Inteligencja obliczeniowa - stud. niestacjonarne

Laboratorium 1: Algorytmy genetyczne

Zadania na laboratoria

Uwaga! Zapisuj na bieżąco obszar roboczy podczas wykonywania zadań. W dowolnym momencie możesz go później załadować i kontynuować zadania.

Zadanie 1

Korzystając z odpowiednich komend konsolowych w pakiecie R wykonaj poniższe czynności.
a) Oblicz ile to jest 45*678
b) Wczytaj pod zmienne dwa wektory: x=(7; 4; 2; 0; 9) oraz y=(2; 1; 5; 3; 3).
c) Oblicz za pomocą jednej operacji wektor, którego współrzędne są sumami współrzędnych wektorów x i y. Wyświetl ten wektor.
d) Oblicz za pomocą jednej operacji iloczyn wektorów.
e) Oblicz iloczyn macierzy korzystając z mnożenia macierzowego.

$$\begin{bmatrix} 0 & 2 & 1 \\ 1 & 6 & 4 \\ 5 & 0 & 3 \end{bmatrix} \qquad \begin{bmatrix} 9 & 8 & 7 \\ 1 & 2 & 7 \\ 4 & 9 & 2 \end{bmatrix}$$

f) Napisz funkcję `suma(a,b)` sumującą podane na wejściu liczby lub wektory i zwracającą wynik.
g) Zapisz obszar roboczy (workspace) jako plik z rozszerzeniem .Rdata. Plik ten zawsze można wczytać na nowo (nawet w domu) i wszystkie wcześniej wykonane komendy i zmienne będą dostepne.

Zadanie 2

Korzystając z odpowiednich komend i załączonego pliku wykonaj zadania.
a) Sprawdź ścieżkę do „working directory” i ewentualnie ustaw ją na własną.
b) Pobierz ze strony i otwórz plik osoby.csv ze wskazaniem, że kolumny mają nagłówki, a separatorem jest przecinek. Następnie wyświetl tabelę danych.
c) Wyświetl same imiona.
d) Wyświetl tylko dane kobiet.
e) Wyświetl średnią wieku za pomocą wbudowanej komendy.
f) Co robi komenda „summary”? Przetestuj i podaj jaki jest średni wiek ludzi z tabeli.
g) Zapisz obszar roboczy.

Zadanie 3

W problemie plecakowym pytamy, jakie przedmioty wziąć do plecaka o ograniczonej objętości, by ich wartość była najwyższa. Załóżmy, że jedziemy na obóz, ale w plecaku zmieścimy przedmioty o łącznej wadze 20. Do wyboru mamy nastepujące przemioty:

ITEM	SURVIVALPOINTS	WEIGHT
pocketknife	10.00	1.00
beans	20.00	5.00

potatoes	15.00	10.00
unions	2.00	1.00
sleeping bag	30.00	7.00
rope	10.00	5.00
compass	30.00	1.00

Jakie przedmioty należy wziąć? Korzystając z pakietu R i poniższych wskazówek rozwiąż ten problem za pomocą algorytmu genetycznego.

a) Instalacja (jednorazowo) i załadowanie biblioteki

```
install.packages("genalg")
library(genalg)
install.packages("ggplot2")
library(ggplot2)
```

b) Dodawanie zbioru danych i limitu plecaka

```
dataset <- data.frame(
      item = c("pocketknife", "beans", "potatoes", "unions",
               "sleeping bag", "rope", "compass"),
      survivalpoints = c(10, 20, 15, 2, 30, 10, 30),
      weight = c(1, 5, 10, 1, 7, 5, 1))

weightlimit <- 20
```

c) Chromosomy (rozwiązania) to ciągi 7-bitów. Dla każdego z 7 przedmiotów wybieramy 0 (nie bierzemy tego przedmiotu) lub 1 (bierzemy ten przedmiot do plecaka). Jakim przedmiotom odpowiada ciąg 1001100? Ile są warte wszystkie przedmioty? Sprawdź za pomocą poniższych komend.

```
chromosome = c(1, 0, 0, 1, 1, 0, 0)
dataset[chromosome == 1, ]

cat(chromosome %*% dataset$survivalpoints)
```

d) Najważniejszy etap to zdefiniowanie funkcji fitness. Sprawdza i zwraca ile ważą przedmioty dla rozwiązania x. Jeśli przekroczyły limit plecaka to zwraca 0. Przeanalizuj poniższą funkcję. Zwróć uwagę na minus w komendzie return. Najlepszy wynik to, wg interpretacji paczki genalng, najniższy wynik. Nasz najwyższy i najlepszy trzeba więc odwrócić na ujemny.

```
fitnessFunc <- function(x) {
     current_solution_survivalpoints <- x %*% dataset$survivalpoints
     current_solution_weight <- x %*% dataset$weight

     if (current_solution_weight > weightlimit)
          return(0) else return(-current_solution_survivalpoints)
}
```

e) Definiujemy algorytm genetyczny wprowadzając odpowiednie parametry i uruchamiamy go. Jakie jest najlepsze rozwiązanie?

```
GAmodel <- rbga.bin(size = 7, popSize = 200, iters = 100,
            mutationChance = 0.01, elitism = T, evalFunc = fitnessFunc)
summary(GAmodel, echo=TRUE)
```

f) Jako dodatkowy punkt zadania można przeprowadzić analizę działania algorytmu obserwując zmieniające się populacje.

```
iter<-100
animate_plot <- function() {
    for (i in seq(1, iter)) {
        temp <- data.frame(Iteracja = c(seq(1, i), seq(1, i)), Legenda = c(rep("Średnia",
            i), rep("Najlepsza", i)), WartoscFitness = c(-GAmodel$mean[1:i], -GAmodel$best[1:i]))

        pl <- ggplot(temp, aes(x = Iteracja, y = WartoscFitness, group = Legenda,
            colour = Legenda)) + geom_line() + scale_x_continuous(limits = c(0,
            iter)) + scale_y_continuous(limits = c(0, 110)) + geom_hline(y =
max(temp$WartoscFitness),
            lty = 2) + annotate("text", x = 1, y = max(temp$WartoscFitness) +
            2, hjust = 0, size = 3, color = "black", label = paste("Najlepsze rozwiązanie:",
            max(temp$WartoscFitness))) + scale_colour_brewer(palette = "Set1")

        print(pl)
    }
}
```

Wywołanie:
`> animate_plot()`
W której iteracji (pokoleniu) znaleziono już najlepsze rozwiązanie?

Praca domowa

Zadanie 1
Przypomnij sobie jaką postać ma formuła logiczna w koniunkcyjnej postaci normalnej o klauzalach co najwyżej długości 3 (w skrócie: 3-CNF). Czy poniższa formuła o 7 klauzulach w postaci 3-CNF jest spełnialna?
$\varphi = (\neg x_1 \vee x_2 \vee x_4) \wedge (\neg x_2 \vee x_3 \vee x_4) \wedge (x_1 \vee \neg x_3 \vee x_4) \wedge (x_1 \vee \neg x_2 \vee \neg x_4) \wedge (x_2 \vee \neg x_3 \vee \neg x_4) \wedge (\neg x_1 \vee x_3 \vee \neg x_4) \wedge (x_1 \vee x_2 \vee x_3)$
Sprawdź dla podstawienia $x_1=1$, $x_2=0$, $x_3=1$, $x_4=1$ oraz dla podstawienia $x_1=1$, $x_2=1$, $x_3=1$, $x_4=1$.

Na stronie http://people.sc.fsu.edu/~jburkardt/data/cnf/cnf.html (a także na stronie http://www.cs.ubc.ca/~hoos/SATLIB/benchm.html) znajdują się spakowane zakodowane formuły 3-CNF w formacie *dimcas cnf*, wraz z objaśnieniem formatu. Powyższa formuła φ w danym formacie byłaby tabelą postaci:

-1	2	4	0
-2	3	4	0
1	-3	4	0
1	-2	-4	0

2 -3 -4 0
-1 3 -4 0
1 2 3 0

Widać, że indeksy iksów odpowiadają numerom w tabeli, a minusy odpowiadają negacji.

a) Zapisz powyższą formułę jako tabelę np. phi1

b) Weź jedną z formuł z powyższych stron zakodowaną jako tabela powyższego typu i zapisz ją do tabeli np. pod zmienną phi2. Być może wygodnie będzie wcześniej usunąć nagłowek ściągniętego pliku. Zapisz obszar roboczy.

c) Mamy daną formułę w postaci tabeli (phi1 z podpunktu a lub phi2 z podpunktu b). Stwórz wektor będący podstawieniem do tej formuły z wartościami 0 lub 1 (dla formuły phi1 może być podstawienie pod iksy np. [1,0,1,1]). Napisz funkcję, która będzie liczyła i zwracała liczbę klauzul (trójek zmiennych w nawiasach) spełnionych (mających wartośc 1) po dokonaniu podstawienia.

```
fitness(podstawienie){
      formula <- phi1
      …
}
```

Przykładowo dla `formula<-phi1` wywołanie komendy `fitness(c(1,0,1,1))` da wynik 6, a wywołanie komendy `fitness(c(1,1,1,1))` da wynik 7.

d) Przetestuj działanie funkcji dla phi2 i dłuższych wektorów podstawień.

Zadanie 2

Problem 3-SAT odpowiada na pytanie „czy dana formuła 3-CNF jest spełnialna?” tzn niezaleznie od tego jakie podstawienie otrzymamy wartość logiczną 1. Problem ten jest NP-zupełny, dlatego nie istnieją efektywne algorytmy rozwiązujące ten problem w wielomianowym czasie.

Można spróbować za to rozwiązać ten problem za pomocą algorytmu genetycznego. Korzystając z pakietu R i biblioteki *genalg* znajdź podstawienie dla jednej z formuł z zamieszczonych na wczesniej wymienionych stronach. Rozwiązanie będzie analogiczne do treści zadania 3 z zajęć, przy czym różnić się będą zmienne `size` (chromosom to podstawienie do formuły, więc size to liczba zmiennych w formule) i `evalFunc` (tu wystarczy wstawić funkcję fitness z poprzedniego zadania ustawiając `formula<-aktualnie_badane_phi` i dodając minus przed liczbą zwracaną na wyjściu). Być może będzie też trzeba zmienić liczbę iteracji i wielkość populacji (warto poeksperymentować).

Przy wizualizowaniu algorytmu na wykresie dopasuj jego granice, tak by wszystko było widoczne i przejrzyste.