



中国计算机学会
China Computer Federation

数据驱动的算法 Data driven algorithms

阿里巴巴 姜碧野

jby.yeah@gmail.com



摘要

- 本次授课内容主要包括高维向量空间的常用算法，大数据应用里的常用算法，以及会介绍机器学习、高性能计算里一些前沿的相关知识。与传统竞赛里主要关注最坏情况复杂度不同，工业界应用中往往关心的是在实际数据上的复杂度，因此“数据驱动”的算法分析显得格外重要。所以除了介绍算法本身之外，也会探讨在现有的竞赛框架下如何对这些带有不确定性的问题进行出题和考察。



个人简介

- 毕业于广东中山纪念中学，2008年入选信息学国家集训队。本科就读于清华大学计算机系，博士毕业于加州大学伯克利分校，机器学习/系统方向。在校期间多次获ACM分赛区冠军，并在15年代表伯克利参加World Final获全球第6名，目前在阿里巴巴(北京)工作。



中国计算机学会
China Computer Federation

大纲

- 主题概览
- 高维向量算法
- 大数据算法
- 优化算法和机器学习
- 高性能计算





Data Science vs. Computer Science

- 在过去的几十年里，计算机在操作系统、编译器、数据库，网络通信等领域得到了极大发展。处理这些离散数据的算法和数据结构也成为计算机科学的主要研究对象
- 如今和未来几十年，人工智能、机器学习、信号处理等数据科学成为新浪潮。与之相关的是对高维空间、连续数据、自然数据的处理。什么会是新的基础算法和数据结构？

By John Hopcroft in his book 《Foundations of Data Science》



Classical algorithms vs. Modern algorithms

- 离散(序列、树、图) vs. 连续(向量、函数)
- 精确的时间复杂度 vs. 基于实际数据的复杂度
- 组合优化算法 vs. 概率、线性代数、连续优化



比赛形式影响解题策略

- OI比赛：数据不可见，解题时不可提交看结果
- ACM, TopCoder：可提交，但有罚时
- CodeJam, IPSC：数据可见
- Kaggle数据科学比赛：数据可见，比赛周期长
- 科研、工业界的实际问题：有标准公开数据集



Worst case analysis vs. Data driven analysis

- OI竞赛往往关注最坏情况复杂度
 - 典型：字符串匹配 KMP算法
 - 考察选手的算法分析、验证能力、代码正确性
- 实际工业界应用关心在真实的数据集上效果如何
 - 典型：搜索引擎



中国计算机学会
China Computer Federation

大纲

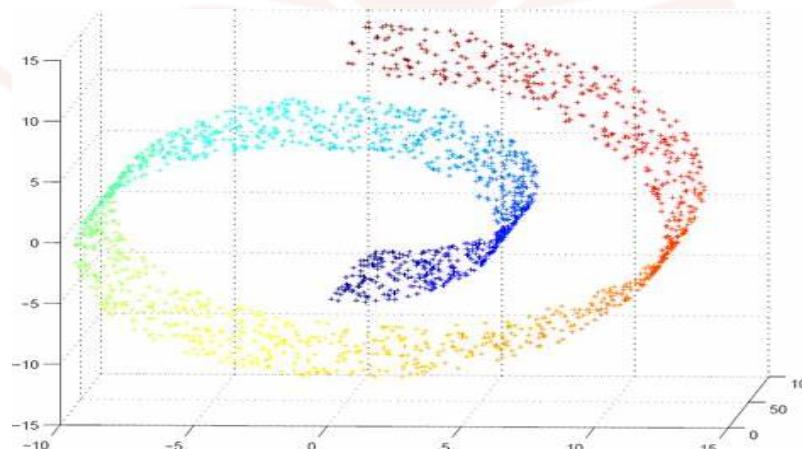
- 主题概览
- **高维向量算法**
- 大数据算法
- 优化算法和机器学习
- 并行计算





高维向量

- 高维向量是数据处理里面最常用的基本数据结构
- 在机器学习、搜索引擎、图像处理等等各行各业有广泛的应用
- 一个高维向量可以代表一张图片、一篇文档、一段音频、一个用户信息
- 高维向量往往具有低维子结构！





简单情况：三维

- 美国ACM某赛区练习赛题目：
 - 在一个三维空间中，有N个三维向量。在它们中找到最近（夹角最小）的一对向量 ($N \leq 50000$)



三维向量最近邻

- 回顾平面上最近点对的分治算法
 - 利用平面的特性，先计算两侧的最优解，再得到分界线附近最优点对数目可枚举的特性
 - 欧氏距离 $(a-b)^2$ 与点积 $a \cdot b$ 存在相关性
- 但如何扩展到三维？



中国计算机学会
China Computer Federation

二维向量最近邻

- N个二维向量，求最小夹角怎么做？
- 等价于在一维求最近点对，直接按向量角度排序即可



二维扩展到三维

- 如果把三维向量投影到二维平面
 - 夹角只减不增
 - 原三维向量夹角小 → 投影后的二维夹角也小
 - 投影后的最近向量，不一定在原空间就接近
- 随机选择若干个投影平面，投影后先排序，再在每个向量附近枚举最优解
- 当前最优解可以作为枚举时的剪枝条件



中国计算机学会
China Computer Federation

启发

- 低维空间有时候具备很多有趣的性质
- 利用降维算法，把高维转成低维



中国计算机学会
China Computer Federation

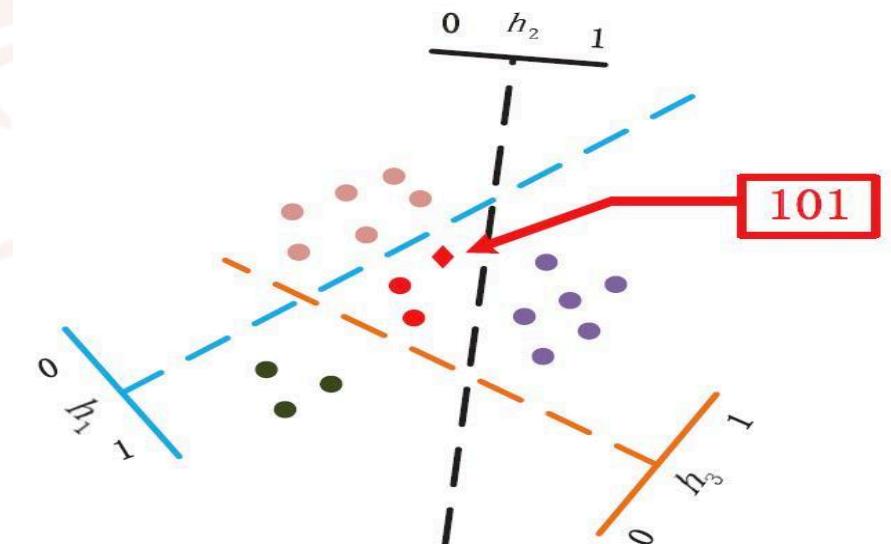
高维向量检索

- 给定一个高维向量，在候选集合中找到与其最接近（夹角最小/点积最大）的向量
- 机器学习、搜索引擎等应用里的核心问题



Locality-Sensitive Hashing

- 利用K个切割平面，把空间切割成 2^K 个子空间，每个子空间用一个长度为K的二进制串唯一表示。
- 对于查询点，枚举与其处于同一个子空间的候选集，找到最优解





Locality-Sensitive Hashing

- 优点：
 - 当候选集比较均匀地分散在各个子空间时，可以大大加速查询效率
- 缺点：
 - 当某些子空间存在大量候选集时（区分度不够），查询效率将大大减慢
- 如何选择切割平面，使得分割较为均匀，是各类LSH扩展算法的核心问题。而且实际算法的效果也非常取决于数据分布本身



切割平面与线段树、KD Tree

- 线段树解决的是最简单的一维情形，分割方式确定可控
- 二维平面的索引构建，就开始存在不确定性了。KD Tree采用横切和纵切交替的方式



切割平面与线段树、KD Tree

- 各种树结构均选择沿着坐标轴进行切割
- LSH对切割平面不做假设，但两个不完全一样的编码，如何分析距离？



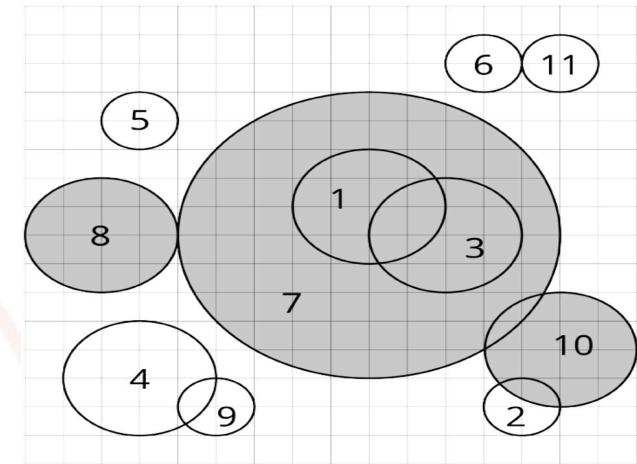
线段树常考而KD Tree不常考

- 线段树由于其确定性，导致其变种在OI竞赛里特别常见。
- 但由于其只能处理一维简单情况，其实在实际应用中很少用
- KD Tree, LSH这类算法在比赛中由于数据未知，所以很难确定其使用情况
 - 为什么不一开始就提供数据？



APIO2018 选圆圈

- 在平面上有n个圆
 - 1. 找到目前半径最大的那个圆 c_i
 - 2. 删除 c_i 以及与 c_i 相交的所有圆
 - 3. 重复1,2直到所有圆被删除
 - 对每个圆，输出它是被哪个圆删除的
-
- 解法：
 - 用KD-tree加速查找，可能会退化成 $O(n^2)$
 - 持久化线段树， $O(n\log^2 n)$ ，实际运行速度较慢





中国计算机学会
China Computer Federation

大纲

- 主题概览
- 高维向量算法
- **大数据算法**
- 优化算法和机器学习
- 高性能计算





大数据算法

- 基本都是线性时间复杂度，且内存受限
- 自然数据往往服从Power Law (二八定律、长尾定律)
- 需要考虑并行计算、分布式计算
- 在搜索引擎、推荐系统等互联网产品里应用广泛



集合大小估计

- 问题：假设你从一个数据流中不断地读取数据 d_i ，但你只有常量级别的内存用以存取中间结果。现在需要你在任意时刻，估算当前已读取的数据里，不同的元素的个数。也就是集合 $\{d_i\}$ 的大小



集合大小估计

- Minhash算法
 - 假设数据在某个hash空间里均匀分布
 - 集合大小N越大， hash空间里撒点的密度也越大→hash值间的平均间距越小→所有hash值中的最小值也越小
 - 用hash值的最小值来预估数据间隔，进而预估N
- 扩展：
 - 如何求交集并集的集合大小？



中国计算机学会
China Computer Federation

搜索引擎

- 传统字符串匹配算法：
 - KMP, Trie
- 如何实现搜索引擎检索？
 - 海量候选集
 - 检索词出现在任意位置



搜索引擎实现

- 倒排索引
 - 以单词/词组为粒度，构建单词->文档列表的索引链
 - 把查询拆分成($\text{word}_1, \text{word}_2 \dots \text{word}_m$)，将每个单词对应的文档列表求交集
 - 再利用机器学习算法对候选集进行个性化排序



KMP vs. 倒排索引?

- KMP的复杂度，是数据无关的，恒定为 $O(N+M)$
- 为什么OI竞赛里，我们不用倒排索引？
 - 我们假设单词集的可能空间是 128^N
 - 546G^{HHRTE} $^{\$\%\$%\#}$ 有可能出现在数据里
 - aaaaaaaaaaaaaaaaab有可能出现在数据里
- 真实世界的文本数据
 - 常用词百万级别，单词服从Power law分布。词组个数也有限
 - To be or not to be?



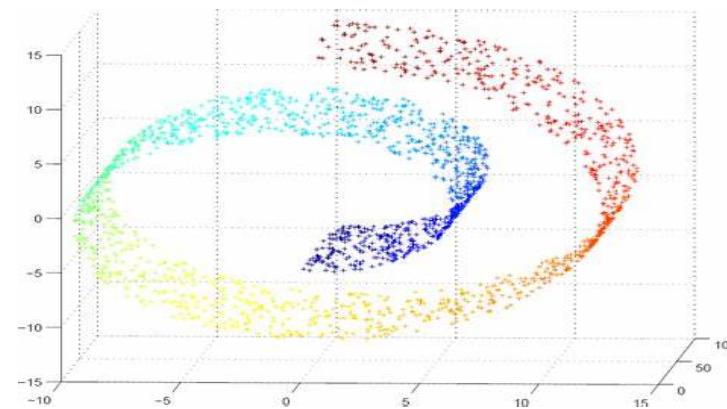
倒排索引 vs. 向量检索

- 每个“单词”，也可以看做是对候选文本集合的一个划分，或是一个hash编码
- 实际应用情况：
 - 图片检索
 - 基于用户画像的商品检索、新闻检索



小结

- 大数据时代，高维空间、聚类、低维子结构是非常重要的思维方式
- 实际问题比worst case简单得多
- 那么，如何找到数据里的低维子结构？





中国计算机学会
China Computer Federation

大纲

- 主题概览
- 高维向量算法
- 大数据算法
- **优化算法和机器学习**
- 高性能计算





中国计算机学会
China Computer Federation

我们的老伙伴

- 顾研：《浅谈随机化思想在几何问题中的应用》
- 《骗分导论》
- 模拟退火骗分大法
- 随机大法好，乱搞出奇迹



中国计算机学会
China Computer Federation

旅行商问题

- 给定一个N个点的无向图，找到里面的一条全遍历路径，使得经过的边的权值最小. $N < 30$
- TOJ某题



旅行商问题

- 随机生成一个N的全排列
- 每次随机选择排列里两个点的位置，如果交换后，路径总代价变小，则交换
- 如此往复，直到找不到这样的两个点为止



N皇后

- 求N皇后的可行解（皇后攻击范围是横竖对角线，需要在 $N \times N$ 的棋盘上放置N个皇后，使她们不能互相攻击。）
- $N \leq 1000000$
- Credit to Chenqifeng



N皇后

- 随机生成一个N的全排列，作为皇后在每行的位置，如此只存在对角线上的互相攻击
- 计算在该排列下，各条对角线“皇后冲突数”的平方和（统计 $i+j$ 和 $i-j$ 重叠的数量）
- 每次随机选取排列里两个皇后的位置，如果交换后，冲突数平方和变小，则交换
- 如此往复，直到找不到这样的两个皇后为止



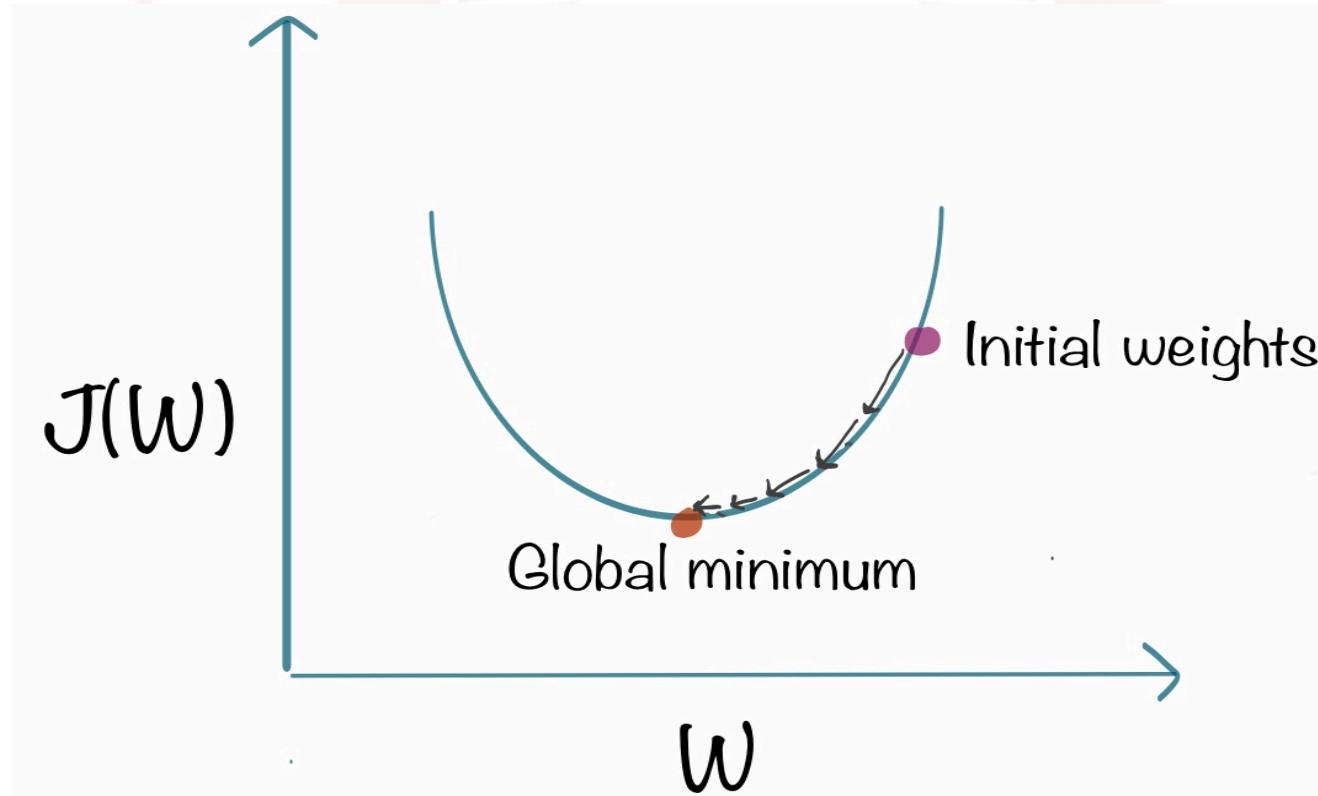
为什么随机调整效果这么好？

- 模拟退火其实就是gradient-free optimization里面的一种
- 随机调整本身就等价于我们在随机寻找离散函数在局部的梯度下降方向



什么是梯度下降？

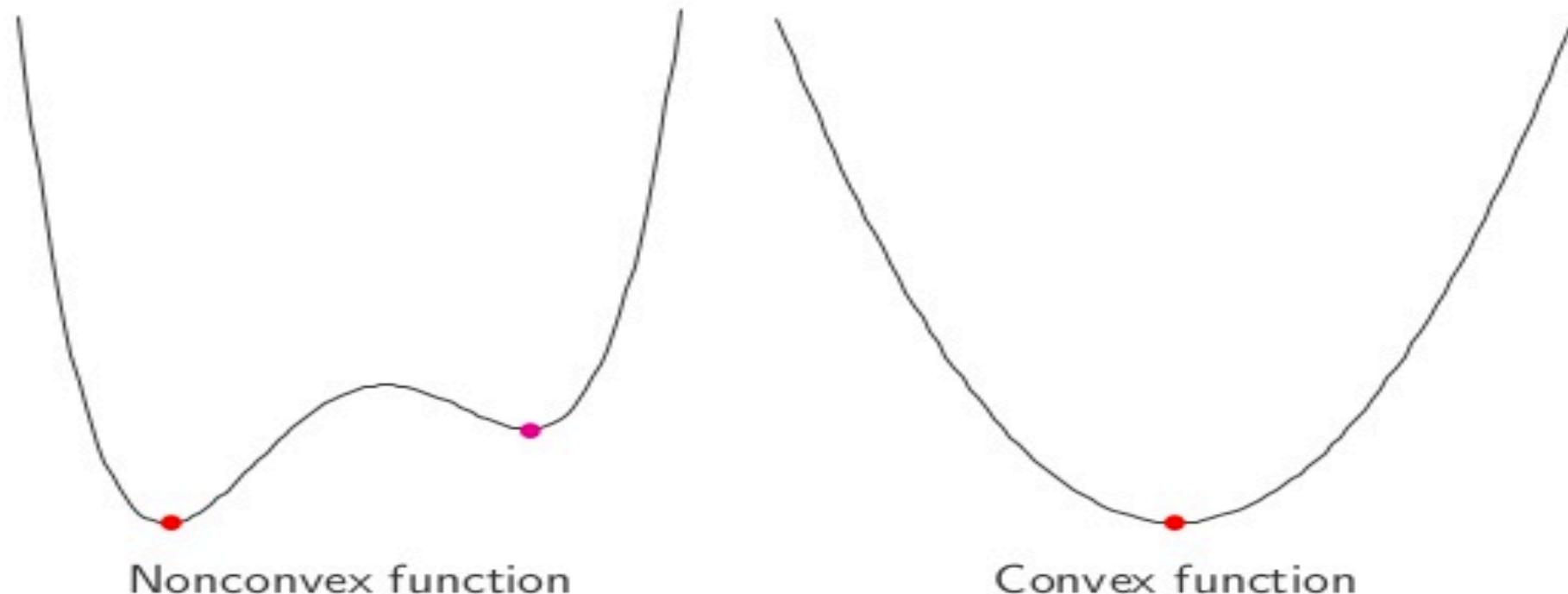
- 现代机器学习里面最重要的一个“算法”





凸优化 vs. 非凸优化

- 非凸函数：局部最优不一定是全局最优





非凸优化

- 实际情况：Gradient-descent works surprisingly well in deep learning
 - 《Gradient descent finds global minima of deep neural networks》
 - 《An Alternative View: When Does SGD Escape Local Minima?》
- 实际情况：模拟退火、加噪音都可以帮助算法逃出局部最优解
- 真“随机大法好”



Optimization

- Let me describe this stage of intellectual development. You read a couple of books and you wake up at 3:00 in the morning and say oh my god, everything is an optimization problem.

-----Stephen Boyd



中国计算机学会
China Computer Federation

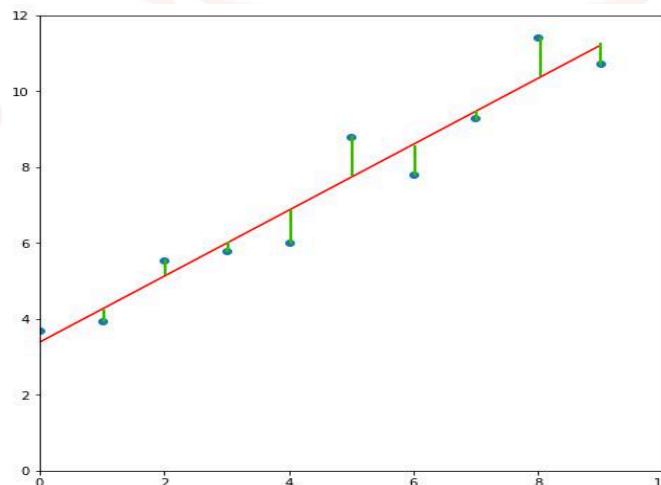
再看N皇后

- 把一个寻找结构型可行解的问题，转换成一个优化问题
- 用Loss function度量当前解与目标解的距离
- 机器学习算法工程师
 - Loss function designer
 - Landscape designer



从最小二乘法说起

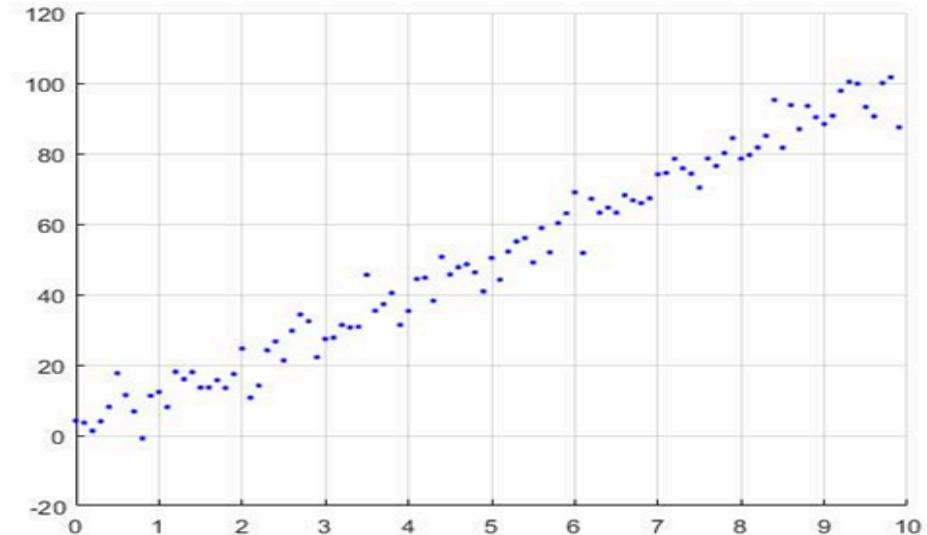
- 给定二维平面的N个点 (x_i, y_i) , 寻找一条直线 $y=ax+b$ 去拟合这些数据, 使得 $\sum \|y_i - \hat{y}_i\|^2$ 最小。其中 $\hat{y}_i = ax_i + b$
- 解法: 展开函数式, 优化目标是关于a,b的二次凸函数, 求得导数为0的点即可





从最小二乘法说起

- 重要概念：
 - (x, y) 是已有的数据
 - a, b 是待优化的函数参数
 - $\sum \|y_i - \hat{y}_i\|^2$ 是优化目标
- 核心假设：
 - 优化得到的 a, b 对于新的 (x, y) 同样适用
 - 机器“学习”的核心问题：泛化性





最小二乘法简单么？

- 看起来只能解决线性问题
- 可以通过基底函数扩展到任意函数，优化问题依然是线性的
- OI竞赛可以考最小二乘法么？

$$y = e^{ax+b}$$

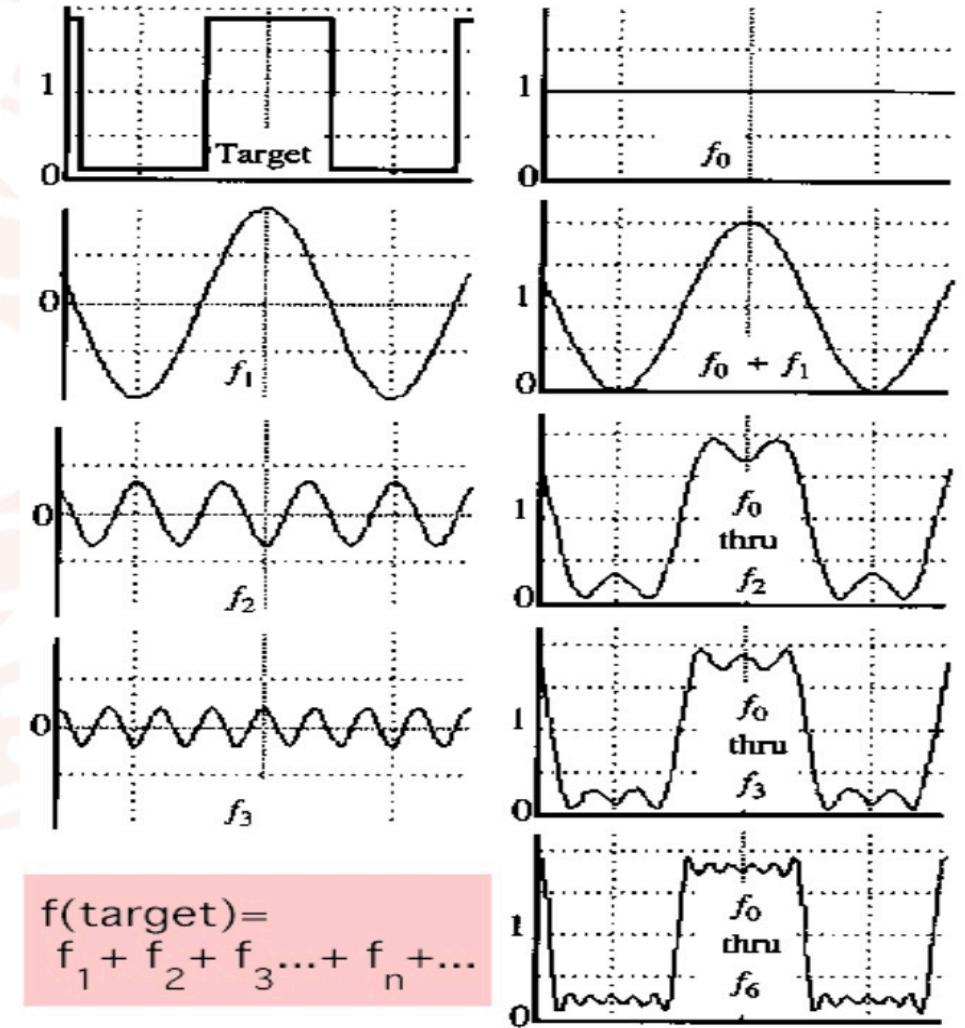
$$y = ax^2 + bx + c$$

$$y = \sum a_i f_i(x)$$



大学数学里很重要的两门课

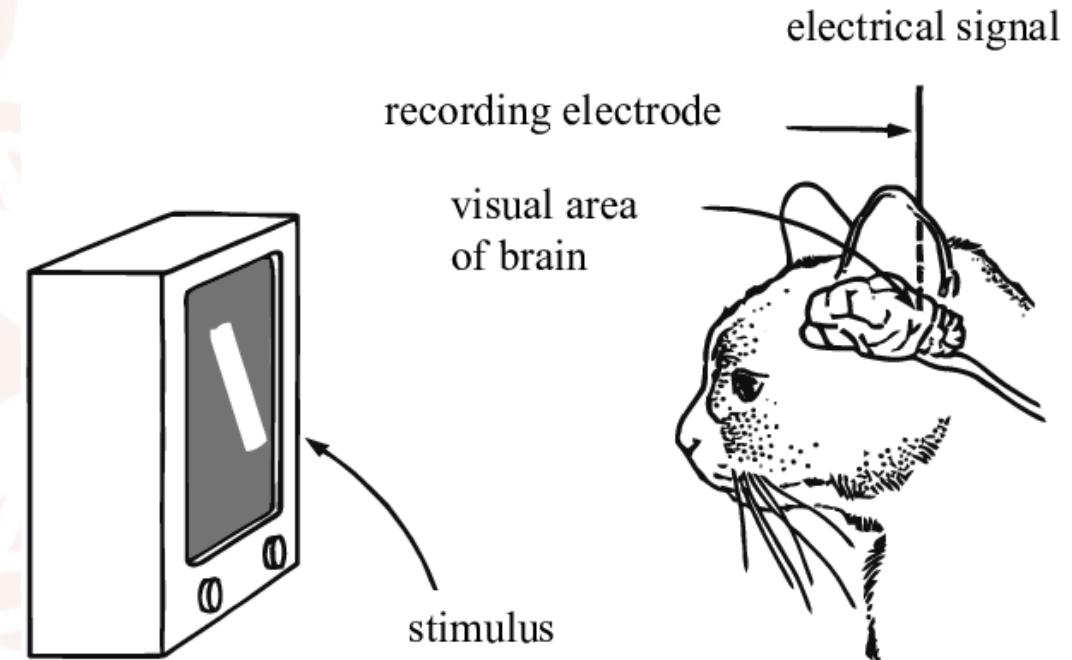
- 线性代数
- 信号处理/傅里叶变换
 - 任何周期函数，可以被表示为无穷多个正弦/余弦函数的带权和





自然信号里面是否存在基底?

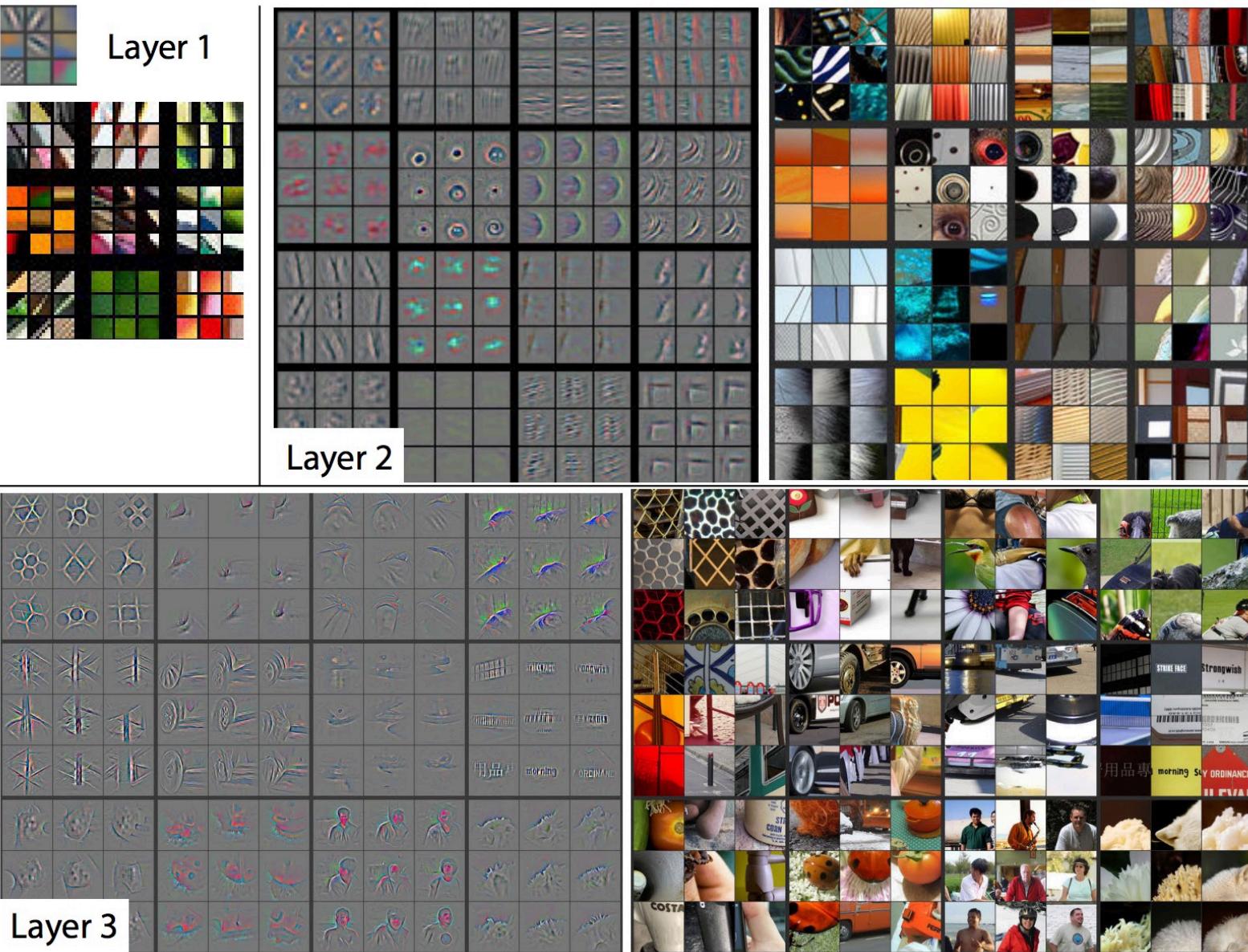
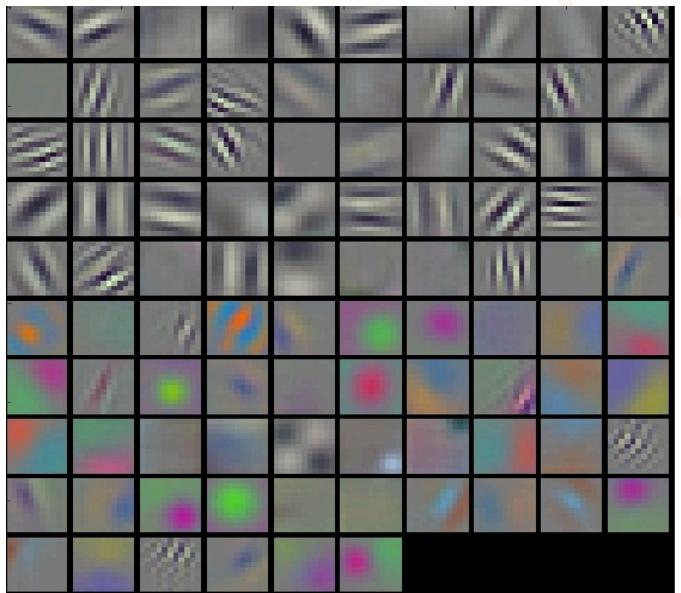
- 当然!
- 音频信号可以用正弦余弦函数表示
- 自然语言可以用单词、词组表示
- 图片信号同样有基本元素
 - 动物大脑里专门处理基本图形的V1区域在1950年代被发现 (Hubel and Wiesel 1981年获诺贝尔奖)



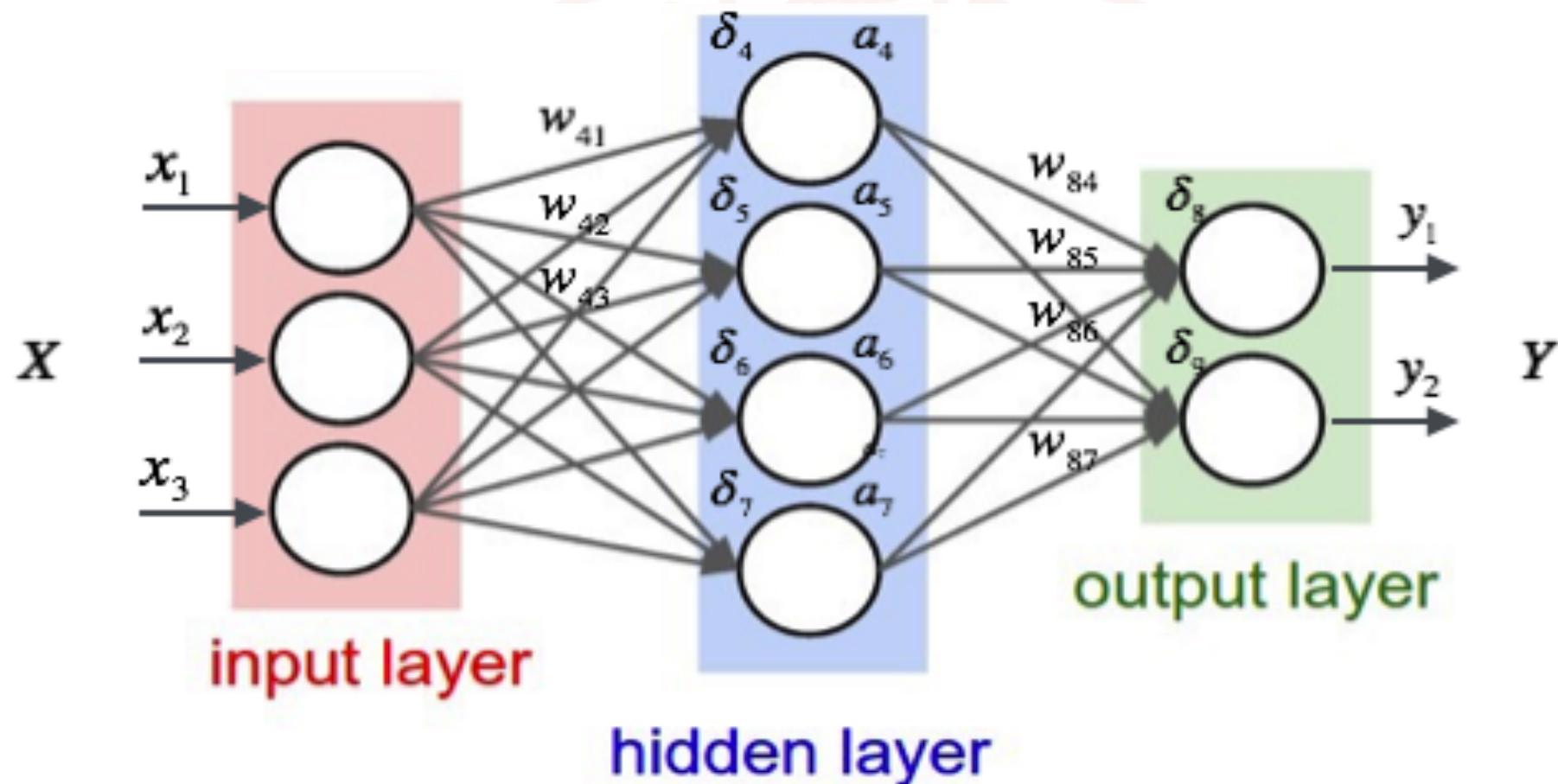


深度学习

- 深度学习学到的基底



深度学习：人工神经网络





深度学习

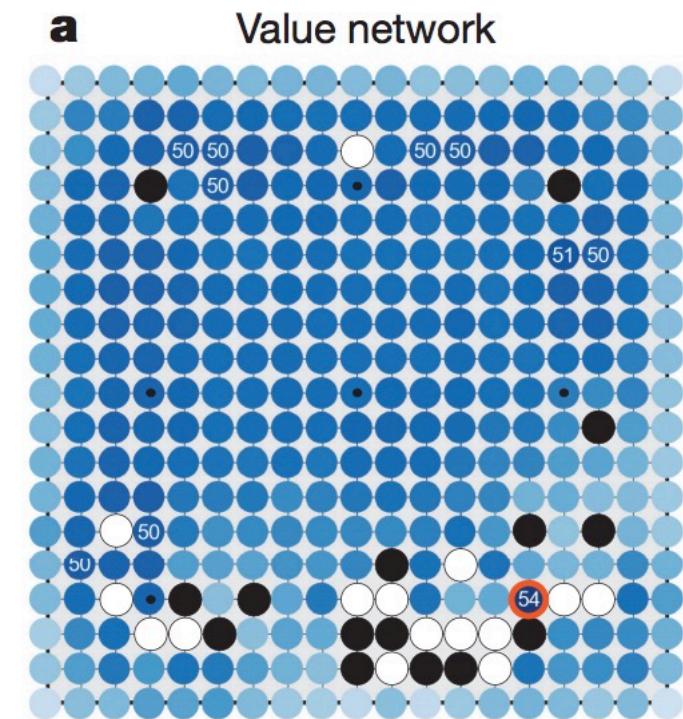
- 归根结底，深度学习是一个**多层次**的优化问题
- 每一层次本身是一个**线性优化**
- 多层级联构建了一个复杂的**可导**函数

$$\min_{\theta} \sum_i \| f_{\theta}(x_i) - y \|$$



AlphaGo: 如何基于数据优化搜索?

- 上一代的搜索算法：
 - A*启发式搜索、AlphaBeta剪枝
- 基于Reinforcement learning的搜索：
 - 先基于人类棋谱学习策略估价函数 $f(x)$
 - 从棋盘状态到“该下哪一步”的估价分布
 - 基于Monte Carlo Tree Search对终局估值
 - 双手互博





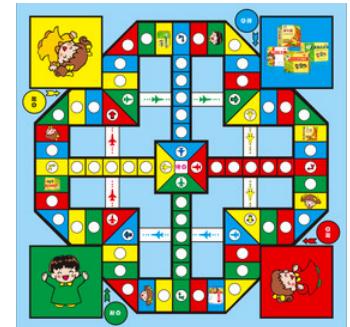
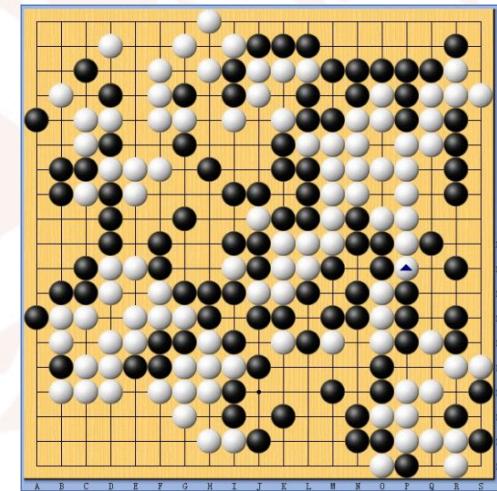
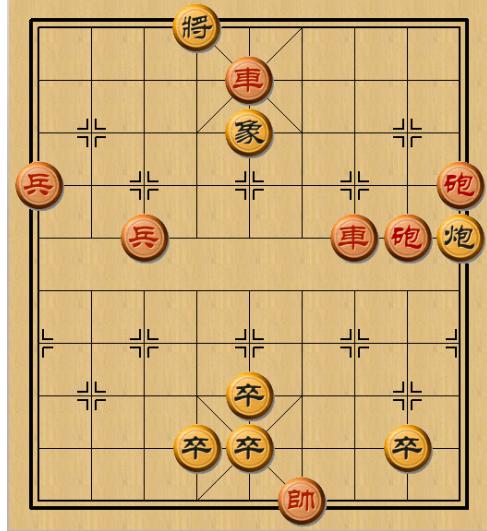
估价函数

- 什么是估价函数?
 - 回忆旅行商问题，搜索了一半的路径后，如何预估后面节点的最大最小路径长度？
- 拿到估价函数后，我们如何使用?
 - 剪枝、优先级选择
 - 类比：网络流SAP算法的“当前弧”优化



对棋盘游戏如何估价？

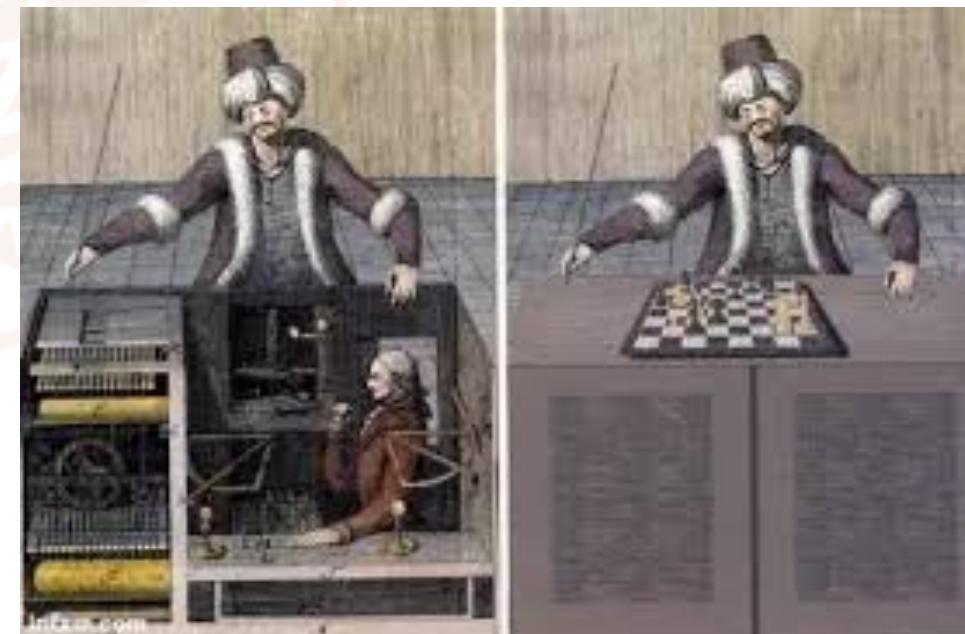
- 人工规则：
 - 五子棋可以数连续串的长度分布
 - 象棋可以去数不同类型兵种个数
 - 围棋呢？
- 基于数据的评估
 - 假设你拿到了一百万局人类下棋的历史数据，如何评估当前棋局？





棋局评估：离线查询 vs 在线查询

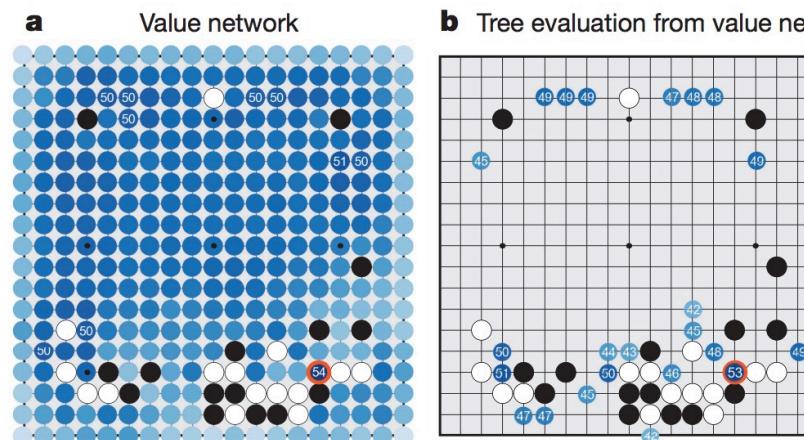
- 假如在你的棋谱数据库里，有一盘棋跟你目前遇到的棋局一模一样，是否可以直接拿来用？
 - 如果棋谱数据库足够大 ≈ 有一个下棋专家实时地帮你下棋
- 如果棋谱库有“类似的”棋局，如何使用？





如何利用历史棋谱数据

- 人的学习
 - 分析棋谱，自我模拟
- 机器学习
 - 计算棋谱相似性？并不需要完全一样，旋转、反转、局部变形影响不大
 - 局部的策略？4*4的围棋小棋盘，总共 $3^{16} = 43046721$ 可能性





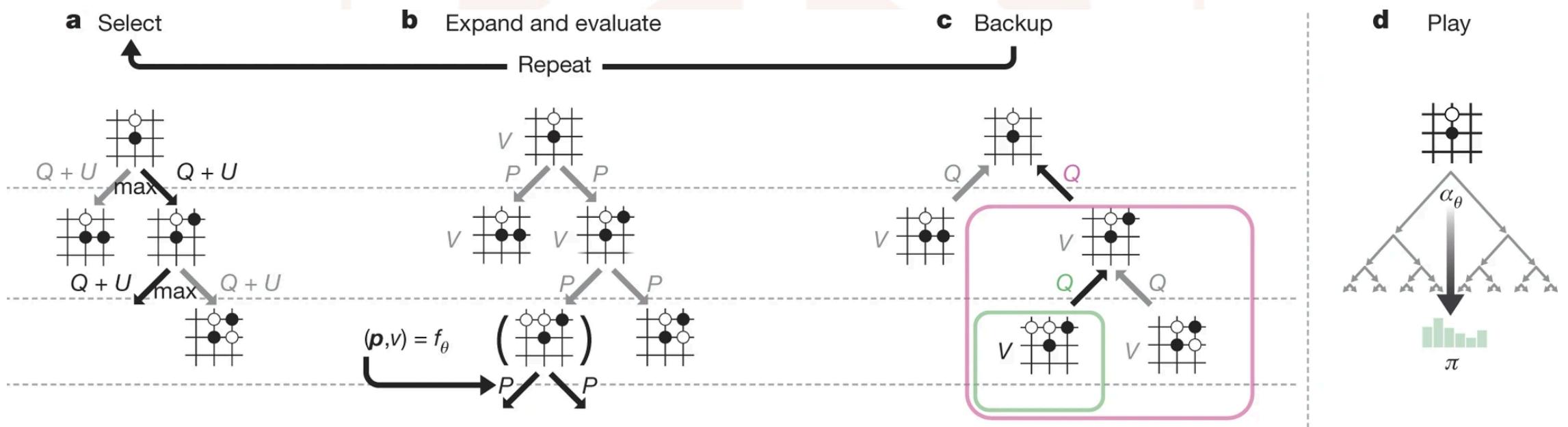
Monte Carlo Tree Search

- 再看旅行商问题，估价函数可以做得更好么？
- 通过不等式估计上下界虽然“准确”，但是“精度”不够
- 如果我们就随机地走几条路径试试呢？
 - 抓大放小，用局部估计整体



Monte Carlo Tree Search

- 搜到一定层次后，模拟博弈双方基于策略函数 $f(x)$ 随机下棋直到分出胜负，重复多次用统计结果来预估初始状态对谁更有利





小结

- 机器学习的框架，跟最小二乘法如出一辙
- 很多问题，可以被机器学习解决，不是因为算法本身强大，而是因为问题本身存在低维子结构
- 用优化算法+数据+对信号的理解找到这种结构
- 基于数据，可以显著提升搜索的效率
- 高中生熟练掌握机器学习，一点问题没有



中国计算机学会
China Computer Federation

大纲

- 主题概览
- 高维向量算法
- 大数据算法
- 优化算法和机器学习
- 高性能计算





算法复杂度

- 当我们在说一个算法的时间复杂度、空间复杂度的时候，我们在说什么？
- 不是说背包是NP问题嘛？为什么可以用DP
- 不是说排序的下界是 $O(n \log n)$ 嘛，为什么桶排序是线性的？
- 同样的算法，在图灵机上的计算次数，跟在CPU上一样么？



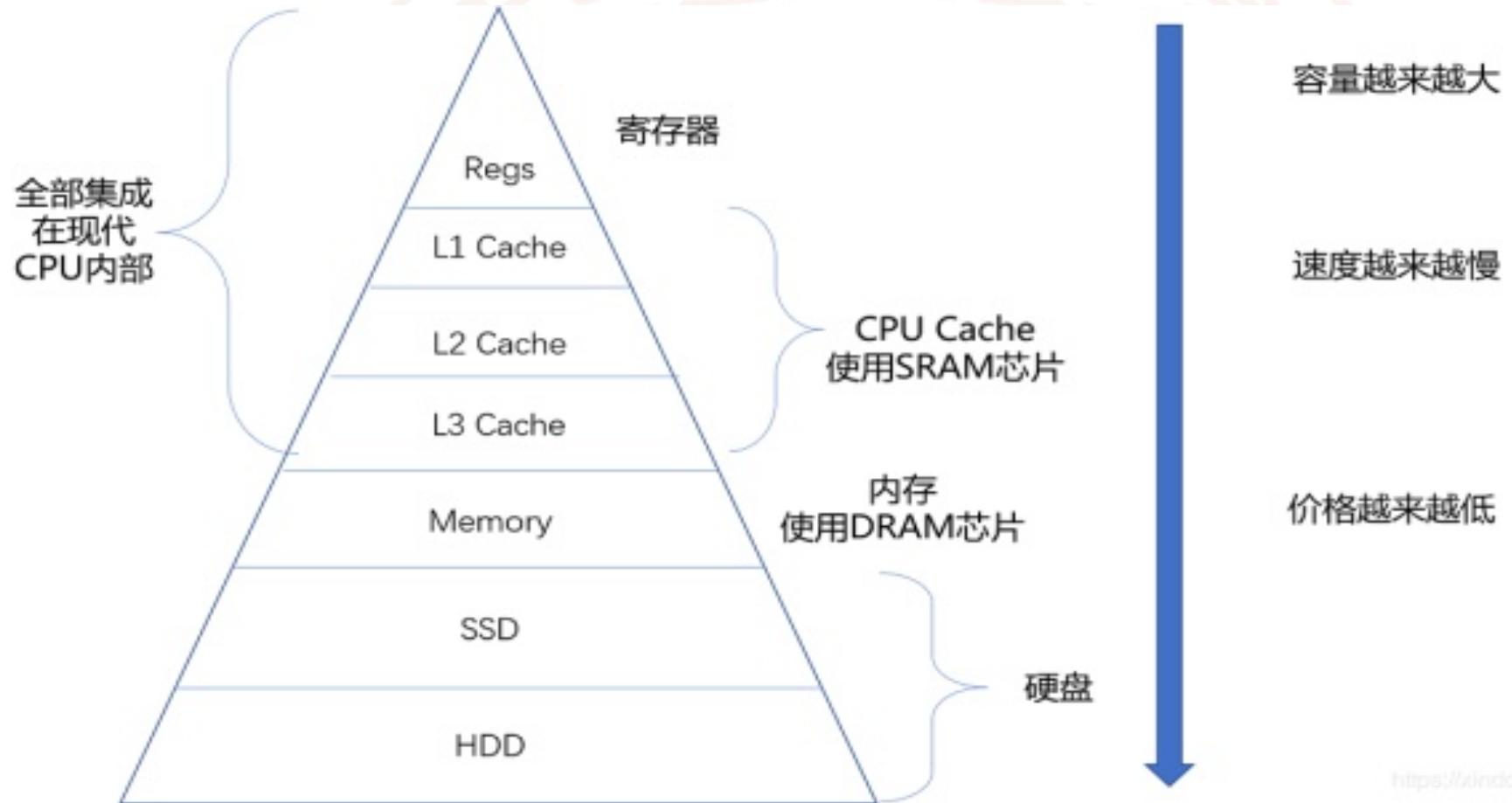
算法复杂度

- 算法复杂度与问题的定义方式, 以及硬件的实现方式 (computational model) 密切相关
- 我们平时说的时间复杂度, 一般指在Random Access Machine上的指令复杂度
 - 每次读写内存时间相等, 每次运算时间相等
 - MIX in TAOCP



实际情况

- 计算机是层级存储结构，不同数据的访问时间大不相同





高性能计算

- 骆可强：《论程序底层优化的一些方法与技巧》
- 压位法
- 位运算
- 常数优化
- “时间”复杂度，与硬件结构密切相关
 - 关注真正耗时的计算指令的数量
 - 比如在数据库应用里，B-tree由于树的查找深度低而广泛应用



矩阵乘法

- 如何写一个快速的矩阵乘法?
- N^2 规模的数据, N^3 规模的计算
- 分块算法, 分小块去计算矩阵乘法, 每次需要的数据都先读取到高速缓存, 尽可能地减少内存读取次数

- **Theorem (Hong & Kung, 1981): Any reorganization of this algorithm (that uses only associativity) is limited to $q = O((M_{\text{fast}})^{1/2})$**
 - #words moved between fast and slow memory = $\Omega(n^3 / (M_{\text{fast}})^{1/2})$
- Hong/Kung theorem is a lower bound on amount of data communicated by matmul
 - Number of words moved between fast and slow memory (cache and DRAM, or DRAM and disk, or ...) = $\Omega(n^3 / M_{\text{fast}}^{1/2})$



中国计算机学会
China Computer Federation

CUDA GPU并行计算

- 成千上万个核同时计算
- 但是.....数据从哪来？核之间如何通信？
- 需要摒弃传统的串行化编程思维



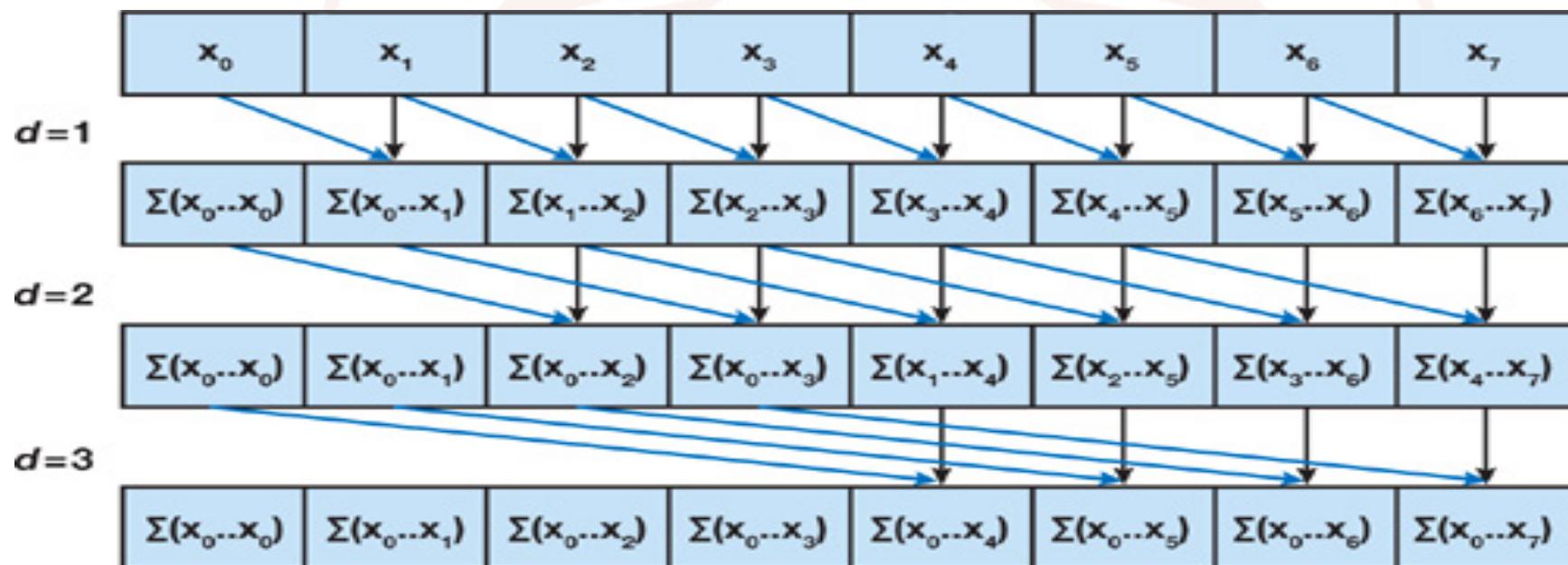
CUDA计算前缀和

- 给你一个长度为N的数组，计算它的前缀和
- 每一轮计算，假设有无穷多个核供你调用，但是每个核只能做一次运算，往一个地址写结果
- 用最少的轮数完成整个计算



CUDA计算前缀和

- 类似树形归并





中国计算机学会
China Computer Federation

如何出题考察并行计算？

- 交互式题目
- 以调用接口次数作为评价标准



小结

- 现代高性能计算，需要考虑数据在硬件里的分布
- 需要考虑并行处理器的执行方式
- 深度学习的成功 = 数据 + 算法 + GPU



OI与应用研究的关系

- 如果你喜欢计算几何：图形学、无人车、游戏
- 如果你喜欢数据结构：数据库、STL
- 如果你喜欢编程语言：编译器、语言设计
- 如果你喜欢模拟题/复杂性：操作系统
- 如果你喜欢数论：密码学、区块链
- 如果你喜欢图论：网络优化
- 如果你喜欢DP：运筹优化
- 如果你喜欢推导定理：计算机理论科学家
- 如果你喜欢随机优化大法：机器学习欢迎你
-还有很多很多



中国计算机学会
China Computer Federation

One last thing

- 计算机 + 艺术 = 人工智能辅助创作
- 计算机 + 制造业 = 新制造
- 计算机 + X = 赋能各行各业



参考文献

- 《Foundations of Data Science》
- UC Berkeley CS270 algorithms and data structures
- UC Berkeley CS267 Applications of Parallel Computers
- 《浅谈随机化思想在几何问题中的应用》
- 《论程序底层优化的一些方法与技巧》
- 《GPU Gems 3》 Chapter39



中国计算机学会
China Computer Federation

Thanks

- 欢迎提问
- Email: jby.yeah@gmail.com

