优化算法引擎

高飞

2019 年 1 月 23 日

目录

1 简介

- 优化算法引擎，包含遗传算法（Genetic Algorithm，GA）、粒子群算法（Particle Swarm Optimization，PSO）。
- 编译器需支持 C++11 及以上标准。

2 遗传算法

2.1 遗传算法简介

遗传算法（Genetic Algorithm）基本思想是通过模拟自然进化过程搜索最优解，是一种通过模拟达尔文生物进化论的自然选择和遗传学机理的计算模型。遗传算法有 3 个最基本的操作：**选择，交叉，变异**。

2.2 算法流程

- 1. **初始化**：设置迭代停止条件，随机生成 N 个个体作为初始群体 $P(0)$。
- 2. **个体评价**：计算群体 $P(t)$ 中各个个体的适应度。
- 3. **选择运算**：将选择算子作用于群体。选择的目的是把优化的个体直接遗传到下一代或通过配对交叉产生新的个体再遗传到下一代。选择操作是建立在群体中个体的适应度评估基础上的。
- 4. **交叉运算**：将交叉算子作用于群体。遗传算法中起核心作用的就是交叉算子。
- 5. **变异运算**：将变异算子作用于群体。即是对群体中的个体串的某些基因座上的基因值作变动。群体 $P(t)$ 经过选择、交叉、变异运算之后得到下一代群体 $P(t+1)$。
- 6. **终止条件判断**：判断是否达到停止条件，若达到，终止计算，否则转到第 2 步。

适应度函数的重要性：适应度函数的选取直接影响遗传算法的**收敛速度**以及能否找到最优解。一般而言，适应度函数是由目标函数变换而成的，对目标函数值域的某种映射变换称为适应度的**尺度变换**（fitness scaling）。

适应度函数设计不当有可能出现欺骗问题：（1）进化初期，个别超常个体控制选择过程；（2）进化末期，个体差异太小导致陷入局部极值。

2.2.1 交叉（Crossover）

SBX 交叉算子 (模拟二进制单点交叉)

p_1 p_2 为父代基因，c_1 c_2 为子代基因, 一对父代基因交叉产生两个子代基因。

$$\begin{cases} c_1 = \dfrac{1}{2}\left[(1+\beta)p_1 + (1-\beta)p_2\right] \\ \\ c_2 = \dfrac{1}{2}\left[(1-\beta)p_1 + (1+\beta)p_2\right] \end{cases}$$

式中 β 为**均匀分布因子**。

$$\beta = \begin{cases} (2u)^{\frac{1}{\eta_c+1}}, & u \le 0.5 \\ \left[\dfrac{1}{2(1-u)}\right]^{\frac{1}{\eta_c+1}}, & u > 0.5 \end{cases}$$

上式中 u 是一个位于 $[0,1]$ 区间内的随机数，η_c 为**交叉分布指数**（大于 0），推荐为 1，η_c 越大，子代个体离父代越远。

以上公式满足：$p_1 + p_2 = c_1 + c_2$，$\beta = \left|\dfrac{c_2 - c_1}{p_2 - p_1}\right|$

2.2.2 变异（Mutation）

多项式变异，其变异算子形式是：$v'_k = v_k + \delta(u_k - l_k)$ v_k 表示一个父个体，u_k 为基因上限，l_k 为基因下限。其中：

$$\delta = \begin{cases} [2u + (1-2u)(1-\delta_1)^{\eta_m+1}]^{\frac{1}{\eta_m+1}}, & u \le 0.5 \\ 1 - [2(1-u) + 2(u-0.5)(1-\delta_2)^{\eta_m+1}]^{\frac{1}{\eta_m+1}}, & u > 0.5 \end{cases}$$

式中 $\delta_1 = \dfrac{v_k - l_k}{u_k - l_k}$，$\delta_2 = \dfrac{u_k - v_k}{u_k - l_k}$，$u$ 是一个位于 $[0,1]$ 区间内的随机数，η_m 是分布指数，推荐为 1。

2.2.3 选择（Selection）

轮盘赌选择

又称比例选择方法。其基本思想是：各个个体被选中的概率与其适应度大小成正比，个体适应度越高，被选中的概率越大。

具体操作如下：

- 1. 计算出群体中每个个体的适应度 $f(x_i)(i = 1, 2, ..., N)$，N 为群体大小；
- 2. 计算出每个个体被遗传到下一代群体中的概率；

$$P(x_i) = \frac{f(x_i)}{\sum_{j=1}^{N} f(x_j)}$$

- 3. 计算出每个个体的累积概率（q_i 称为个体 x_i 的累积概率）；

$$q_i = \sum_{j=1}^{i} P(x_j)$$

- 4. 在 $[0,1]$ 区间内产生一个均匀分布的随机数 r；
- 5. 若 $r < q_1$，则选择个体 1，否则，选择个体 k，使得 $q_{k-1} < r$ q_k 成立；
- 6. 重复步骤 4、5 共 N 次

3 粒子群算法

3.1 算法简介

- PSO 算法思想源于对鸟/鱼群捕食行为的研究，模拟鸟集群飞行觅食的行为，鸟之间通过集体的协作使群体达到最优目的，是一种基于 Swarm Intelligence 的优化方法；
- 没有遗传算法的“交叉”(Crossover) 和“变异”(Mutation) 操作，它通过追随当前搜索到的最优值来寻找全局最优；
- 与其他现代优化方法相比的一个明显特色是所需要调整的参数很少、简单易行，收敛速度快。

3.2 算法流程

- 第 1 步：在初始化范围内，对粒子群进行随机初始化，包括随机位置和速度；
- **第 2 步**：计算每个粒子的适应值；
- **第 3 步**：更新粒子个体的历史最优位置；
- **第 4 步**：更新粒子群体的历史最优位置；
- **第 5 步**：更新粒子的速度和位置；
- **第 6 步**：若未达到终止条件，则转第 2 步；

3.3 基本原理

D **维空间有** N **个粒子：**

- 粒子 i 位置：$X_i = (x_{i1}, x_{i2}, ..., x_{iD})$
- 粒子 i 速度：$V_i = (v_{i1}, v_{i2}, ..., v_{iD})$
- 粒子 i 经历过的最好位置：$P_i = (p_{i1}, p_{i2}, ..., p_{iD})$
- 种群所经历过的最好位置：$G = (g_1, g_2, ..., g_D)$

Note：第 d 维位置变化范围：$[X_{min,d}, X_{max,d}]$，速度变化范围：$[-V_{max,d}, V_{max,d}]$。若超出边界，则该维的速度或位置被限制为该维最大速度或边界位置。

粒子 i **的第** d **维速度更新公式 (标准 PSO)：**

$$v_{id}^k = wv_{id}^{k-1} + c_1 r_{1d}\left(p_{id} - x_{id}^{k-1}\right) + c_2 r_{2d}\left(g_d - x_{id}^{k-1}\right),\ \ i \in [1, N], d \in [1, D]$$

粒子 i **的第** d **维位置更新公式：**

$$x_{id}^k = x_{id}^{k-1} + v_{id}^{k-1}$$

- v_{id}^k——第 k 次迭代，粒子 i 飞行速度矢量的第 d 维分量；
- x_{id}^k——第 k 次迭代，粒子 i 位置矢量的第 d 维分量；
- c_1, c_2——学习因子，也称加速常数 (acceleration constant)，调节学习最大步长，一般取值范围是 $[0, 4]$，通常取 $c_1 = c_2 = 2$；

- r_{1d}, r_{2d}——两个随机数，取值分为 $[0,1]$，以增加搜索随机性；
- w——惯性权重，非负数，调节对解空间的搜索范围；

粒子速度更新公式包含三部分：

- 第一部分为**“惯性部分”**，即对粒子先前速度的记忆；
- 第二部分为**“自我认知”**部分，可理解为粒子 i 当前位置与自己最好位置之间的距离；
- 第三部分为**“社会经验”**部分，表示粒子间的信息共享与合作，可理解为粒子 i 当前位置与群体最好位置之间的距离。

1. 群体大小 N 是一个整数，N 很小时陷入局部最优解的可能性很大；N 很大时 PSO 的优化能力很好，但是当群体数目增长至一定水平时，再增长将不再有显著作用，而且数目越大计算量也越大。群体规模 N 一般取 $20 \sim 40$，对较难或特定类别的问题可以取到 $100 \sim 200$。

2. 粒子群的最大速度 V_{max} 对维护算法的探索能力与开发能力的平衡很重要，V_{max} 较大时，探索能力强，但粒子容易越过最优解；V_{max} 较小时，开发能力强，但是容易陷入局部最优解。V_{max} 一般设为每维变量变化范围的 $10\% \sim 20\%$。

3. 学习因子 $c_2 = 0$ 称为**自我认识型**粒子群算法，即“只有自我，没有社会”，完全没有信息的社会共享，导致算法收敛速度缓慢；学习因子 $c_1 = 0$ 称为**无私型**粒子群算法，即“只有社会，没有自我”，会迅速丧失群体多样性，容易陷入局部最优解而无法跳出；c_1, c_2 都不为 0，称为**完全型**粒子群算法，完全型粒子群算法更容易保持收敛速度和搜索效果的均衡，是较好的选择。

3.4 改进的 PSO 算法

3.4.1 惯性权重线性递减的粒子群算法 (PSO-W)

参数 w, c_1, c_2 的选择分别关系粒子速度的 3 个部分：**惯性部分**、**自身部分**和**社会部分**在搜索中的作用。如何选择、优化和调整参数，使得算法既能避免早熟又能比较快的收敛，对工程实践具有重要意义。

惯性权重 w 描述粒子上一代速度对当前代速度的影响。w 值较大，全局寻优能力强，局部寻优能力弱；反之，则局部寻优能力强。当问题空间较大时，为了在搜索速度和搜索精度之间达到平衡，通常做法是使算法在前期有较高的全局搜索能力以得到合适的种子，而在后期有较高的局部搜索能力以提高收敛精度，所以 w 不宜为一个固定的常数。

$$w = w_{max} - (w_{max} - w_{min}) \frac{k}{k_{max}}$$

w_{max} 最大惯性权重，w_{min} 最小惯性权重，k 当前迭代次数，k_{max} 为算法迭代总次数。通常 w_{max} 取 0.9，w_{min} 取 0.4。较大的 w 有较好的全局收敛能力，较小的 w 则有较强的局部收敛能力。因此，随着迭代次数的增加，惯性权重 w 应不断减少，从而使得粒子群算法在初期具有较强的全局收敛能力，而晚期具有较强的局部收敛能力。

3.4.2 带收缩因子的粒子群算法 (PSO-X)

学习因子 c_1 和 c_2 决定了微粒本身经验信息和其他微粒的经验信息对微粒运行轨迹的影响，反映了微粒群之间的信息交流。设置 c_1 较大的值，会使微粒过多地在局部范围内徘徊，而较大的 c_2 的值，则又会促使微粒过早收敛到局部最小值。微粒有效地控制飞行速度，使算法达到全局探测与

局部开采两者间的有效平衡，**Clerc 构造**了引入收缩因子的 PSO 模型，采用了压缩因子，这种调整方法通过合适选取参数，可确保 PSO 算法的收敛性，并可取消对速度的边界限制。速度公式如下：

$$v_{id}^{k+1} = K\left[v_{id}^{k} + c_1 r_{1d}\left(p_{id}^{k} - x_{id}^{k}\right) + c_2 r_{2d}\left(g_{d}^{k} - x_{id}^{k}\right)\right]$$

$$K = \frac{2}{\left|2 - C - \sqrt{C^2 - 4C}\right|}, C = c_1 + c_2,\ C > 4$$

K **为收缩因子**。通常取 $c_1 = c_2 = 2.05$，则 K 0.7298。实验表明，与使用惯性权重的 PSO 算法相比，使用收敛因子的 PSO 有更快的收敛速度。其实只要恰当的选取 w 和 c_1 c_2，两种算法是一样的。当惯性权重 PSO 中取 $w = 0.7298$，$c_1 = c_2 = K \times 2.05 = 1.49618$ 时，**两种算法等效**，因此使用收敛因子的 PSO 可以看作使用惯性权重 PSO 的特例。恰当的选取算法的参数值可以改善算法的性能。