组合优化与凸优化
第8章 复合优化与智能优化

刘绍辉
计算机科学与技术学院 哈尔滨工业大学
shliu@hit.edu.cn
2023年春季

回顾

- 凸优化问题$min_x f(x), s.t. h_i(x) \le 0, i = 1, \cdots, m, Ax = b$,目标函数和约束函数都是二次可微
  - 构造对数障碍函数(log barrier)：$\phi(x) = -\sum_{(i=1)}^{m} log(-(h_i(x))$
  - 忽略等式约束：$min_x f(x) + \sum_{i=1}^{m} I_{\{h_i(x) \le 0\}}(x)$
  - 使用逼近表示：$\min_x f(x) - \frac{1}{t} \cdot \sum_{i=1}^{m} log\left(-h_i(x)\right), t > 0$是一个大数，$t$越大，近似越精确，$h_i(x) \to 0$,对数障碍函数逼近无穷
  - $\phi(x)$的梯度和Hessian矩阵
    - ✓$\nabla\phi(x) = -\sum_{i=1}^{m} \frac{1}{h_i(x)} \nabla h_i(x)$
    - ✓$\nabla^2\phi(x) = \sum_{i=1}^{m} \frac{1}{h_i(x)^2} \nabla h_i(x) \nabla h_i(x)^T - \sum_{i=1}^{m} \frac{1}{h_i(x)} \nabla^2 h_i(x)$

回顾

◆用之替换不等式

- $\min_x tf(x) + \phi(x), s.t. Ax = b$
- $t > 0$，其KKT条件为：

$$Ax^\star(t) = b, \quad h_i(x^\star(t)) < 0, \quad i = 1, \ldots m$$

$$t\nabla f(x^\star(t)) - \sum_{i=1}^{m} \frac{1}{h_i(x^\star(t))} \nabla h_i(x^\star(t)) + A^T w = 0$$

- $t \to \infty$,期望$x^*(t) \to x^*$

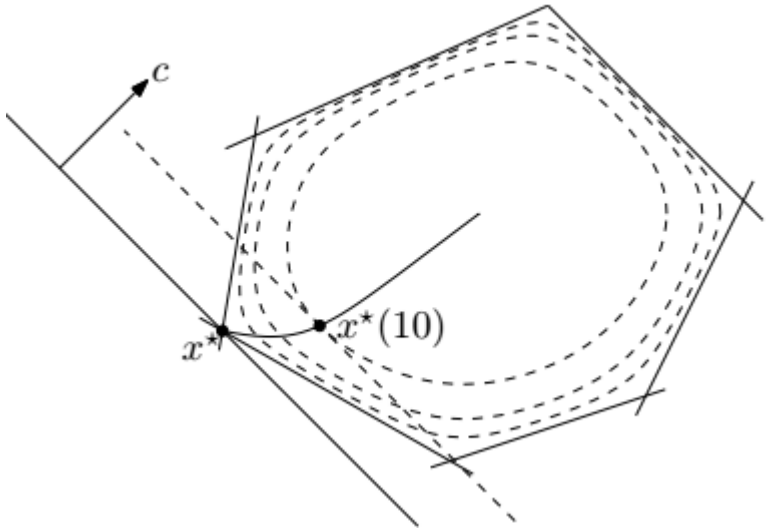
◆例子

- 考虑线性规划问题的障碍函数
  - ✓$min_x tc^T x - \sum_{i=1}^{m} log(e_i - d_i^T x)$
  - ✓障碍函数对应多面体约束：$Dx \leq e$

回顾

◆必要条件为

- $\mathbf{0} = \boldsymbol{tc} - \sum_{i=1}^{m} \frac{1}{e_i - d_i^T x^*(t)} \boldsymbol{d_i}$
- 表明梯度必须与$-\boldsymbol{c}$平行，超平面$\{\boldsymbol{x}: \boldsymbol{c}^T\boldsymbol{x} = \boldsymbol{c}^T\boldsymbol{x}^*(\boldsymbol{t})\}$与$\boldsymbol{\phi}$的轮廓在$\boldsymbol{x}^*(\boldsymbol{t})$相切



复合优化问题

- **一般将目标函数可写为多个函数之和的形式，每个函数可能具有不同的性质**
  - $min_{x \in R^n} \psi(x) = f(x) + h(x), f(x)$可微，$h(x)$不可微
  - 例如正则项可以看做这类问题的一种。
- **如何求解？**
  - 实际上深度学习当中的都是这类问题，因此之前介绍的各种梯度下降法的改进版本，**Adagrad,动量法，Nesterov动量法，RMSProp,AdaDelta,Adam等方法都可以！**
  - 另外前面的约束最优化方法都可以使用，但一般没有充分利用$h(x)$的特点
  - 此外，增广拉格朗日方法，交替方向乘子法，坐标轮换下降法，临近点**PPA**方法等都可以使用

变分不等式(Variational Inequalities)

- 设$\Omega \subset R^n$是非空闭凸集，$F: R^n \to R^n$, 考虑如下变分不等式：$u^* \in \Omega, (u - u^*)^T F(u^*) \geq 0 \ \ (v0), , \forall u \in \Omega$.如果算子映射F满足$(u-v)^T\big(F(u)-F(v)\big) \geq 0$ 则称算子单调。上面的不等式称为单调变分不等式。
- 感觉很抽象？想象一下可行下降方向的表示方法吧！
- $min\{f(x)|x \in \Omega\}$,若函数f可微凸函数，此时为凸规划问题，在迭代求解器最优解时，其迭代方向必须是可行下降方向，必须满足是可行的和下降的条件。如果该点不存在可行下降方向，则表明该点为最优点，
- $x^{k+1} = x^k + \lambda^k d^k$,一般情况方向采用梯度相关的表达，因此方向$d^k \cdot \nabla f\big(x^k\big) < 0$, 且要求$f\big(x^{k+1}\big) - f\big(x^k\big) > 0$,即如果$x^*$点为最优点，必然会有$(x' - x^*)^T \nabla f(x^*) \geq 0, \forall x' \in \Omega$
- 可微凸函数f必定满足如下式子：

$$f(x) \geq f(y) + \nabla f(y)^T (x - y), f(y) \geq f(x) + \nabla f(x)^T (y - x)$$

- 两式相加立即得$(x-y)^T\big(\nabla f(x) - \nabla f(y)\big) \geq 0$,从而凸函数的梯度算子是单调的，也满足上述的单调变分不等式
- 引理(凸函数之和最优性条件)：设$X \subset R^n$ 为闭凸集，函数$\theta(x), f(x)$都是凸函数，且$f(x)$可微，记x^*是凸优化问题$min_{x \in X}\{\theta(x) + f(x)\}$的解，则等价于$x^* \in X, \theta(x) - \theta(x^*) + (x - x^*)^T \nabla f(x^*) \geq 0, \forall\, x \in X$;如果$\theta(x)$也可微，则相应条件可写为$x^* \in X, (x - x^*)^T\big(\nabla\theta(x^*) + \nabla f(x^*)\big) \geq 0, \forall x \in X$

变分不等式(Variational Inequalities)

- **线性约束凸优化问题等价的变分不等式**
- 考虑如下线性约束凸优化问题

$$min_{x\in X}\, f(x), s.t. Ax = b(or \geq b),\text{这里}f: R^n \to R, A \in R^{m\times n}, b \in R^m$$

- 该问题的拉格朗日函数可表示为：$L(x,\lambda) = f(x) - \lambda^T(Ax-b)$,注意如果$Ax = b, L: R^n \times R^m \to R$;如果$Ax \geq b, L: R^n \times R_+^m \to R$;如果$Ax \leq b, L(x,\lambda) = f(x) + \lambda^T(Ax-b)$, 且$L: R^n \times R_+^m \to R$
- 若(x^*,λ^*)满足：$L_{\lambda\in\Lambda}(x^*,\lambda) \leq L(x^*,\lambda^*) \leq L_{x\in X}(x,\lambda^*)$,则称该点为Lagrange函数的鞍点
- 即鞍点必须满足如下方程

$$\begin{cases} x^* = argmin_x\, L(x,\lambda^*) \\ \lambda^* = argmin_\lambda L(x^*,\lambda) \end{cases}$$

- 上述子问题的最优性条件为：

$$\begin{cases} f(x) - f(x^*) + (x-x^*)^T\left(-A^T\lambda^*\right) \geq 0, \forall x \in X \\ (\lambda-\lambda^*)^T(Ax^*-b) \geq 0, \forall \lambda \in \Lambda \end{cases} \quad (v1)$$

- 这个最优性条件可以表示成一个单调的混合变分不等式：

$$f(x) - f(x^*) + (w-w^*)^T F(w^*) \geq 0, \forall w \in \Omega, w^* \in \Omega \quad (v2)$$

变分不等式(Variational Inequalities)

这里$w=\begin{pmatrix}x\\ \lambda\end{pmatrix}, F(w)=\begin{pmatrix}-A^T\lambda\\ Ax-b\end{pmatrix}, \Omega=\mathbf{X}\times\Lambda$ $(v3)$

- 证明：由$(v1)$到$(v2)$显然，直接将$(v1)$中的两式相加即得$(v2)$;反之，$(v2)$中由于w的任意性，令$w=(x,\lambda^*)$和令$w=(x^*,\lambda)$就得到$(v1)$。并称$(v2,v3)$为单调变分不等式,这是因为$f(x)$为凸，且算子F单调。实际上$(w-\tilde{w})^T\big(F(w)-F(\tilde{w})\big)=0$满足单调的定义。
- 可分离结构型凸优化问题等价的变分不等式
- 对如下约束凸优化问题 $min_{x\in X,y\in Y}\ f_1(x)+f_2(y), s.t. Ax+By=b$ $(v3)$
- 通过类似的分析，其最优性条件也可表示为单调的混合变分不等式：

$$w^*\in\Omega, f(u)-f(u^*)+(w-w^*)^TF(w^*)\geq 0, \forall w\in\Omega \quad (v4)$$

$$\text{这里}u=\begin{pmatrix}x\\ y\end{pmatrix}, w=\begin{pmatrix}x\\ y\\ \lambda\end{pmatrix}, F(w)=\begin{pmatrix}-A^T\lambda\\ -B^T\lambda\\ Ax+By-b\end{pmatrix} \quad (v5)$$

- $f(u)=f_1(x)+f_2(y), \Omega=\mathbf{X}\times Y\times R^m$ $(v6)$
- 对三个算子的可分离线性结构型约束凸优化问题：$min_{x\in X,y\in Y,z\in Z}\ f_1(x)+f_2(y)+f_3(z), s.t. Ax+By+Cz=b$ $(v7)$也有类似的结果，其最优性条件可表示成一个单调的混合变分不等式

变分不等式(Variational Inequalities)

$$w^* \in \Omega, f(u) - f(u^*) + (w - w^*)^T F(w^*) \geq 0, \forall w \in \Omega \quad (v8)$$

$$\text{这里} u = \begin{pmatrix} x \\ y \\ z \end{pmatrix}, w = \begin{pmatrix} x \\ y \\ z \\ \lambda \end{pmatrix}, F(w) = \begin{pmatrix} -A^T\lambda \\ -B^T\lambda \\ -C^T\lambda \\ Ax+By+Cz-b \end{pmatrix} \quad (v9)$$

$$f(u) = f_1(x) + f_2(y) + f_3(z), \Omega = X \times Y \times Z \times R^m \ (v10)$$

- 注意，上述两个算子可分离结构型约束优化问题，可以用相当有效的交替方向乘子法(ADMM)求解(R1)，而直接将ADMM推广到求解三个算子可分离结构约束凸优化问题$(v8 - v10)$,已有结果(R2)表明这不一定收敛，因此，对多于两个算子的可分离约束凸最优化问题，需要用一些基于变分不等式的预测校正方法去处理(R3,R4)。
- 这些变分不等式本质上与$(v0)$没有区别，因此，可以用针对$(v0)$收缩算法的成果应用到这些变分不等式上。

R1. R. Glowinski, Numerical Methods for Nonlinear Variational Problems, Springer-Verlag, New York, Berlin, Heidelberg, Tokyo, 1984.

R2. C. H. Chen, B. S. He, Y. Y. Ye and X. M. Yuan, The direct extension of ADMM for multi-block convex minimization problems is not necessarily convergent, Online published in Mathematical Programming, Series A

R3. B. S. He, M. Tao and X.M. Yuan, Alternating direction method with Gaussian back substitution for separable convex programming, SIAM Journal on Optimization 22, 313-340, 2012.

R4. B.S. He, M. Tao and X.M. Yuan, A splitting method for separable convex programming, IMA Journal of Numerical Analysis, 31, 394-426, 2015.

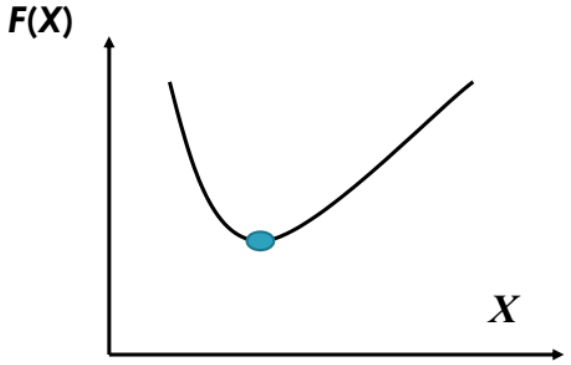
Proximal Point Algorithms(PPA)算法

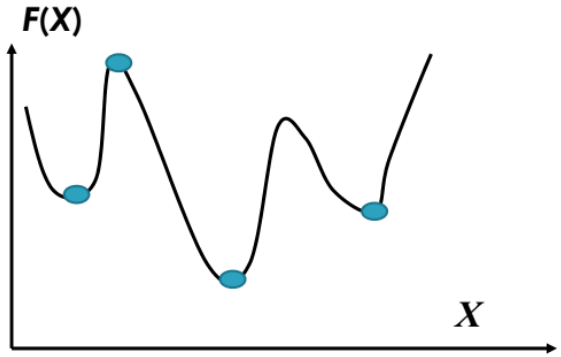
- 邻近点算法是求解凸优化和单调变分不等式的一类基本算法。
- 简单凸优化的邻临近点算法
- 问题$min_{x\in X}\ f(x), f: R^n \to R$为凸函数，$X$为闭凸集 的PPA算法如下：
- 对给定的$r>0, x^k$,求得$\bar{x}^k = Argmin_{x\in X}\left\{f(x)+\frac{r}{2}||x-x^k||^2\right\}\quad (p1)$
- 称$\bar{x}^k$是k-次迭代的邻近点

8.0 启发式算法

◆优化方法分类

- ➢全局vs局部
- ➢单目标vs多目标
- ➢无约束vs有约束
- ➢基于梯度vs无梯度





◆梯度方法的限制

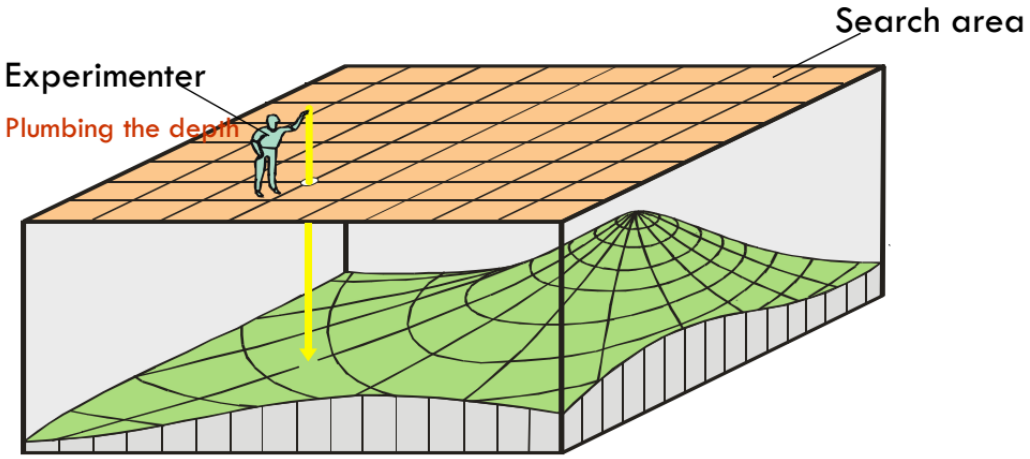
- ➢目标函数是单峰或单谷函数
  - ✓否则，容易陷入局部极小点

◆典型的无梯度方法

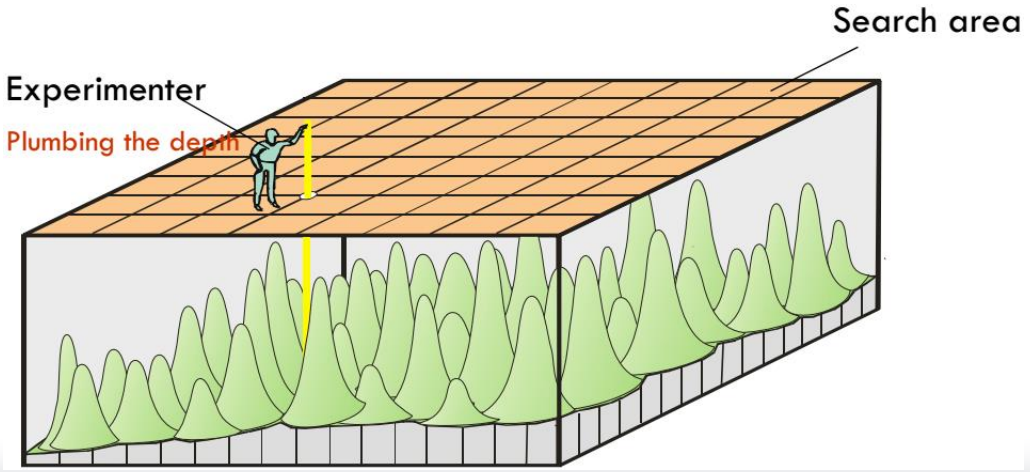
- ➢启发式方法（Heuristic methods):black box 方法
  - ✓解复杂的多变量组合优化问题（大规模、非线性、非凸）

8.0 启发式算法

◆目标函数是光滑函数



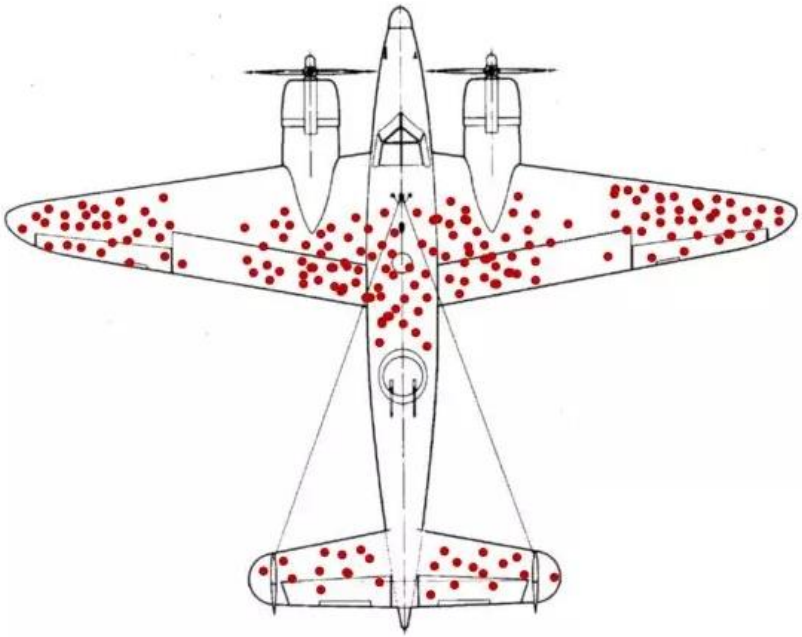
◆目标函数非光滑



8.0 启发式算法-优化求解

◆防止幸存者偏差(survivorship bias)

- 样本不合理，基于数据的建模往往会依赖于数据本身，因而易造成这个问题



- 深度学习基本都是基于数据驱动的求解方法，是否会存在这个问题？要如何避免呢？

8.0 启发式算法 *"If something can go wrong, it will."*

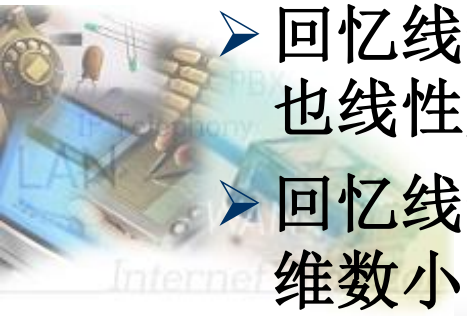
- “墨菲定律”
  - 原话是这样说的：If there are two or more ways to do something, and one of those ways can result in a catastrophe, then someone will do it.（如果有两种或两种以上的方式去做某件事情，而其中一种选择方式将导致灾难，则必定有人会作出这种选择。）
  - 概率优化算法，启发式优化算法
    - ✓有概率跳出局部最优点，则一定会有这种可能！
    - ✓当然，你得尝试的次数足够多！
  - 软件测试
    - ✓充分的测试，一旦有BUG，迟早有人会碰到！
- 优化问题如果是NP-hard的问题
  - 必须采用启发式求解,一般称为近似算法

组合优化问题(Combinatorial Optimization)

- 无论如何，组合优化问题一般都以线性规划、整数规划为基础
- 常见的经典组合优化问题
  - 划分问题(Partitioning);调度问题(Scheduling);装箱问题（Bin Packing);背包问题（Knapsack）;指派问题（Assignment）;旅行商问题（Travelling Salesman Problem);斯坦纳最小树问题(Steiner Minimal Tree);网络中的组合优化问题、最小生成树(Minimal Spanning Tree);最大流问题（Max-flow Problem);最小费用流问题(Min-cost flow Problem)
- 组合优化
  - 实例集合中的每个实例I,有一个有穷的可行解集合$S(I)$,目标函数$f(I,\sigma),\sigma \in S(I)$:最优解$\sigma^*$: $f(I,\sigma^*) \geq f(I,\sigma),\forall\sigma \in S(I)$,或者$f(I,\sigma^*) \leq f(I,\sigma),\forall\sigma \in S(I)$,

组合优化问题

- **许多组合优化问题可以用拟阵(Matroid)来进行抽象的形式化表达**
  - 给定系统$(E,\mathcal{F})$,其中E是有限集合，$\mathcal{F}$是集合E上的幂集合，也就是$\mathcal{F} \subseteq 2^E$,代价函数$c: \mathcal{F} \to R$,寻找代价最小或最大的幂集中的元素
  - 模函数c可定义为：$c(X) = c(\emptyset) + \sum_{x \in X}(c(\{x\} - c(\emptyset)), \forall X \subseteq E$,或者等价于：$c: E \to R, c(X) = \sum_{e \in X} c(e)$
- **其中$\mathcal{F}$一般描述了一个独立系统或者拟阵，这与线性代数中的线性无关性类似**
  - 回忆线性规划中的基向量，线性无关，则任何基向量的子集列也线性无关
  - 回忆线性规划中的基向量不唯一，但等价；如果一个基向量的维数小于列向量空间的维数，则必定可以从一个基向量空间中取出一个向量加入来形成完整的基向量



组合优化问题-独立系统

- 一个集合系统$(E, \mathcal{F})$称为独立系统：
  - 空集$\emptyset \in \mathcal{F}$;
  - 继承性：$X \subseteq Y \in \mathcal{F} \Rightarrow X \in \mathcal{F}$
- 集合$\mathcal{F}$的元素称为独立的，而$2^E \backslash \mathcal{F}$ 的元素称为相关的
  - 最小相关集合称为环(circuits)
  - 最大独立集合称为基(bases)
  - X $\subseteq E$,X的最大独立子集称为X的基
- 若$(E, \mathcal{F})$为独立系统，则定义$X \subseteq E$的秩$r(X) = max\{|Y|: Y \subseteq X, Y \in \mathcal{F}\}$;定义$X$的闭包$\sigma(X) = \{y \in E: r(X \cup \{y\}) = r(X)\}$

组合优化问题-独立系统

- 此时很多组合优化问题可形式化为如下两个问题
  - 独立系统的最大化问题：
    - ✓实例：$(E, \mathcal{F}), c: E \to R$
    - ✓任务：找到$X \in \mathcal{F}$,使得$c(X) = \sum_{(e \in X)} c(e)$最小
  - 独立系统的最小化问题：
    - ✓实例：$(E, \mathcal{F}), c: E \to R$
    - ✓任务：找到基B,使得$c(B)$最小
- 例如：最大加权稳定集问题

组合优化问题-独立系统

◆TSP问题

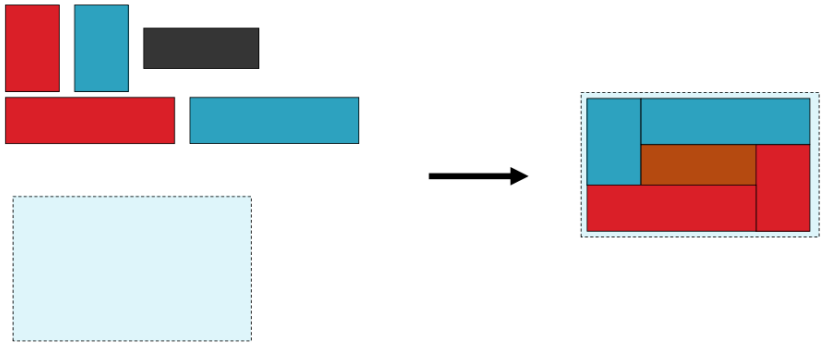
- 给定完全无向图G和权重$c: E(G) \to R^+$,在G中找一个最小权Hamiltonian环
- 这里：$E = E(G), \mathcal{F} = \{F \subseteq E: F$是$G$中汉密尔顿环的边的子集$\}$

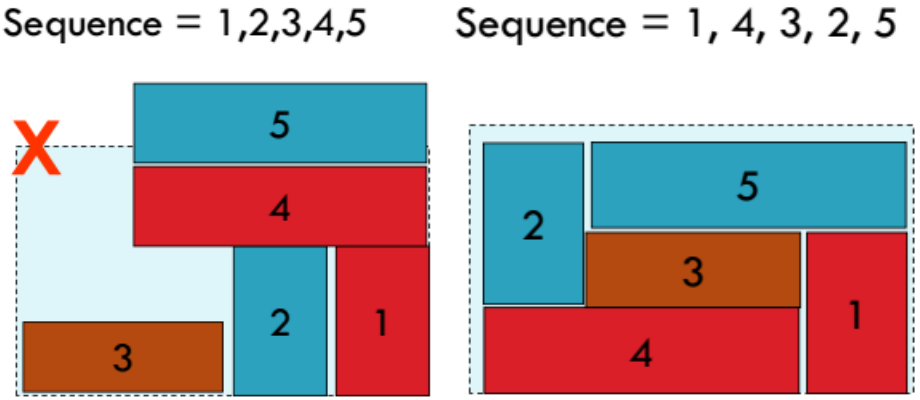
◆最短路径问题

- 给定图G(有向或无向),$c: E(G) \to R, s.t \in V(G)$使得$t$从$s$可达，在$G$中找关于$c$的最短$s-t$路径
- 这里：$E = E(G), \mathcal{F} = \{F \subseteq E: F$是一个$s-t$路径的边的子集$\}$

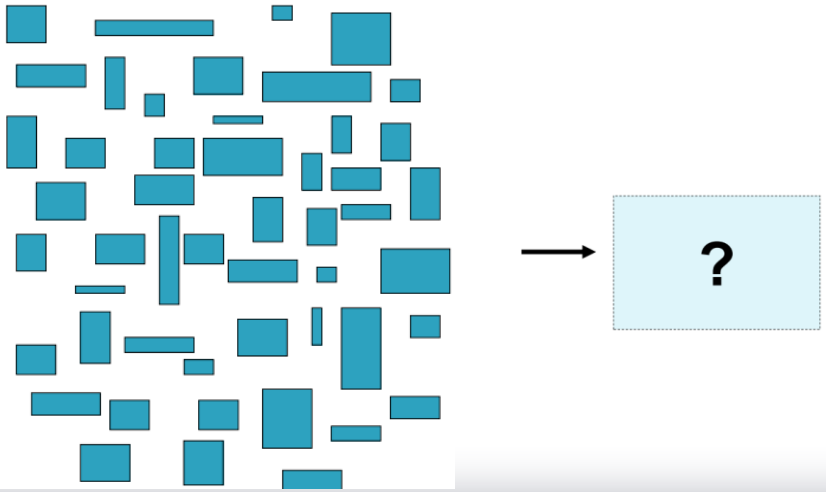
8.0 启发式算法

◆平面装箱问题





◆扩展到更多的块



50片：
50！
=30414093201713378043612608166064768844377641568960512000000000000
计算机每秒评估1000次，需要
96442456880115988215412887386050129516671872047693150 6年

8.0 启发式算法

◆最优化算法

- ➢求解该问题的每个实例的解
- ➢尤其是线性规划，凸规划问题
- ➢但多次强调，需要在精度和速度之间做均衡！

◆启发式算法（heuristic）

- ➢一种技术，使得在可接受的计算费用内寻找到最好的解，但不一定能保证所得解的可行性和最优性
- ➢甚至在多数情况下，无法阐述所得解同最优解的近似程度
- ➢例如TSP问题（旅行商问题）

城市数	24	25	26	27	28	29	30	31
时间	1s	24s	10m	4.3h	4.9d	136.5d	10.8y	325y

8.0 启发式算法

◆启发式算法的特点

- 不考虑算法所得解与最优解的偏离程度
- 若采用分析最坏情况下的误差界限来评价启发式算法

◆近似算法

- 设A是一个问题。记问题A的任何一个实例I的最优解和启发式算法H解的目标值分别为$z_{OPT}(I)$和$z_H(I)$。于是对于某个$\epsilon \geq 0$，称H是A的ϵ近似算法（$\epsilon-$ app$\boldsymbol{roximation\ algorithm}$）当且仅当
- $|z_H(I) - z_{OPT}(I)| \leq \epsilon |z_{OPT}(I)|, \forall I \in A$
- 启发式算法与近似算法的区别
  - ✓启发式算法集合包含近似算法集合
  - ✓近似算法强调给出算法最坏情况的误差界限，启发式算法不考虑
  - ✓最坏情况误差界估计需要较好的数学和技巧，即使这样也很难，只能通过启发式算法解决

8.0 启发式算法

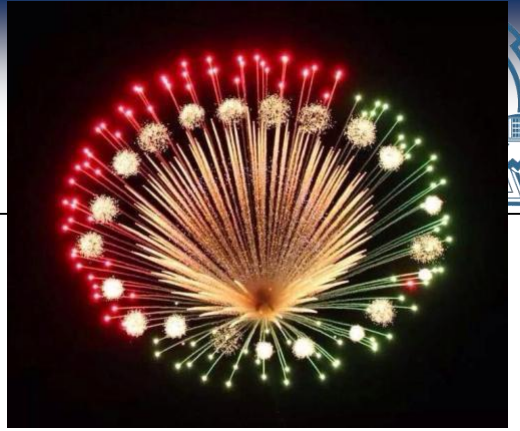
- 启发式算法如背包问题的贪婪算法（greedy）
- 近年来得到快速发展
  - 数学模型是实际问题的简化
    - ✓ 忽略因素、数据不精确、参数估计不准确，从而使得最优化算法比启发式算法的解更不准确
  - 有些难的组合优化问题可能还没有找到最优算法，即使存在，由算法复杂性理论，其计算时间也无法接受，超出实际计算能力
  - 一些启发式算法可以用在最优算法中，如分支定界算法中可以用启发式算法估界
  - 简单易行；直观易被接受
  - 速度快，适合实时应用
  - 多数情况下，程序简单，易于修改
- 缺点
  - 不能保证求得最优解
  - 表现不稳定
  - 算法好坏依赖于实际问题，不同算法之间难以比较

概率算法

墨菲定律和幸存者偏差

Murphy's Law

8.0 启发式算法



- **现代优化算法：大都具有一定的随机性**
  - 禁忌搜索(tabu search)
  - 模拟退火（Simulated annealing）
  - 遗传算法（genetic algorithms）:在图像处理、信号处理、模式识别等领域有着广泛的应用
  - 人工神经网络（neural networks）
  - 粒子群优化算法（particle swarm optimization）一种模仿鸟群、鱼群觅食行为发展起来的一种进化算法
  - 差分进化算法（Differential Evolution）:群体进化,模拟了群体中的个体的合作与竞争的过程
  - 人工蜂群算法（Artificial Bee Colony Algorithm，ABC）
  - 蚁群优化算法（Ant Colony Optimization）
  - 人工鱼群优化算法（Artificial Fish Swarm Algorithm）
  - 菌群优化算法Bacterial Foraging Optimization
  - 杜鹃搜索算法（Cuckoo search，CS）模仿杜鹃鸟寻窝产卵活动的群集智能优化算法
  - 群搜索算法（Group Search Optimizer）是一种基于发现者，跟随者，游荡者模型而产生的算法
  - 灰狼算法（Grey Wolf Optimizer）
  - 萤火虫算法(Firefly Algorithm)是一种模仿萤火虫之间信息交流，相互吸引集合，警戒危险
  - 烟花优化算法（Fireworks Algorithm）
  - 鲸鱼算法（Whale Optimization Algorithm）
  - 水波算法（Water wave optimization）是根据水波理论提出的优化算法。什么是水波理论？简单来说就是水波的宽度越小，其频率越高，频率与水波宽度的平方根成反比

8.0 启发式算法

◆基本起源

算法	由来	类别
1遗传算法	进化论	人类理论
2粒子群优化算法	鸟群觅食	生物生存
3差分进化算法	进化论	人类理论
4人工蜂群算法	蜜蜂觅食	生物生存
5蚁群算法	蚂蚁觅食	生物生存
6人工鱼群算法	鱼群觅食	生物生存
7杜鹃搜索算法	杜鹃产卵	生物生存
8萤火虫算法	萤火虫求偶	生物生存
9灰狼算法	狼群觅食	生物生存
10鲸鱼算法	鲸鱼觅食	生物生存
11群搜索算法	生产消费跟随模型	人类理论
12混合蛙跳算法	青蛙觅食	生物生存
13烟花算法	烟花	物理现象
14菌群优化算法	菌群生长	生物生存

8.0 启发式算法

◆更新策略

算法	类别
1遗传算法	不跟随最优解
2粒子群优化算法	跟随最优解
3差分进化算法	不跟随最优解
4人工蜂群算法	跟随最优解
5蚁群算法	跟随最优解
6人工鱼群算法	跟随最优解
7杜鹃搜索算法	跟随最优解
8萤火虫算法	跟随最优解
9灰狼算法	跟随最优解
10鲸鱼算法	跟随最优解
11群搜索算法	跟随最优解
12混合蛙跳算法	跟随最优解
13烟花算法	跟随最优解
14菌群优化算法	跟随最优解

8.0 启发式算法

◆烟花算法（Fireworks Algorithm：FWA）

- 每一个烟花的位置都代表一个可行解
- 爆炸产生：正常的火星与特别的火星。
- 每个火星都会爆炸产生数个正常火星，某些火星有一定的概率产生一个特别的火星。
  - ✓正常的火星根据当前火星的振幅随机均匀分布在该火星的周围，而特别的火星将在当前火星附近以正态分布方式产生。
  - ✓每次迭代产生的火星数量多于每一代应有的火星数，算法将参照火星位置的优劣，随机留下指定数量的火星，以保持火星数目的稳定。

8.0 启发式算法

◆烟花算法

- 火星：位置X以及振幅A，
  - ✓振幅与所有火星的亮度值有关，亮度越高的火星的振幅越小，反之振幅越大。（这里亮度表示火星的适应度值，越亮表示这个位置的氧气浓度越高，适应度越好）

适应度值f越小越优的情况：

$$A_i = A_{max}\frac{f(X_i) - f_{min} + \xi}{\sum_{k=1}^{N}(f(X_i) - f_{min}) + \xi}, f_{min} = f_{best}$$

适应度值f越大越优的情况：

$$A_i = A_{max}\frac{f_{max} - f(X_i) + \xi}{\sum_{k=1}^{N}(f_{max} - f(X_i)) + \xi}, f_{max} = f_{best}$$

产生正常火星的数量：

$$S_i = S_{max}\frac{f(X_i) - f_{min} + \xi}{\sum_{k=1}^{N}(f(X_i) - f_{min}) + \xi}, f_{max} = f_{best}$$

$$S_i = \begin{cases} round(a * S_{all}), S_i < a * S_{all} \\ round(b * S_{all}), S_i > b * S_{all}, a < b < 1 \\ round(S_i), a * S_{all} \le S_i \le b * S_{all}, a < b < 1 \end{cases}$$

$$S_i = S_{max}\frac{f_{max} - f(X_i) + \xi}{\sum_{k=1}^{N}(f_{max} - f(X_i)) + \xi}, f_{min} = f_{best}$$

8.0 启发式算法

- **产生正常火星**
  - z为随机正整数

$$X_{i,j}^{k+1}=\begin{cases}A_i * rand(-1,1)+X_{i,j}^{k}, j\in\{d_1,d_2,\ldots,d_z\}, 1\le z\le d\\ X_{i,j}^{k}, j\notin\{d_1,d_2,\ldots,d_z\}\end{cases}$$

- **产生特别火星**

$$X_{i,j}^{k+1}=\begin{cases}randGauss(1,1) * X_{i,j}^{k}, j\in\{d_1,d_2,\ldots,d_z\}, 1\le z\le d\\ X_{i,j}^{k}, j\notin\{d_1,d_2,\ldots,d_z\}\end{cases}$$

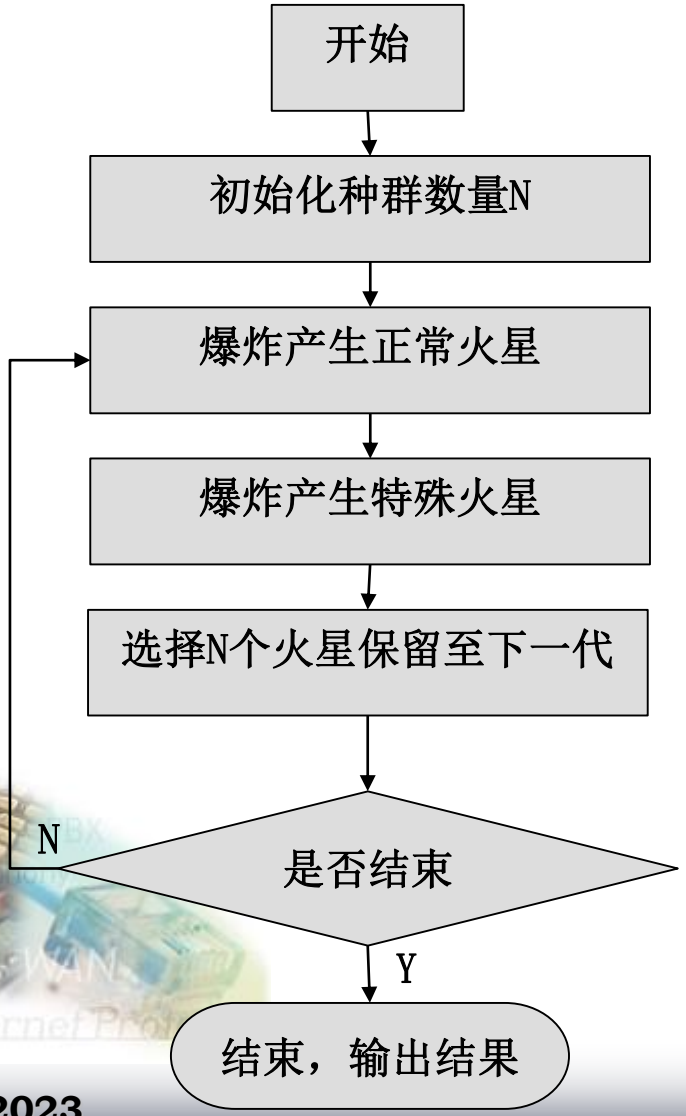
- **火星保留到下一代**
  - N个火星：产生S_{all}个正常火星以及m个特别的火星，但每一代中只能从这$N+S_{all}+m$个火星中选择N个火星保留至下一代
  - 每次从中选择最优的火星保留至下一代

$$p(X_i)=\frac{R(X_i)}{\sum_{k=1}^{S_{all}+m+N}R(X_k)}\qquad R(X_i)=\sum_j R(X_j)$$

  - $R(X)$表示该火星距其他所有火星的距离之和，即距其它火星越远的火星，被选择保留至下一代的概率较大

8.0 启发式算法

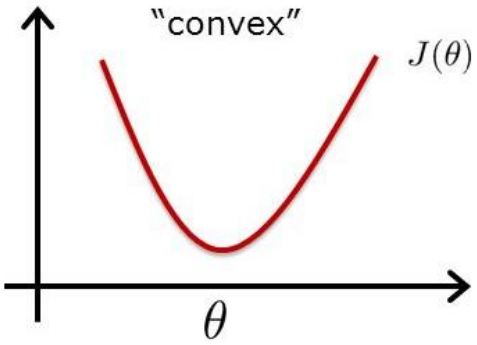
◆烟花算法

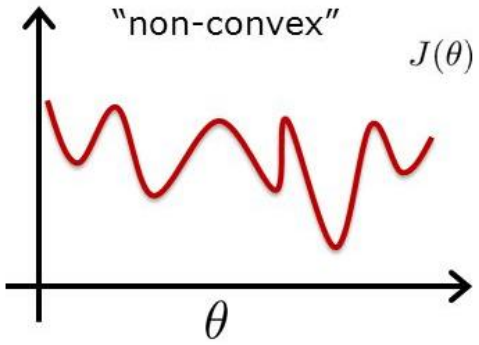


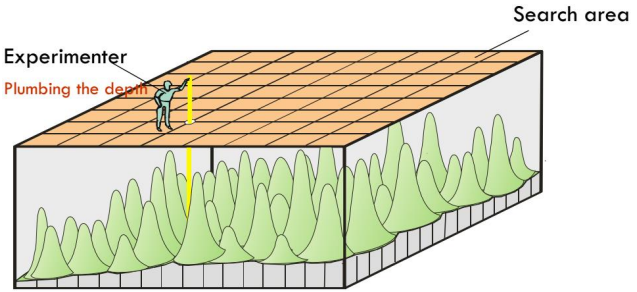
8.2人工神经网络与优化

◆深度学习中的大多数优化问题都是非凸的

- ➢机器学习也类似
- ➢甚至有可能是NP-Hard的







- ➢可控梯度下降 $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \nabla f(\boldsymbol{\theta}_t)$
  - ✓$\nabla \neq 0$ ➔ ∃ descent direction
  - ✓ 如果Hessian矩阵 (∇^2)大, 允许梯度∇ 有较大的波动!
  - ✓为确保下降，根据平滑性选取小步 $\nabla^2 f(\boldsymbol{\theta}) \preccurlyeq \beta I$

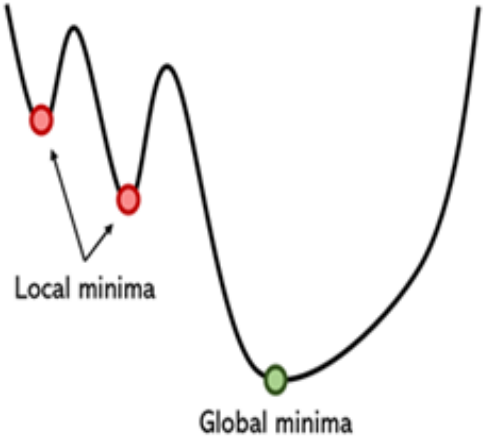
8.3 随机神经网络

◆BP网络

- 可能陷入局部最优？
- **梯度下降：**算法赋子网络的是只会“下山”而不会“爬山”的能力

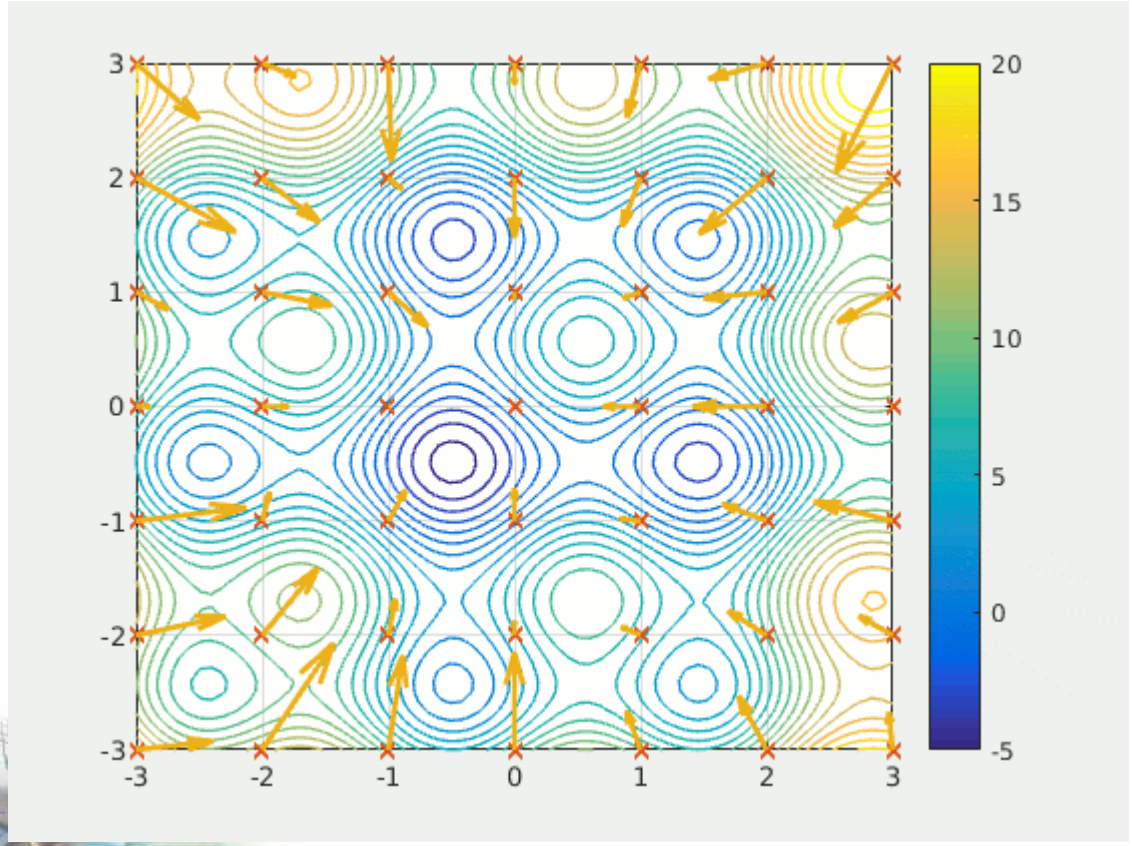
- 由于优化器深受局部极小值驱使，即便它极力摆脱后者，也要用很长时间。因收敛速度慢，梯度下降法通常十分冗长，即使是适应像批量梯度下降这样的大数据集之后也是如此。
- 学习率决定优化器的可靠程度和风险度。学习率设置过高可能会导致优化器忽略全局最小值，而过低则会导致运行时崩溃。解决此问题需要设置随着衰减而变化的学习率，但是在决定学习率的许多其他变量中选择衰减率很困难
- 梯度下降对优化器的初始化尤为敏感。例如，优化器在第二个局部最小值而非第一个局部最小值附近进行初始化，则优化器的性能可能会更优，但这都是随机决定的。
- 梯度下降需要梯度，这意味着除了无法处理不可微函数之外，还容易出现诸如梯度消失或梯度爆炸等梯度问题。

损失函数



8.3 随机神经网络

◆PSO粒子群优化算法找极值点



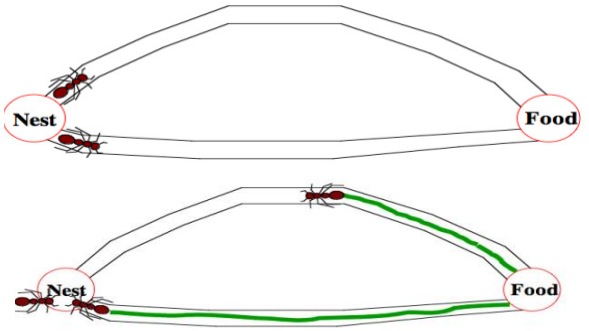
PSO通过观察和模仿其邻居节点的情况来改进搜索

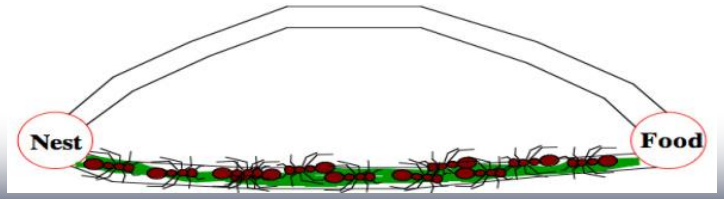
每个PSO个体都具有记忆能力.

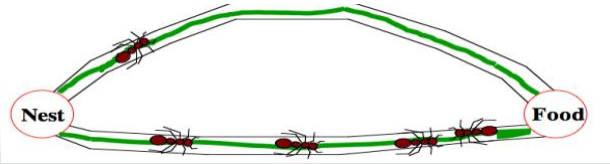
PSO具有简单算法和低开销.

使其在某些环境中比其它算法更流行

其计算只与速度有关







8.3随机神经网络

◆有没有方法改进这种情况？

- (1)在学习阶段，随机网络不像其他网络那样基于某种确定性算法调整权值，而是按某种概率分布进行修改。
- (2)在运行阶段，随机网络不是按某种确定性的网络方程进行状态演变，而是按某种概率分布决定其状态的转移。神经元的净输入不能决定其状态取1还是取0，但能决定其状态取1还是取0的概率

◆避免陷入局部极小点

- 模拟退火算法是随机网络中解决能量局部极小问题的一个有效方法
- 模拟金属退火过程，先将物体加热至高温，使其原子处于高速运动状态，此时物体具有较高的内能；然后，缓慢降温，随着温度的下降，原子运动速度减慢，内能下降；最后，整个物体达到内能最低的状态。这个状态就是最优解

8.3随机神经网络

◆在随机网络学习过程

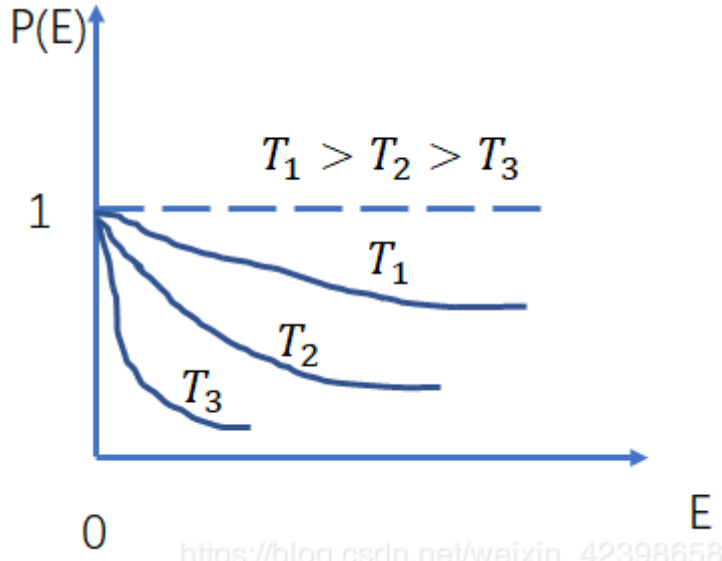
- ➢网络权值作随机变化，
- ➢计算变化后的网络能量函数
  - ✓网络权值的修改准则
  - ✓若权值变化后能量变小，则接受这种变化；
  - ✓否则按预先选定的概率分布接受权值的这种变化
  - ✓从而赋予网络一定的“爬山”能力
  - ✓**Metropol**等提出的模拟退火算法

◆ 设X代表某一物质体系的微观状态（一组状态变量，如粒子的速度和位置等），E(X)表示该物质在某微观状态下的内能，对于给定温度T，如果体系处于热平衡状态，则在降温退火过程中，其处于某能量状态的概率与温度的关系遵循Boltzmann分布规律。

8.3随机神经网络

- 分布函数为

$$P(E)\propto e^{[-\frac{E(X)}{KT}]}$$



- 其中K为玻尔兹曼常数
- 温度一定，能量越高，概率越低
- 物质趋向于向能量低的地方演变
- 由此可见，温度参数T越高，状态越容易变化“为了使物质体系最终收敛到低温下的平衡态，应在退火开始时设置较高的温度，然后逐渐降温，最后物质体系将以相当高的概率收敛到最低能量状态。
- 用随机神经网络解决优化问题时，通过数学算法模拟了以上退火过程

8.3随机神经网络

◆模拟退火

- 固体加热后再慢慢冷却，固体在压力下加热，其内部能量增加且分子随机运动。随后，固体慢慢冷却，热能一般会慢慢减少，但有时也以Boltzmann概率随机增加，即在温度t,能量增加幅度ΔE的概率密度为$e^{-\frac{\Delta E}{kt}}$,其中k是Boltzmann常数。如果冷却相当慢且降温足够大，则最终状态是无压力下的，且所有分子都按最小势能形式排列
- 如何与优化问题对应？
  - ✓在$x \in \mathcal{X}, minf(x)$的优化问题，用上述物理冷却过程来求解一个组合优化问题。对模拟退火算法，x对应材料的状态，$f(x)$对应其能量水平，最优解x^*对应具有最小能量的材料状态。
  - ✓当前状态间的随机转换，即由$x^{(k)} \rightarrow x^{(k+1)}$的移动由上述的Boltzmann分布决定，此分布依赖温度参数t
  - ✓温度高，更可能接受上坡移动，即向更高能的状态移动，则阻止算法收敛到已经找到的第一个局部最小值；随着温度降低，逐渐收敛到整体最小值

8.3随机神经网络

- **模拟退火**
  - $k=0$的初值为$x^{(0)}$,温度为t_0,k表示迭代变量，此算法在几个阶段内运行，阶段标号用$j=0,1,2,\cdots$表示，每个阶段含有多步迭代，第j个阶段的长度为m_j，每次迭代如下：
  - 1.在$x^{(k)}$的邻域内$N(x^{(k)})$，根据提案密度$g^{(k)}(\cdot|x^{(k)})$选取候选解x^*;
  - 2.随机决定是否采用x^*作为下一个候选解或还是仍用当前解，特别地，以概率：

$$min\left(1,e^{\left\{\frac{\left[f(x^{(k)})-f(x^*)\right]}{t_j}\right\}}\right)$$取$x^{(k+1)}=x^*$,否则令$x^{(k+1)}=x^{(k)}$

  - 3.重复第1，2步m_j次
  - 4.增加j且更新$t_j=\alpha(t_{j-1}),m_j=\beta(m_{j-1})$,转至第1步
- **如果根据总迭代次数的限制或事先给定的t_j,m_j，算法不停止，则可以用绝对或相对收敛准则来控制。一般停止准则由最小温度来表示，停止后的最有候选解就是估计的最小值**
- **此外，函数$\alpha(\cdot)$应使温度慢慢递减至0，每个温度m_j中的迭代次数应较大且关于j单增，理想的$\beta(\cdot)$应使m_j为p的指数**

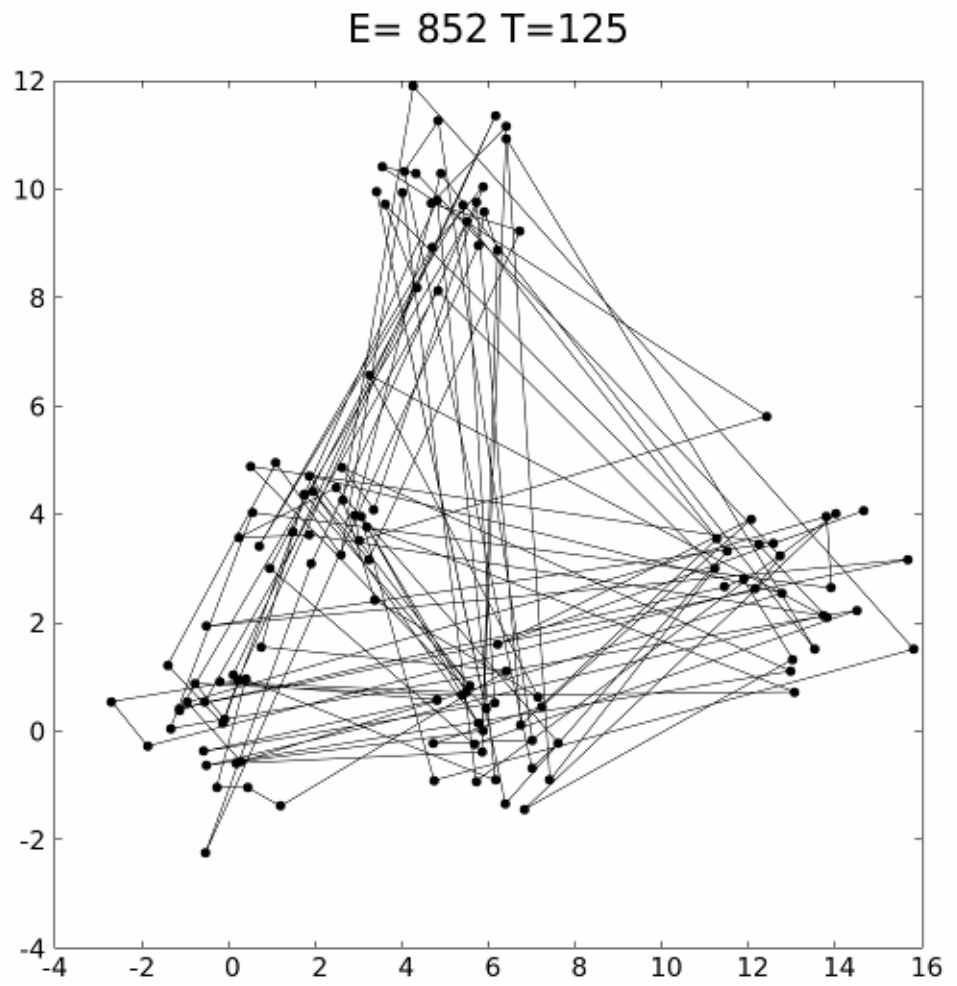
随机下降算法：逃脱局部最优解

8.3随机神经网络

◆模拟退火解旅行商问题

➢如何定义旅行商问题？

标号1-n，任意一次旅行设为x表示其一个排列，考虑如何生成x的一个邻域？2邻域有$\frac{n(n-3)}{2}$个，比完全解空间$\frac{(n-1)!}{2}$个旅行要小很多！



8.3随机神经网络

◆模拟

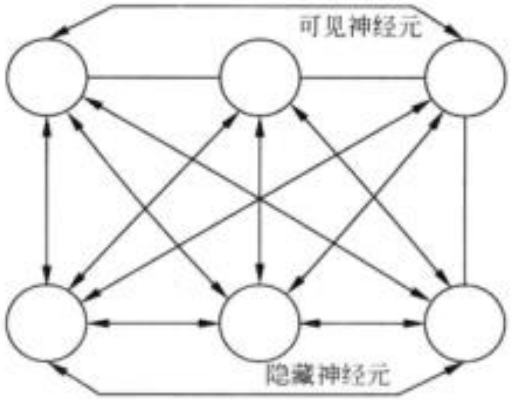
- 定义一个网络温度以模仿物质的退火温度，取网络能量为欲优化的目标函数。网络运行开始时温度较高，调整权值时允许目标数偶尔向增大的方向变化，以使网络能跳出那些能量的局部极小点。随着网络温度不断下降至0，最终以概率1稳定在其能量数的全局最小点，从而获得最优解

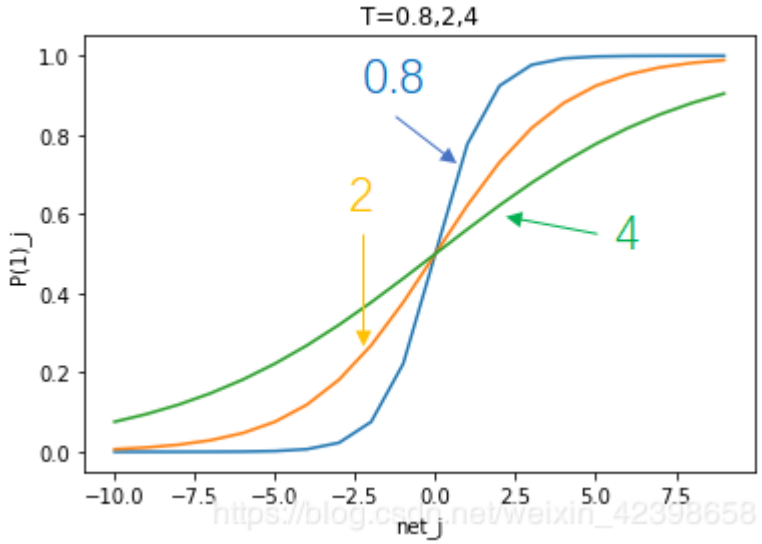
◆玻尔兹曼机-随机神经网络

- **G.E.Hinton等于1983~1986，神经元只有两种输出状态，即单极性二进制的0或1。状态的取值根据概率统计法则决定，由于这种概率统计法则的表达形式与著名统计力学家Boltzmann提出的Boltzmann分布类似，故将这种网络取名BoltzmannMachine(BM).**

8.3随机神经网络

- **玻尔兹曼机**
  - 结构如图所示





- **节点输入：$net_j = \sum_i^n (w_{ij}x_i - T_j)$**
- **实际输出按一定概率计算，如，节点输出为1的概率表示为：$P_j(1) = \frac{1}{1+e^{-\frac{net_j}{T}}}$，输出为0的概率表示为$1 - P_j(1)$**
- **网络能量函数E**
- $E(t) = -\frac{1}{2}x^T(t)wx(t) + x^T(t)T = -\frac{1}{2}\sum_{j=1}^n \sum_{i=1}^n w_{ij}x_i x_j + \sum_{i=1}^n T_i\, x_i$
- BM网络按异步方式工作，每次第j个神经元改变状态

$$\Delta E(t) = -\Delta x_j(t) net_j(t)$$

- ***分析，$net_j > 0$和$net_j < 0$都会发生什么情况？***

BM网络具有从局部极小的低谷中跳出的“爬山”能力？

8.3随机神经网络

- 网络在运行过程中不断地搜索更低的能量极小值，直到达到能量的全局最小。从模拟退火法的原理可以看出，温度T不断下降可使网络能量的“爬山”能力由强减弱，这正是保证BM网络能成功搜索到能量全局最小的有效措施
- BM网络的玻尔兹曼分布
- 设$x_j=1, E_1$，$x\_j=0$，E_0，$x_j: 1\to 0$，$\Delta x_j=-1$，则$\Delta E=E_0-E_1=-(-1)net_j=net_j$
  - $P_j(1)=\frac{1}{1+e^{-\frac{net_j}{T}}}=\frac{1}{1+e^{-\frac{\Delta E}{T}}}$，$P\_j(0)=1-P\_j(1)$
  - $\frac{P_j(1)}{P_j(0)}=\frac{e^{-\frac{E_0}{T}}}{e^{-\frac{E_1}{T}}}$，类似的，$x_j: 0\to 1$也有类似的公式，将其推广
- **玻尔兹曼分布**
  - 网络中任意两个状态出现的概率与对应能量之间的关系：$\frac{P(\alpha)}{P(\beta)}=\frac{e^{-\frac{E_\alpha}{T}}}{e^{-\frac{E_\beta}{T}}}$

8.3随机神经网络

◆玻尔兹曼分布

- BM网络处于某一状态的概率主要取决于此状态下的能量E，能量越低，概率越大；
- BM网络处于某一状态的概率还取决于温度参数T,温度越高，不同状态出现的概率越接近，网络能量较易跳出局部极小而搜索全局最小；温度低则情况相反。这正是采用模拟退火方法搜索全局最小的原因所在。

$$\frac{P(\alpha)}{P(\beta)}=\frac{e^{-\frac{E_\alpha}{T}}}{e^{-\frac{E_\beta}{T}}}$$

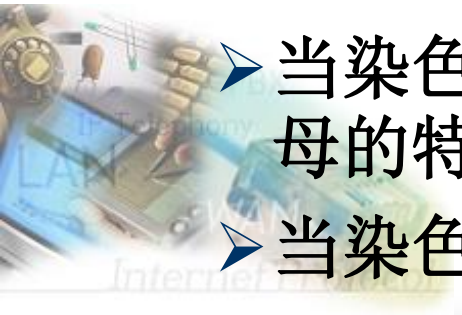
8.4 进化算法和遗传算法

◆借助生物学中的适者生存的规律

- 优胜劣汰

◆遗传算法的主要特点

- 进化发生在解的编码上，这些编码按生物学上的术语称为染色体。由于对解进行了编码，优化问题的一切性质都通过编码来研究。遗传算法涉及编码和解码
- 自然选择规律决定哪些染色体产生超过平均数的后代。遗传算法中通过优化问题的目标人为地构造适应函数以达到好的染色体产生超过平均数的后代
- 当染色体结合时，双亲的遗传基因的结合使得子女保持父母的特征
- 当染色体结合后，随机的变异会造成子代同父代的不同



8.4 进化算法和遗传算法

◆用遗传算法求解$f(x)=x^2, 0\le x\le 31, x$为整数的最大值

- 简单表示解的编码采用二进制表示，采用5位二进制表达
- 染色体：10000:16,11111:31, 01001:9,00010:2
- 每个分量称为基因，每个基因两种状态0和1.
- 模拟生物进化，首先产生群体，随机取四个染色体组成群体：$x_1=(00000), x_2=(11001), x_3=(01111), x_4=(01000)$,群体有4个个体
- 适应函数根据目标函数确定：$fitness(x)=f(x)=x^2$,因此：$f(x_1)=0, f(x_2)=25^2, f(x_3)=15^2, f(x_4)=8^2$
- 定义第i个个体入选种群的概率：
- $p(x_i)=\frac{fitness(x_i)}{\sum_i(fitness(x_i))}$,显然，适应函数值大的个体生存概率大
- 若群体中选四个个体称为种群，最可能选上的是$x_2=(11001)$，x_3，x_4
- *若他们结合，采用如下的交叉方式，则*
- $x_2=(11001)\rightarrow y_1=(11111)$
- $x_3=(01111)\rightarrow y_2=(01001)$

$x_2=(11001)\rightarrow y_3=(11000)$

$x_4=(01000)\rightarrow y_4=(01001)$

8.4 进化算法和遗传算法

- 用遗传算法求解$f(x)=x^2, 0\le x\le 31, x$为整数的最大值
  - $x_2=(11001)\rightarrow y_1=(11111)$
  - $x_3=(01111)\rightarrow y_2=(01001)$
  - 若y_4第一个基因发生变异，$y_4=(11001)$

$x_2=(11001)\rightarrow y_3=(11000)$
$x_4=(01000)\rightarrow y_4=(01001)$

- 遗传算法
  - Step1.选择问题的一个编码；给出一个有N个染色体的初始群体$pop(1), t:=1;$
  - Step2.对群体$pop(t)$中的每个染色体$pop_i(t)$计算它的适应函数

$$f_i=fitness(pop_i(t))$$

  - Step3.若停止准则满足，则算法终止；否则，计算概率

$$p_i=\frac{f_i}{\sum_j f_j}, i=1,2,\cdots,N$$

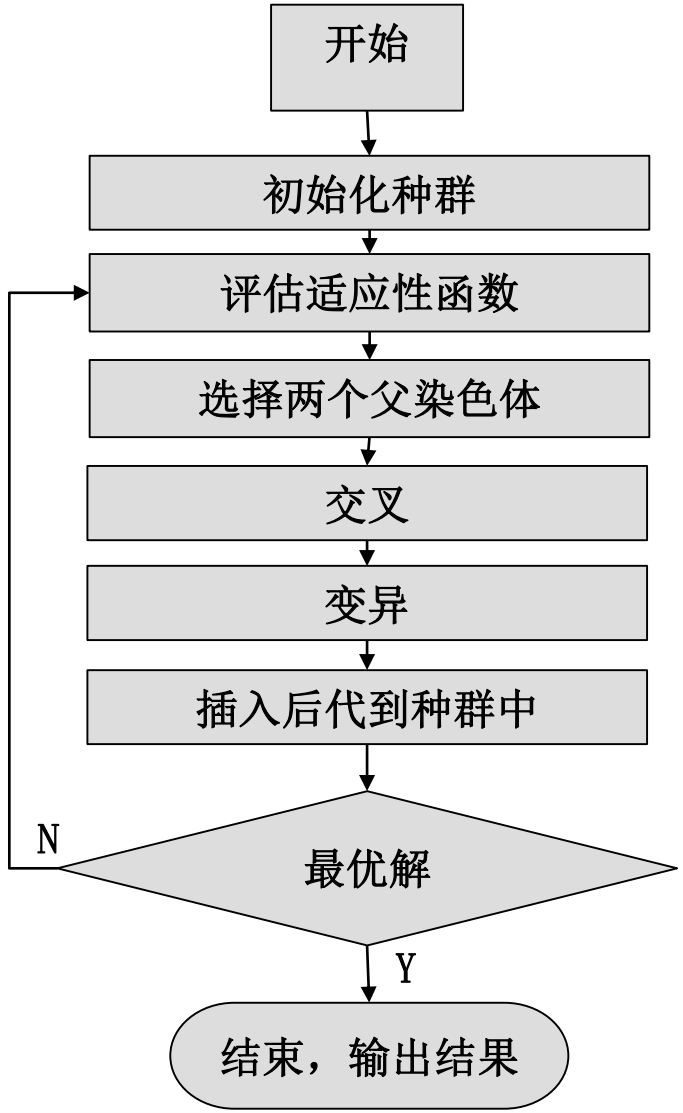
  - 以该概率随机在种群$pop(t)$中随机选一些染色体构成一个种群

$$newpop(t+1)=\{pop_j(t)|j=1,2,\cdots,N\}$$

  - Step4.通过交配，交配概率为P_C,得到一个有N个染色体的$crosspop(t+1);$
  - Step5.以一个较小的概率p,使得一个染色体的一个基因发生变异，形成$mutpop(t+1); t:=t+1$,一个新的群体$pop(t)=mutpop(t)$，返回第二步Step2.
- 模拟退火和遗传算法都可以采用马尔科夫链来做收敛分析!

8.4 进化算法和遗传算法

◆GA



Thank You !