Algorytmy ewolucyjne

zarys algorytmu ewolucyjnego

Piotr Lipiński

lipinski@ii.uni.wroc.pl

wersja robocza

Wprowadzenie

Algorytmy ewolucyjne zostały zaprojektowane do rozwiązywania problemu maksymalizacji funkcji celu $F : \Omega \to \mathbb{R}$ określonej na przestrzeni poszukiwań Ω.

Kilka szczególnych przypadków przestrzeni poszukiwań:

- klasyczne algorytmy genetyczne: $\Omega = \{0, 1\}^d$,
- klasyczne strategie ewolucyjne: $\Omega = \mathbb{R}^d$,
- algorytmy genetyczne z operatorami CX, PMX, OX: $\Omega = S_d$.

Naturalnie, algorytmy ewolucyjne mogą być używane także do rozwiązywania problemu minimalizacji funkcji celu.

wersja robocza

Wprowadzenie

Algorytmy ewolucyjne są inspirowane przez darwinowską teorie ewolucji populacji biologicznych, dlatego są zazwyczaj prezentowane przy użyciu specyficznego słownictwa.

Osobnik to struktura zawierająca m.in. informacje o argumencie funkcji celu. Taka informacja zwana jest chromosomem.

W klasycznych algorytmach genetycznych, osobnik składa się tylko z chromosomu, który jest binarnym wektorem długości d reprezentującym element przestrzeni poszukiwań $\Omega = \{0, 1\}^d$.

wersja robocza

Wprowadzenie

W klasycznych strategiach ewolucyjnych, osobnik zawiera, oprócz chromosomu, który jest wektorem rzeczywistym długości d reprezentującym element przestrzeni poszukiwań $\Omega = \mathbb{R}^d$, kilka dodatkowych parametrów używanych w procesie ewolucji.

Jeśli chromosom jest wektorem binarnym lub rzeczywistym długości d, to każda współrzędna takiego wektora zwana jest genem chromosomu.

Jeden osobnik zawsze odpowiada jednemu elementowi przestrzeni poszukiwań, który jest określony przez chromosom.

wersja robocza

Wprowadzenie

Populacja to zbiór określonej, skończonej liczby osobników. Odpowiada ona zbiorowi elementów przestrzeni poszukiwań określonych przez chromosomy osobników tworzących populację.

W populacji $\mathcal{P} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$ złożonej z N osobników $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$, każdy osobnik $\mathbf{x}_i$ odpowiada elementowi przestrzeni poszukiwań, więc może zostać mu przypisana wartośc funkcji celu, oznaczana jako $F(\mathbf{x}_i)$.

wersja robocza

Wprowadzenie

W celu porównania wartości funkcji celu w ramach jednej populacji, dla każdego osobnika $\mathbf{x}_i$ obliczana jest inna wartość, zwana wartością przystosowania i oznaczana przez $f(\mathbf{x}_i)$

$$f(\mathbf{x}_i) = \frac{F(\mathbf{x}_i) - F_{min}}{\sum_{j=1}^{N}(F(\mathbf{x}_j) - F_{min})},$$

gdzie $F_{min} = \min\{F(\mathbf{x}_1), F(\mathbf{x}_2), \ldots, F(\mathbf{x}_N)\}$.

Łatwo zauważyć, że

$$0 \leq \ f(\mathbf{x}_i) \leq 1 \quad \text{oraz} \quad \sum_{j=1}^{N} f(\mathbf{x}_j) = 1.$$

wersja robocza

Wprowadzenie

Osobniki o wysokiej wartości przystosowania są czasem zwane silnymi osobnikami lub dobrymi osobnikami.

Osobniki o niskiej wartości przystosowania są czasem zwane słabymi osobnikami lub złymi osobnikami.

wersja robocza

Wprowadzenie

Algorytm ewolucyjny przetwarza populację osobników. Każdy osobnik reprezentuje potencjalne rozwiązanie problemu, zakodowane w swoim genotypie zawartym w chromosomie. Genotyp pozwala na obliczenie fenotypu czyli argumentu funkcji celu.

(często osobnik jest utożsamiany ze swoim genotypem, a genotyp z fenotypem)

wersja robocza

Wprowadzenie

chromosom – wektor (traktowany jako struktura)

gen – współrzędna wektora (traktowana jako struktura)

genotyp – wartość wektora

fenotyp – element przestrzeni poszukiwań Ω

osobnik – wektor

populacja – macierz

(rozróżnienie między strukturą a wartością wektora jest pomijane, jeśli nie prowadzi to do nieporozumień)

wersja robocza

Schemat algorytmu ewolucyjnego

```
EVOLUTIONARY-ALGORITHM(F, N)
 1  P ← RANDOM-POPULATION(N);
 2  POPULATION-EVALUATION(P, F);
 3  while not TERMINATION-CONDITION(P)
 4  do
 5      EVOLUTIONARY-OPERATOR-1(P);
 6      EVOLUTIONARY-OPERATOR-2(P);
 7      EVOLUTIONARY-OPERATOR-3(P);
 8      ...
 9      EVOLUTIONARY-OPERATOR-K(P);
10      POPULATION-EVALUATION(P, F);
```

wersja robocza

Uwagi

- Specyficzna wiedza o problemie może zostać wykorzystana przy generowaniu populacji początkowej.
- Wykorzystując tę wiedzę można także konstruowac nowe operatory ewolucyjne.
- Warunki zakończenia powinny odpowiadać oczekiwaniom dotyczącym poszukiwanego rozwiązania.
- Należy uwzględniać ekonomiczność algorytmu.
- Jak uwzględnić ograniczenia zadania ?

wersja robocza

Algorytmy genetyczne

Prosty algorytm genetyczny (ang. *Simple Genetic Algorithm*, *SGA*) został zaproponowany przez Hollanda w 1975 r., a następnie rozwijany przez Goldberga, Michalewicza i innych.

Do celów akademickich została stworzona uproszczona wersja tego algorytmu zwana uproszczonym prostym algorytmem genetycznym (ang. *Simplified Simple Genetic Algorithm*, *SSGA*).

wersja robocza

Oznaczenia

Dla ustalenia uwagi rozważany będzie problem maksymalizacji funkcji celu $F : \Omega \to \mathbb{R}$, gdzie $\Omega = \{0, 1\}^d$ dla ustalonego d.

Populacja składa się z N osobników. Może być reprezentowana przez macierz $\mathbf{X}$ rozmiaru $N \times d$. Wiersz $\mathbf{x}_i$ tej macierzy odpowiada chromosomowi i-tego osobnika ($i = 1, 2, \ldots, N$). Współrzędna x_{ij} wektora $\mathbf{x}_i$ odpowiada j-temu genowi tego chromosomu ($j = 1, 2, \ldots, d$).

wersja robocza

Funkcje pułapki

OneMax – wartością funkcji jest liczba jedynek w danym ciągu binarnym

Pattern – wartością funkcji jest liczba miejsc, na których dany ciąg binarny jest zgodny z określonym wzorcem

DeceptiveOneMax – wartością funkcji jest liczba jedynek w danym ciągu binarnym, za wyjątkiem ciągu binarnego złożonego z samych zer, dla którego wartością funkcji jest długość ciągu plus jeden

K-DeceptiveOneMax – wartością funkcji jest suma wartości funkcji DeceptiveOneMax dla kolejnych odcinków o ustalonej długości danego ciągu binarnego

wersja robocza

SSGA

SIMPLIFIED-SIMPLE-GENETIC-ALGORITHM(F, N, M)

1 $\mathcal{P} \leftarrow$ RANDOM-POPULATION(N);
2 POPULATION-EVALUATION$(\mathcal{P}, F)$;
3 **while not** TERMINATION-CONDITION$(\mathcal{P})$
4 **do**
5 BLOCK-SELECTION$(\mathcal{P}, M)$;
6 UNIFORM-CROSSOVER$(\mathcal{P})$;
7 POPULATION-EVALUATION$(\mathcal{P}, F)$;

wersja robocza

SSGA – ewolucja

Populacja jest inicjowana losowo. Każdy gen x_{ij} każdego chromosomu $\mathbf{x}_i$ jest inicjowany losowo z rozkładem jednostajnym.

Populacja ewoluuje pod wpływem dwóch operatorów ewolucyjnych, mianowicie selekcji blokowej i krzyżowania jednostajnego, aż do osiągnięcia zbieżności.

wersja robocza

SSGA – Block Selection

M najsłabszych osobników jest usuwanych i M najsilniejszych osobników zajmuje ich miejsca.

Niech $i_1 < i_2 < \ldots < i_M$ będą indeksami M najsłabszych osobników w bieżącej populacji $\mathbf{X}$. Niech $j_1 < j_2 < \ldots < j_M$ będą indeksami M najsilniejszych osobników w bieżącej populacji $\mathbf{X}$.

Operator selekcji blokowej zastępuje $\mathbf{x}_{i_1}, \mathbf{x}_{i_2}, \ldots, \mathbf{x}_{i_M}$ odpowiednio przez $\mathbf{x}_{j_1}, \mathbf{x}_{j_2}, \ldots, \mathbf{x}_{j_M}$.

Zatem

$$\mathcal{P} = \{\mathbf{x}_1, \ldots, \mathbf{x}_{i_1-1}, \mathbf{x}_{j_1}, \mathbf{x}_{i_1+1}, \ldots, \mathbf{x}_{i_M-1}, \mathbf{x}_{j_M}, \mathbf{x}_{i_M+1}, \ldots, \mathbf{x}_N\}$$

po całej operacji.

SSGA – Uniform Crossover

W krzyżowaniu jednostajnym, populacja jest dzielona w pary. Każda para osobników $(\mathbf{x}_i, \mathbf{x}_j)$ generuje dwójkę potomków $\tilde{\mathbf{x}}_i$ i $\tilde{\mathbf{x}}_j$, którzy zastępują swoich rodziców. Potomek jest tworzony z dwójki rodziców przez dziedziczenie wartości każdego genu od losowo wybranego jednego z rodziców, podczas gdy drugi potomek dziedziczy wartość danego genu od pozostałego z rodziców.

wersja robocza

SGA

SIMPLE-GENETIC-ALGORITHM$(F, N, M, \theta_C, \theta_M)$

1 $\mathcal{P} \leftarrow$ RANDOM-POPULATION(N);
2 POPULATION-EVALUATION$(\mathcal{P}, F)$;
3 **while not** TERMINATION-CONDITION$(\mathcal{P})$
4 **do**
5 $\quad \mathcal{P}^{(P)} \leftarrow$ PARENT-SELECTION$(\mathcal{P}, M)$;
6 $\quad \mathcal{P}^{(C)} \leftarrow$ CROSSOVER$(\mathcal{P}^{(P)}, \theta_C)$;
7 $\quad$ MUTATION$(\mathcal{P}^{(C)}, \theta_M)$;
8 $\quad$ REPLACEMENT$(\mathcal{P}, \mathcal{P}^{(C)})$;
9 $\quad$ POPULATION-EVALUATION$(\mathcal{P}, F)$;

wersja robocza

SGA – Parent Selection

Z populacji bieżącej

$$\mathcal{P} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\},$$

algorytm tworzy nową populację

$$\mathcal{P}^{(P)} = \{\mathbf{x}_1^{(P)}, \mathbf{x}_2^{(P)}, \ldots, \mathbf{x}_M^{(P)}\},$$

zwaną *populacją rodziców*.

Osobniki są wybierane losowo w taki sposób; że prawdopodobieństwo wyboru osobnika $\mathbf{x}_i$ jest równe wartości jego przystosowania $f(\mathbf{x}_i)$.

Czasami taki wybór osobników nazywa się *metodą ruletki*.

wersja robocza

SGA – Crossover

Osobniki w populacji rodziców $\mathcal{P}^{(P)}$ są losowo łączone w pary, zakładając, że M jest parzyste, i każda para osobników $(\mathbf{x}_i^{(P)}, \mathbf{x}_j^{(P)})$ generuje dwóch potomków $\mathbf{x}_i^{(C)}$ oraz $\mathbf{x}_j^{(C)}$, które tworzą nową populację $\mathcal{P}^{(C)}$, zwaną *populacją potomków* lub *populacją dzieci*.

wersja robocza

SGA – Crossover

Z dużym prawdopodobieństwem θ_C, dzieci powstają w taki sposób, że losowo wybiera się punkt krzyżowania k, $0 \leq k \leq d$, co wprowadza podział chromosomu na dwa segmenty. Następnie, pierwsze dziecko dziedziczy od pierwszego rodzica geny w pierwszym segmencie oraz od drugiego rodzica geny w drugim segmencie, podczas kiedy drugie dziecko dziedziczy od drugiego rodzica geny w pierwszym segmencie oraz od pierwszego rodzica geny w drugim segmencie:

$$x_{il}^{(C)} = x_{il}^{(P)}, \quad \text{for } 1 \leq l \leq k,$$

$$x_{il}^{(C)} = x_{jl}^{(P)}, \quad \text{for } k < l \leq d$$

$$x_{jl}^{(C)} = x_{jl}^{(P)}, \quad \text{for } 1 \leq l \leq k,$$

$$x_{jl}^{(C)} = x_{il}^{(P)}, \quad \text{for } k < l \leq d.$$

wersja robocza

SGA – Crossover

Z małym prawdopodobieństwem $1 - \theta_C$, dzieci po prostu dziedziczą geny od swoich rodziców:

$$\mathbf{x}_i^{(C)} = \mathbf{x}_i^{(P)} \quad \text{oraz} \quad \mathbf{x}_j^{(C)} = \mathbf{x}_j^{(P)}.$$

wersja robocza

SGA – Mutation

Każdy gen każdego osobnika $\mathcal{P}^{(C)}$ jest negowany z bardzo małym prawdopodobieństwem θ_M.

wersja robocza

SGA – Replacement

Osobniki z populacji dzieci $\mathcal{P}^{(C)}$ zastępują osobniki z bieżącej populacji $\mathcal{P}$ w taki sposób, że wielkość populacji pozostaje taka sama.

Jest kilka znanych metod zastępowania, takich jak (μ, λ) oraz $(\mu + \lambda)$.

W (μ, λ), najlepsze osobniki z populacji dzieci tworzą nową populację, która zastępuje populacje bieżącą.

W $(\mu + \lambda)$, najlepsze osobniki z sumy mnogościowej bieżącej populacji i populacji dzieci tworzą nową populację, która zastępuje populacje bieżącą.

wersja robocza

SGA – Tournament Selection

W obu przypadkach, zamiast deterministycznego wyboru polegającego na wybraniu najlepszych osobników, można zastosować tzw. niedeterministyczną *metodę turnieju*.

W metodzie turnieju, najpierw wybierana jest losowo pewna liczba osobników, a następnie najlepszy z nich jest kwalifikowany do nowej populacji. Taki proces jest powtarzany wielokrotnie aż do nowej populacji zostanie wybrana odpowiednia liczba osobników.

Uwagi

- Algorytm SSGA jest bardzo prostym algorytmem i raczej nie jest stosowany w praktyce.
- Algorytm SGA posiada obecnie wiele nowszych wersji, nieraz w niczym juz nie przypominających oryginalnego SGA, i to głownie te nowe wersje są stosowane w praktyce.
- Istnieje wiele metod modyfikacji algorytmu SGA i dostosowywania do konkretnego problemu m.in.: skalowanie funkcji przystosowania, krzyżowanie z wielopunktowe, metody wielostartowe, częściowa zagłada.
- Uzasadnienie algorytmu opiera się na tzw. *teorii schematów*.

wersja robocza

CGA

Algorytm CGA (ang. *Compact Genetic Algorithm, CGA*) został zaproponowany przez Harika, Lobo i Goldberga w 1997 r. (raport TCGA-97006). Pomysł wprowadzony w CGA dotyczy rozkładów prawdopodobieństwa zmiennych losowych, które modelują wartości poszczególnych genów chromosomu.

wersja robocza