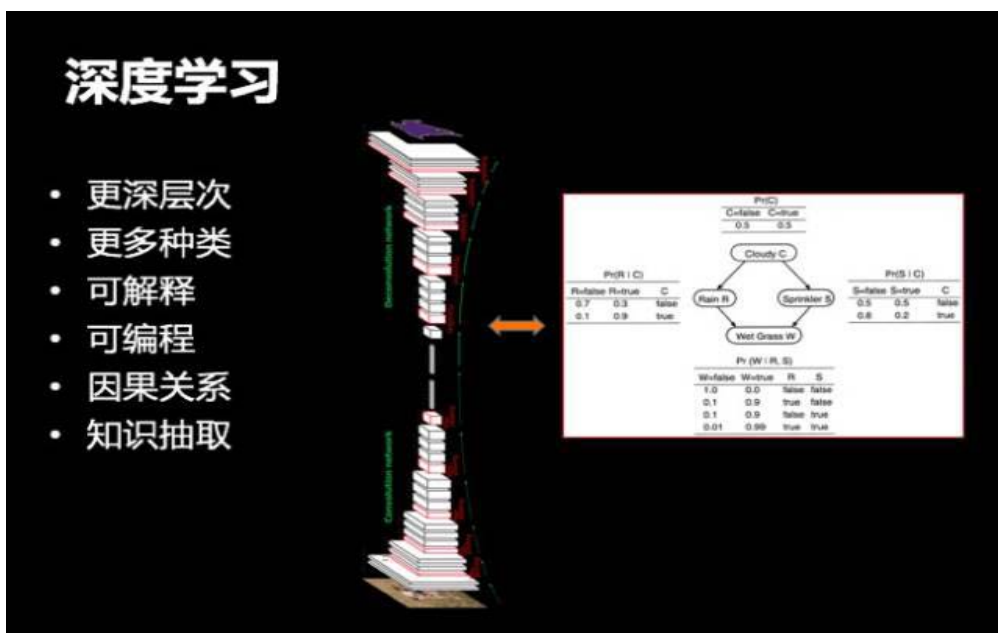
二、AI 的技术风口在哪？

我们大家会关心的一个问题，是人工智能的技术在哪些方向可能会

有大的突破。

深度学习

首先，是深度学习会继续发展。这里的发展不仅是在层次的增加，还包括深度学习的可解释性、以及对深度学习所获的结论的自我因果表达。例如，如何把非结构化的数据作为原始数据，训练出一个统计模型，再把这个模型变成某种知识的表达——这是一种表示学习。这种技术对于非结构化数据，尤其对于自然语言里面的知识学习，是很有帮助的。另外，深度学习模型的结构设计是深度学习的一个难点。这些结构在今天都是非常需要由人来设计的。还有一个研究问题是如何让逻辑推理和深度学习一起工作，这样也可以增加深度学习的可解释性。比如，建立一个贝叶斯模型需要有很多的设计者的经验，到现在为止，基本上是由人来设定的。如果我们能从深度学习的学习过程中衍生出一个贝叶斯模型，那么，学习、解释和推理就可以统一起来了。



迁移学习

迁移学习也是我和戴文渊（第四范式创始人、首席执行官）一直在做

的工作。给定一个深度学习的网络，比如一个 encoder 网络和一个 decoder 网络，我们可以看它学习和迁移的过程，作为新的数据来训练另外一个可解释的模型，也可以作为一个新的迁移学习算法的输出。即一个学生 A 在观察另外一个学生 B 学习，A 的目的是学习 B 的学习方法，B 就不断地在学新的领域，每换一个领域就为 A 提供一个新的数据样本，A 利用这些新的样本就能学会在领域之间做迁移。所以这种过程叫做观察网络。有了这种一边学习、一边学习学习方法的算法，就可以在机器学习的过程中，学会迁移的方法。

自然语言的表示学习与机器阅读

表示学习是当数据和任务没有直接相关时也可以学，一个重要的例子叫做 self-taught learning，即我们通过很多 supervise 的数据、图像，可以学出一种最好的表达。用这个表达加上任务，就可以很快地学会这种知识表示。这时非结构化的数据就相当有用了。比如，给出一段话让机器去阅读，机器学习可以自动地发现一些值得关注的点。比如，给定一个文章中的实体和一个未知变量有这样的关系，然后用户可以问你这个未知变量是什么。能够达到这样的效果是因为深度模型已经具有了一种关注，这种关注是可以通过观众的学习（Attention）来表达。其结果就好像我们一目了然地看了一本书，我们会把关键词和它们的关系抓取出来。这实际上是利用类似人的一种直觉来进行学习。

人机对话系统

应该说有一个领域已经发展到了临界点，就是人机对话系统领域。现在在这个领域，某些相对垂直的方面已经收取了足够多的数据，一个是客服，一个是汽车（车内的人车对话）；还有一种是特定场景的特定任务，像 Amazon Echo，你可以跟它讲话，可以说“你给我放个歌吧”或者“你播一下新闻”，Amazon Echo 里面是围了一圈的 8 个麦克风，这个阵列可以探测到人是否在和它说话，比如我和别人说话的时候，脸转过去，它就不会有反应。这种唤醒功能是非常准确的。它的另外一个功能是当你的双手

没办法去控制手机的时候，可以用语音来控制，案例场景是客厅和厨房，在美国 Amazon Echo 特别受家庭主妇的欢迎，所以像这种特定的场景，如果收集了足够的数据，是可以训练出这样强大的对话系统来的。

强化迁移学习

我们可以想象，未来深度学习、强化学习和迁移学习的结合，可以实现以下几个突破——反馈可以延迟、可以个性化，把一个通用模型施加到任何个体上面，这样一个复合模型可以叫做强化迁移学习模型。

人工智能的可靠性模型

AI as Reliable Services 是 AAAI 前主席 Thomas Dietterich 在 AAAI 2016 上给出的一个主题，人工智能只能作为一些例证证明能够做些什么事情，比如下棋，无人驾驶，但很多时候它还是不可靠的。它不像现在的一个商用软件一样，能让你放心地去使用，以保证它的错误率肯定不会超过很小的比例。相反，AI 在犯错的时候可能错得非常厉害，所以用平均值来代表一个准确率是不恰当的，相反，应该更多地要考虑它的置信区间。换言之，小白用户拿一些人工智能的模块来搭一个系统，这个系统就应该能被搭出来，而且它的效果应该是在一个固定的范围以内的，所以人工智能应该像软件工程一样做出来。

第四范式核心产品“先知平台”一直就在往这个方向发展，先知把人工智能的模块工程化、并在一定程度上保证了可靠性，从而让普通用户用来搭建自己的人工智能系统。

三、AI 的商业风口在哪？

上面我们考虑了人工智能的技术发展。下面我们看看商业领域。我们刚才列举了 AI 成功的 5 大必要条件：高质量的大数据、清晰的问题定义和领域边界、懂人工智能且擅长应用和算法的跨界人才、足够的计算资源、持续的外部反馈。满足这五个条件的领域，才有可能在未来出现人工智能的爆发。

智能客服

人机交互的智能客服，产生了很多外界公开的数据以及内部的数据、知识库等，都可以用来制造机器人。尤其是可以用客服过去的数据来做训练，这个数据量现在在垂直领域是逐渐在增加的。现在的对话系统也已经逐渐成为深度学习和强化学习的焦点。

新闻领域

另外一个比较看好的领域是新闻领域，新闻的分发和自动写作。有很多编辑、解说、自动校对、作家等，其实是数据量足够多的，有这么多的文本，而且外界反馈也越来越多了。给一篇文章，可以用机器学习来做自动摘要。这样一个工作的外部反馈来自哪里呢？实际上我们写的那些 paper 就是一个外部反馈，因为每篇 paper 都有摘要，如果一篇 paper 被收了，就说明摘要写的还不错，所以外部反馈还是可以实现的。

这里分享一个有趣的实验，是香港科大同学做的“自动写小说”项目。主要有两个步骤，一步是让它读很多书，一步是这样训练出一个模型，这个模型再让它变成一个生成式的模型，这样就能用来写小说了。举个例子，我们提供《射雕英雄传》和《笑傲江湖》，把这两个结合起来，就可以写一部新的小说了。

特定任务的智能机器人

例如 Amazon 的 KIVA 机器人，大家可能知道 Amazon 一个很大的优势就是所有的仓储都是由机器人来完成的，但是它也有工人，被雇来用手做抓取，因为现在机器人的抓取是非常难的，那么人和机器的优点就结合起来了。此外，医疗机器人也是非常专业的一个领域，它可以给人开刀缝线，但它不是自动的，而是通过远程控制的，但控制的精密度非常高，如果它收集到足够量的数据，是可以达到自动的效果的，以后我们可能开刀就由机器人来代劳了。

在医护领域，无障碍辅助的应用领域痛点特别强烈，现在数据量可能还不是特别多，因为毕竟这一群体还是少数人，但是痛点很强，所以未来

也许会有数据。

AI+有机食品

我们在香港曾去访问过一个有机食品工厂，这个实验室里的每一株菜，周边的所有环境全都记录起来，比如湿度、温度、光照，然后就可以收集这样的数据训练一个机器学习的模型，最后用这个模型来做蔬菜。所以得来的蔬菜滋味可以控制，要脆感还是要甜的，都可以通过模型学习出来。

FINTECH智能投顾

最后来说一说金融，其实金融是一个非常好的领域，第四范式在金融领域也积累了很多成功案例。金融领域里的任务都是非常清楚的，而且每个任务的数据都有痕迹、有数据足迹，数据的维度也是多维度的数据，有外界的、也有内界的，非结构数据比较多，例如文本和报告。数据也是形成了孤岛，链条也非常长，并且链条里面都有衔接。

在金融领域现在美国比较时髦的一个概念叫投研、投顾和投资。投研是说研究整个市场的基本面，就好像我们研究舆情分析一样，但舆情只是其中的一部分；投顾是说在美国的银行给很多客户做理财分析，然后做理财的配置，这些工作可以由机器人来做；投资是说机器人自己就是一个客户，它可以去投资。

四、多年后的 AI 社会

最后说一下我认为多年后的 AI 社会是怎么样的。我觉得未来应该是几个人在运行一个公司，每一个人都能率领成千上万个机器人，这些机器人在做不同的事情，也是它被训练得很擅长的事情。我们现在在一个传统行业里，往往是 20% 的人在干 80% 的工作，那么这 20% 的人就是未来的运营公司的人，剩下 80% 的人所做的工作将交由机器来完成。一个公司的自动化、智能化程度，也代表了这个公司在商业上的反应速度和竞争力。

人工智能给人类带来的变革是非常深远的，人工智能不仅仅是一场比赛、一个应用，而是整个社会真正地彻底地在改变。机器和人将成为一个共同的“军队”不断地攻克堡垒，推动人类进程向更好的方向发展。

版权声明

InfoQ 中文站出品

架构师特刊：范式大学

©2017北京极客邦科技有限公司

本书版权为北京极客邦科技有限公司所有，未经出版者预先的书面许可，不得以任何方式复制或抄袭本书的任何部分，本书任何部分不得用于再印刷，存储于可重复使用的系统，或者以任何方式进行电子、机械、复印和录制等形式传播。

本书提到的公司产品或者使用到的商标为产品公司所有。

如果读者要了解具体的商标和注册信息，应该联系相应的公司。

出版：北京极客邦科技有限公司

北京市朝阳区洛娃大厦C座1607

欢迎共同参与 InfoQ 中文站的内容建设工作，包括原创投稿和翻译，请联系 editors@cn.infoq.com。

网 址： www.infoq.com.cn