Politechnika Łódzka

Wydział: EEIA
Grupa: 7D30
Dzień tygodnia: Niedziela

Rok akad.: 2017/2018
Semestr: VII

Sprawozdanie z ćwiczenia z
Podstawy Sztucznej Inteligencji

Temat ćwiczenia: **Algorytm Genetyczny**

Data wykonania ćwiczenia	Podpis	Data oddania sprawozdania	Podpis
01.12.2017		01.12.2017	

Wykonał:
Radosław Subczyński
Nr inteksu 200137

Problem definiuje środowisko, w którym istnieje pewna populacja osobników. Każdy z osobników ma przypisany pewien zbiór informacji stanowiących jego genotyp, a będących podstawą do utworzenia fenotypu. Fenotyp to zbiór cech podlegających ocenie funkcji przystosowania modelującej środowisko. Innymi słowy - genotyp opisuje proponowane rozwiązanie problemu, a funkcja przystosowania ocenia, jak dobre jest to rozwiązanie.

Genotyp składa się z chromosomów, gdzie zakodowany jest fenotyp i ewentualnie pewne informacje pomocnicze dla algorytmu genetycznego. Chromosom składa się z genów.

Wspólnymi cechami algorytmów ewolucyjnych, odróżniającymi je od innych, tradycyjnych metod optymalizacji, są:

1. stosowanie operatorów genetycznych, które dostosowane są do postaci rozwiązań,
2. przetwarzanie populacji rozwiązań, prowadzące do równoległego przeszukiwania przestrzeni rozwiązań z różnych punktów,
3. w celu ukierunkowania procesu przeszukiwania wystarczającą informacją jest jakość aktualnych rozwiązań,
4. celowe wprowadzenie elementów losowych.

Zapis algorytmu[

Najczęściej działanie algorytmu przebiega następująco:

1. Losowana jest pewna populacja początkowa.
2. Populacja poddawana jest ocenie (selekcja). Najlepiej przystosowane osobniki biorą udział w procesie reprodukcji.
3. Genotypy wybranych osobników poddawane są operatorom ewolucyjnym:
    1. są ze sobą kojarzone poprzez złączanie genotypów rodziców (krzyżowanie),
    2. przeprowadzana jest mutacja, czyli wprowadzenie drobnych losowych zmian.
4. Rodzi się drugie (kolejne) pokolenie. Aby utrzymać stałą liczbę osobników w populacji te najlepsze (według funkcji oceniającej fenotyp) są powielane, a najsłabsze usuwane. Jeżeli nie znaleziono dostatecznie dobrego rozwiązania, algorytm powraca do kroku drugiego. W przeciwnym wypadku wybieramy najlepszego osobnika z populacji - jego genotyp to uzyskany wynik.

Działanie algorytmu genetycznego obejmuje kilka zagadnień potrzebnych do ustalenia:

1. ustalenie genomu jako reprezentanta wyniku
2. ustalenie funkcji przystosowania/dopasowania
3. ustalenie operatorów przeszukiwania

Funkcja przystosowania

Proces wyboru osobników poddanych ocenie następuje według określonych kryteriów. Kryteria te zapisuje się w postaci *funkcji oceny* albo inaczej *funkcji przystosowania*. Algorytm genetyczny dąży zwykle do jej minimalizacji (maksymalizacji). Jako kryterium często przyjmowana jest funkcja celu rozważanego problemu optymalizacji.

Funkcja oceny

Funkcja oceny to *miara jakości* dowolnego osobnika (fenotypu, rozwiązania) w populacji. Dla każdego osobnika jest ona obliczana na podstawie pewnego modelu rozwiązywanego problemu. Załóżmy dla przykładu, że chcemy zaprojektować obwód elektryczny o pewnej charakterystyce. Funkcja oceny będzie premiowała rozwiązania najbardziej zbliżonej do tej charakterystyki, zbudowane z najmniejszej liczby elementów. W procesie selekcji faworyzowane będą najlepiej przystosowane osobniki. One staną się "rodzicami" dla populacji.

Metody selekcji[edytuj]

Istnieje wiele metod selekcji. Dla przykładu można przedstawić tzw. metodę ruletki. Budujemy wirtualnie koło, którego wycinki odpowiadają poszczególnym osobnikom. Im lepszy osobnik, tym większy wycinek koła zajmuje. Rozmiar wycinków może zależeć od wartości funkcji oceny, jeśli wysoka wartość oceny oznacza wysokie przystosowanie. W takim układzie prawdopodobieństwo, że lepszy osobnik zostanie wybrany jako rodzic, jest większe. Niestety ewolucja przy takim algorytmie z każdym krokiem zwalnia. Jeżeli osobniki są podobne, to każdy dostaje równy wycinek koła fortuny i presja selekcyjna spada. Algorytm słabiej rozróżnia osobniki dobre od słabszych.

Pozbawiona tej wady jest metoda rankingowa. Obliczamy dla każdego osobnika *funkcję oceny* i ustawiamy je w szeregu najlepszy-najgorszy. Pierwsi na liście dostają prawo do rozmnażania, a reszta usuwana jest z populacji. Wadą metody jest niewrażliwość na różnice pomiędzy kolejnymi osobnikami w kolejce. Może się okazać, że sąsiadujące na liście rozwiązania mają różne wartości funkcji oceny, ale dostają prawie taką samą ilość potomstwa.

Istnieją także metody *selekcji wielokryterialnej*. Tworzymy kilka różnych funkcji oceny (oceniających pewne wybrane cechy osobników osobno). Dla przykładu osobniki mogą być ułożone nie w jednym, ale w kilku szeregach najlepszy-najgorszy, a proces selekcji jest bardziej złożony.

Jak widać, *selekcja* daje większe prawdopodobieństwo reprodukcji osobnikom o dużym przystosowaniu, więc kolejne pokolenia są coraz lepiej przystosowane. Spada jednak różnorodność genotypu populacji - populacja z czasem zostaje zmonopolizowana przez nieznacznie różniące się (lub wręcz identyczne) odmiany tego samego osobnika. Objawia się to zbieżnością kolejnych, najlepszych rozwiązań do pewnej granicy. Czasami zbieżność jest przedwczesna, a ewolucja utyka i uzyskane rozwiązania przedstawiają pewne ekstrema lokalne. Mogą być one dalekie od oczekiwanych rozwiązań globalnych, czyli tych najlepszych w całej przeszukiwanej przestrzeni. Częściowym rozwiązaniem tego problemu jest losowa mutacja genotypu osobników.

```
public class AlgortymGenetyczny {

    private int populationCount;
    private int sumpaPrzystosowania;
    private ArrayList<PopulationItems> populationsList = new ArrayList<>();
    private ArrayList<PopulationItems> startpopulationsList = new ArrayList<>();


    AlgortymGenetyczny(){
        populationCount = randomNumberInRange(1,100);
        startpopulationsList = createPopulationsList(populationCount);
        populationsList = startpopulationsList;
        getSumaPrzysotsoania(populationsList);
        sortPopulationList(populationsList);
        showPopulationList(populationsList);
        startSelectionChromosome(populationsList);
    }

    private void startSelectionChromosome(ArrayList<PopulationItems> populationsList) {
        PopulationItems populationItems1;
        PopulationItems populationItems2;
        int losowaniePar = populationsList.size()/2;
        for (int i =0;i<losowaniePar;i++) {
            populationItems1 =
populationsList.get(randomNumberInRange(0,populationsList.size()-1));
            populationItems2 =
populationsList.get(randomNumberInRange(0,populationsList.size()-1));
            krzyzowanie(populationItems1,populationItems2);
            mutacja(populationItems1);
        }

    }

    private void mutacja(PopulationItems populationItems1) {
    }

    private void krzyzowanie(PopulationItems populationItems1, PopulationItems
populationItems2) {
        System.out.println("Para Do krzyzowania :");
        System.out.println("1 Osobnik do krzyzowania "
+populationItems1.getBinaryScoreScore());
        System.out.println("2 Osobnik do krzyzowania "
+populationItems2.getBinaryScoreScore());
        System.out.println("");

        String sufixPref1 = populationItems1.getBinaryScoreScore().substring(0,4);
        String prefixPref1 = populationItems1.getBinaryScoreScore().substring(4,8);
        System.out.println("1 Osobnik do krzyzowania " +sufixPref1 +" : " +
prefixPref1);

        String sufixPref2 = populationItems2.getBinaryScoreScore().substring(0,4);
        String prefixPref2 =populationItems2.getBinaryScoreScore().substring(4,8);
        System.out.println("2 Osobnik do krzyzowania " +sufixPref2 +" : " +
prefixPref2);

        populationItems1.setScore(Integer.parseInt(sufixPref1+prefixPref2, 2));
        populationItems2.setScore(Integer.parseInt(sufixPref2+prefixPref1, 2));

    }

    private void sortPopulationList(ArrayList<PopulationItems> populationsList) {
        Collections.sort(populationsList);
    }

    private void getSumaPrzysotsoania(ArrayList<PopulationItems> populationsList) {
        for(PopulationItems populationItems : populationsList){
            sumpaPrzystosowania +=populationItems.getAssessmentOfAdaptation();
```

```
        }
        System.out.println("Suma przystosowania to : " + sumpaPrzystosowania);
    }

    private int  randomNumberInRange(int min, int max) {
        Random random = new Random();
        return random.nextInt((max - min) + 1) + min;
    }

    private ArrayList<PopulationItems> createPopulationsList(int populationCount){
        ArrayList<PopulationItems> tempArrayList = new ArrayList<>();
        for(int i = 0 ; i<populationCount ; i++){
            tempArrayList.add(new PopulationItems(randomNumberInRange(1,127)));
        }

        return tempArrayList;
    }

    private void showPopulationList(ArrayList<PopulationItems> populations){
        System.out.println("Wielkosc Listy populacji :" + populations.size());
        for(PopulationItems populationItems : populations){
            System.out.println("Osobnik wynik: "+populationItems.getScore() +
                    " Osobnik wynik w zapis binarny:
"+populationItems.getBinaryScoreScore() + " Ocena przystosowania: "
            + (double)populationItems.getAssessmentOfAdaptation()/sumpaPrzystosowania);
        }
    }
}
```