Sprawozdanie z projektu

Podstawy sztucznej inteligencji

Treść zadania

PW.AE.2. Maksymalizacja funkcji z użyciem algorytmu (μ+λ) i programowania ewolucyjnego.

Opis: Program znajduje maksimum funkcji

$$f: [-20,20]^n \to \mathbb{R}, \qquad f(x) = \prod_{i=1}^{n} cos(x_i) - 0.01 \sum_{i=1}^{n} \frac{x_i^2}{i}$$

Algorytm: Algorytm ewolucyjny (μ+λ) i algorytm programowania ewolucyjnego. Należy ustalić w przybliżeniu najlepsze parametry działania obu algorytmów dla n=10,30,100.

Interfejs: Program powinien demonstrować na bieżąco populację i wykres jakości znalezionego rozwiązania w funkcji numeru pokolenia.

Decyzje projektowe

Wybór języka: przez wgląd na to, iż projekt miał posiadać elementy graficzne oraz miał być w miarę uniwersalny, wybraliśmy język Java.

Parametry przekazywane do tworzenia populacji początkowej:

- μ - liczba osobników w populacji
- n – wymiarowość osobnika
- σ – max. generowane losowo odchylenie standardowe

Parametry przekazywane do algorytmu ewolucyjnego (μ + λ):

- λ – ile potomków w każdym pokoleniu
- c – oszacowana odległość pomiędzy maksimami lokalnymi

Parametry przekazywane do obu algorytmów:

- max rounds – ile maksymalnie generacji może być wygenerowanych
- min fitness – minimalna wartość funkcji przystosowania, aby zakończyć

Uznaliśmy, że nie ma sensu przekazywanie wszystkich parametrów od razu przy tworzeniu populacji początkowej, co więcej uznaliśmy, że pokolenie początkowe będzie generowane, a algorytm będzie działać na jego kopii, co umożliwi wykonanie na tych samych danych wielu powtórzeń algorytmu, z różnymi parametrami.

Ukrytymi parametrami, których nie przestawia użytkownik, uznaliśmy, że będą:

- Funkcja dopasowania, którą jednak łatwo można zmienić gdyż jest to w aplikacji funkcja statyczna w osobnej klasie
- Min, max – przedziały, z których losowane będą osobniki, które też można łatwo zmienić, gdyż są to parametry przekazywane funkcji generującej populację
- Funkcja wyboru osobników do następnej populacji, którą jednak także łatwo zmienić gdyż tak jak funkcja dopasowania jest to w funkcja statyczna w osobnej klasie

Użytkownik ma również możliwość oglądania wynikowej populacji w rzutowaniu na wybrane przez siebie wymiary, które może zmienić w aplikacji i oglądać wynikową populację w różnych wymiarach. Może również wyłączyć opcję podglądu populacji w wymiarach (przydatne dla dużej liczby osobników w algorytmie programowania ewolucyjnego).

Uznaliśmy, iż dla krzyżowania w algorytmie (μ + λ) użyjemy uśredniania przy tworzeniu zarodka gdyż jest prostsze w implementacji, zaś algorytm (μ + λ) i tak działa bardzo sprawnie nawet dla wielowymiarowych osobników, więc nie było potrzeby zmiany krzyżowania na wykorzystujące interpolację. Dodatkowo w tym algorytmie z racji tego, że podczas testów otrzymaliśmy wyniki wskazujące na to, że funkcja posiada wiele maksimów lokalnych rozmieszczonych mniej więcej w tych samych odległościach, wprowadziliśmy do algorytmu strategię doboru podobnych osobników, których szansa na reprodukcję zależy od wprowadzonego parametru c.

Uznaliśmy także, iż najlepszą opcją wyboru osobników do następnego pokolenia w obu algorytmach będzie algorytm selekcji rankingowej, dzięki któremu nie trzeba przeskalowywać wyników funkcji dopasowania, a który sprawdza się dość dobrze. W programie jest także zaimplementowany algorytm μ-najlepszych oraz wybieranie losowe z zachowaniem k-procentowej grupy najlepszych osobników, który łatwo można zamienić z selekcją rankingową.

Uznaliśmy także, że nie ma sensu móc utracić najlepszego osobnika z poprzedniej populacji, więc w programie jest też zapisana strategia elitarna, która powoduje przepisanie najlepszego osobnika z poprzedniego pokolenia, w miejsce najgorszego osobnika z nowego pokolenia.

Wnioski i najlepsze parametry

Algorytm programowania ewolucyjnego:

Algorytm jest zrównoleglony co powoduje jego bardzo szybkie wykonanie nawet dla wielu osobników. Dla niewielkiej ilości wymiarów, algorytm potrafi znaleźć rozwiązanie szybciej od algorytmu (μ + λ), jednakże wraz ze wzrostem liczby wymiarów, maleje szansa na trafienie osobnika znajdującego się obok maksimum globalnego, zaś przez brak krzyżowania, osobnik potrafi praktycznie nie być w stanie wyjść z maksimum lokalnego.

Zwiększenie liczby osobników pozwala na znalezienie lepszych osobników w mniejszej ilości przebiegów za to trwających dłużej.

Do algorytmu programowania ewolucyjnego naturalnym podejściem jest stworzenie jak największej liczby osobników i oczekiwanie, że któryś trafi w maksimum globalne, co właśnie wraz ze wzrostem liczby wymiarów jest coraz trudniejsze, jednakże dla n = 10, 30 algorytm sprawdza się dość dobrze z następującymi parametrami:

N = 10:

- μ = 30000
- σ = 5
- max rounds = 70
- min fitness = 0.9

Znajdywane w takiej konfiguracji rozwiązanie oscyluje na poziomie **0.8**-**0.95**, zaś czas wykonania (bez pokazywania populacji na wykresie) to w granicach **1**-**3** sekund.

N = 30:

- μ = 10000
- σ = 8
- max rounds = 500
- min fitness = 0.9

Znajdywane w takiej konfiguracji rozwiązanie oscyluje na poziomie **0.6**-**0.9**, zaś czas wykonania to w granicach **6**-**12** sekund.

N = 100:

- μ = 1000
- σ = 8
- max rounds = 3000
- min fitness = 0.9

Znajdywane w takiej konfiguracji rozwiązanie nie jest zadowalające, ponieważ rzadko przekracza 0 i zazwyczaj oscyluje w granicach **-0.5 – -0.05**, zaś czas wykonania sięga ok. **12** sekund.

Algorytm (μ + λ):

Algorytm nawet dla wielowymiarowych osobników działa bardzo sprawnie. Nie potrzebuje wielu osobników do znalezienia maksimum globalnego. Nie reprodukuje przez krzyżowanie słabych osobników po dobraniu odpowiedniego parametru c, zaś jedyną jego wadą jest to, że nie wykonuje się równolegle.

Parametr c jest we wszystkich przejściach taki sam, dobrany został na podstawie obserwacji i doświadczeń. Za mała jego wartość wpływa negatywnie na czas obiegu jednej generacji, gdyż mało który osobnik leży na tyle blisko, żeby mieć dużą szansę reprodukcji, zaś za duża jego wartość powoduje wybieranie osobników, których potomkowie będą znajdować się prawdopodobnie w minimach lokalnych.

Do algorytmu (μ + λ) naturalnym podejściem jest stworzenie jak najmniejszej liczby osobników i wielokrotnie więcej potomków reprodukowanych. Dla wielowymiarowych osobników, mniejsza ilość w populacji jest tym bardziej wskazana, gdyż znacznie skraca długość obliczeń, zaś dzięki krzyżowaniu i tak mamy dużą szansę trafić na maksimum globalne.

N = 10:

- μ = 100
- λ = 700
- σ = 3
- c = 3
- max rounds = 100
- min fitness = 0.999

Znajdywane w takiej konfiguracji rozwiązanie oscyluje na poziomie **0.95-0.999**, zaś czas wykonania to mniej niż **0.5** sekundy.

N = 30:

- μ = 100
- λ = 700
- σ = 3
- c = 3
- max rounds = 100
- min fitness = 0.999

Znajdywane w takiej konfiguracji rozwiązanie oscyluje na poziomie **0.95-0.999**, zaś czas wykonania to **0.5-1** sekundy.

N = 100:

- μ = 20
- λ = 140
- σ = 5
- c = 3
- max rounds = 1000
- min fitness = 0.999

Znajdywane w takiej konfiguracji rozwiązanie oscyluje na poziomie **0.99-0.999**, zaś czas wykonania to **5-7** sekund.

Instrukcja do programu

Po uruchomieniu wpisujemy parametry, aby utworzyć populację początkową:

- μ - liczba osobników w populacji
- n – wymiarowość osobnika
- σ – max. odchylenie standardowe generowane losowo

Następnie przyciskiem *Generate new data* tworzymy niezmienną populację, która będzie populacją wejściową (początkową) dla algorytmów.

Aby uruchomić algorytm (μ + λ) należy wypełnić pola parametrów:

- λ – liczba potomków generowanych w każdym pokoleniu
- c – szacowana odległość pomiędzy maksimami lokalnymi
- max rounds – maksymalna liczba utworzonych generacji
- min fitness – minimalna wartość funkcji przystosowania, aby zakończyć algorytm

Po kliknięciu przycisku *Solve MPL*, algorytm zacznie działać zaś jego wyniki zostaną wyświetlone na dwóch wykresach:

1. Wykres współrzędnych w 2 wybranych wymiarach dla populacji
2. Wykres jakości wartości funkcji dopasowania najlepszego osobnika w funkcji generacji

Wyświetlanie punktów na wykresie 1 można wyłączyć, odznaczając check box'a w prawej górnej części interfejsu aplikacji – *Show chart*. Jest to przydatne przy dużej liczbie osobników, gdy wykres współrzędnych jest zazwyczaj mało interesujący (zwłaszcza przy dużej liczbie wymiarów), zaś głównie pragniemy zanalizować wykres jakości najlepszego osobnika w generacji.

Aby uruchomić algorytm programowania ewolucyjnego należy wypełnić pola parametrów:

- max rounds – maksymalna liczba utworzonych generacji
- min fitness – minimalna wartość funkcji przystosowania, aby zakończyć algorytm

Po kliknięciu przycisku Solve EP, algorytm zacznie działać zaś jego wyniki podobnie jak w algorytmie (μ + λ) - zostaną wyświetlone na dwóch wykresach. Przy dużej ilości osobników zaleca się wyłączenie wyświetlania populacji na wykresie współrzędnych.

Struktury danych użyte w programie

Program jest maksymalnie uniwersalny poprzez zastosowanie abstrakcyjnych interfejsów i oferuje łatwość w dołączaniu własnych klas, modułów czy funkcji. Dzieli się na struktury danowe, funkcyjne oraz graficzne.

Struktury danowe:

- Individual – interfejs pojedynczego osobnika, który musi implementować klasa definiująca osobnika, jeśli chce używać algorytmów w programie
- Population – interfejs populacji, który musi implementować klasa agregująca osobników, jeśli chce używać algorytmów w programie
- Indiv – klasa reprezentująca osobnika, implementująca interfejs Individual
- Popul – klasa reprezentująca populację, implementująca interfejs Population

Struktury funkcyjne:

- Fitness – klasa posiadająca jedynie 1 metodę statyczną fitFun(Individual) obliczającą dla danego osobnika jest wartość funkcji przystosowania
- Choosing – klasa odpowiadająca za statyczne metody wyboru nowej populacji z 2 jej przekazanych, posiada statyczna metodę choose(Population, Population), która jest metodą wołaną przez algorytmy domyślnie
- Solver – klasa odpowiadające za statyczne metody algorytmów ewolucyjnych, do których przekazywane są struktury wraz z parametrami, w metodach automatycznie uwzględniona jest strategia elitarna, zachowująca najlepszego osobnika z pokolenia

Struktury graficzne:

- Controller – klasa służąca obsłudze interfejsu graficznego aplikacji, posiada metody wywoływane przy naciśnięciu odpowiednich przycisków, jak i również prywatne np. do wyświetlania danych na wykresie