Algorytmy zrandomizowane

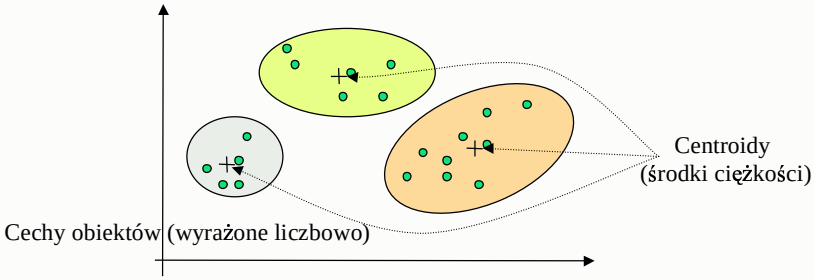
www.qed.pl/ai/nai2003

PLAN WYKŁADU

- Inne zadania optymalizacyjne
  - grupowanie
- Generowanie liczb losowych
- Metody Monte Carlo i Las Vegas
  - przykłady zastosowań
- Przeszukiwanie losowe
  - metoda tabu
  - wychładzanie

INNE ZADANIA OPTYMALIZACYJNE - GRUPOWANIE

Algorytmy łączące obiekty w większe grupy na podstawie ich wzajemnego podobieństwa (uczenie bez nadzoru).

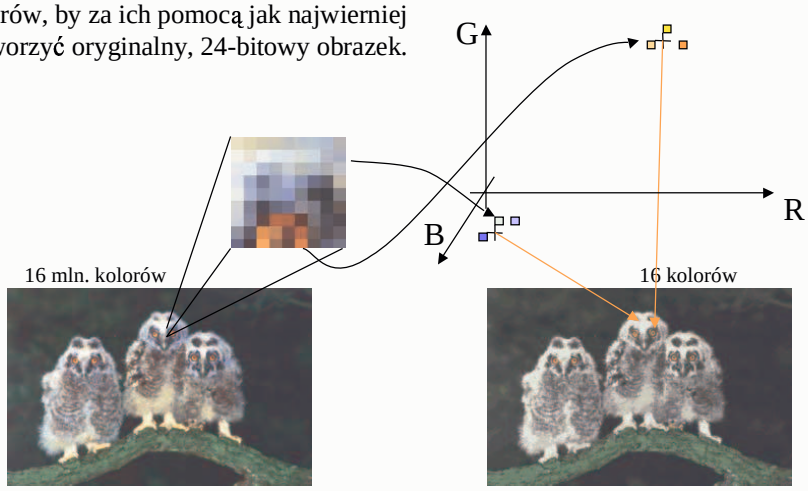


Kryterium „podobieństwa" obiektów oparte jest na ich wzajemnej odległości.

Zadanie optymalizacyjne: znaleźć taki podział, żeby odległości między obiektami w jednej klasie były jak najmniejsze, a między klasami - jak największe.

Algorytmy grupujące - przykład zastosowania

Zadanie kwantyzacji kolorów: znaleźć takich 16 kolorów, by za ich pomocą jak najwierniej odtworzyć oryginalny, 24-bitowy obrazek.



Zwykle stosowane algorytmy: zbliżone do zachłannych (np. k-means) lub wspinaczki (np. alg. centroidów).

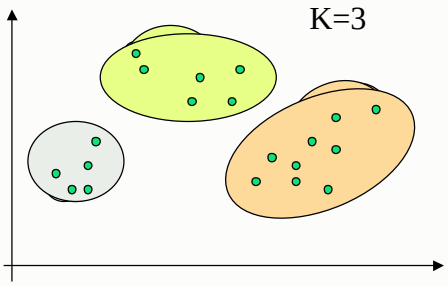
GRUPOWANIE - K-MEANS (PRZYKŁAD ALGORYTMU)

Założenia: mamy podzielić zbiór obiektów na K rozłącznych grup.

1. Znajdujemy K najdalszych punktów i zakładamy tam grupy.

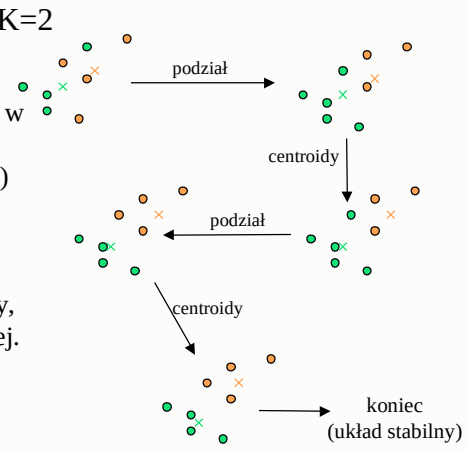
2. Znajdujemy obiekt najbliższy centrum jednej z grup i dołączamy go (**strategia zachłanna**).

3. Powtarzamy czynność 2 do momentu wyczerpania się obiektów.



GRUPOWANIE - ALG. CENTROIDÓW (PRZYKŁAD ALGORYTMU)

- Dzielimy zbiór na K grup w sposób losowy.
- Liczymy środek (centroid) każdej grupy.
- Dokonujemy ponownego podziału obiektów, przypisując je do tej grupy, której środek leży najbliżej.
- Powtarzamy od drugiego kroku póki następują zmiany przyporządkowania.



ALGORYTMY ZRANDOMIZOWANE

Algorytmy, których działanie uzależnione jest od czynników losowych.

• Algorytmy typu **Monte Carlo**: dają (po pewnym czasie) wynik optymalny z prawdopodobieństwem bliskim 1

np. algorytm szukania pokrycia macierzy przez wielokrotny losowy wybór kolejnych kolumn (o ile pokrywają nowe wiersze macierzy)

• Algorytmy typu **Las Vegas**: dają zawsze wynik optymalny, a randomizacja służy jedynie poprawie szybkości działania.

np. zrandomizowany QuickSort

SKĄD WZIĄĆ LICZBY LOSOWE

Źródła „prawdziwej" losowości:

- zegar systemowy
- użytkownik (np. czas naciśnięcia klawisza, ruch myszki)
- przyrządy pomiarowe (np. szum wzmacniacza, licznik Geigera obok próbki promieniotwórczej)

W praktyce te źródła losowości służą do inicjowania ciągów liczb pseudolosowych w generatorach programowych.

Przykład:
generator Marsaglii (1991): $x_n = (x_{n-s} + x_{n-r} + c) \bmod M$,
gdzie c = 1, jeśli poprzednia suma przekroczyła M, c = 0 w p.p.
Np: $x_n = (x_{n-2} + x_{n-21} + c) \bmod 6$, okres: 10^{16}
Np: $x_n = (x_{n-22} - x_{n-43} - c) \bmod 2^{32}-5$, okres: 10^{414}

Generator z niektórych kompilatorów C/C++ (np. GCC):
$x_n = (1103515245\ x_{n-1} + 12345) \bmod 2^{32}$
$rand() = (x_n / 2^{16}) \bmod 2^{15}$

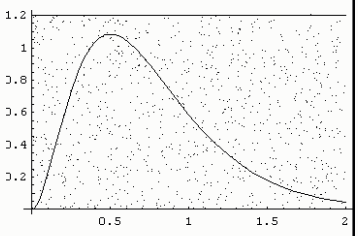
LICZBY LOSOWE O ZADANYM ROZKŁADZIE

- Metoda ogólna: **odwracanie dystrybuanty**.

Mamy wygenerować zmienną losową z rozkładu o znanej gęstości f(x).
Niech F(x) - dystrybuanta tego rozkładu. Wtedy:
wynik = $F^{-1}(u)$
jest szukaną zmienną losową, o ile *u* - wartość losowa z przedziału [0,1).

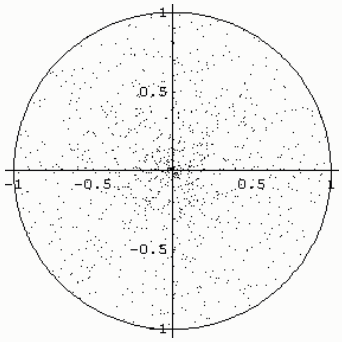
- **Metoda eliminacji**:

Znamy gęstość f(x) na pewnej dziedzinie D, jednak nie znamy F^{-1}. Niech M - ograniczenie górne f(x).



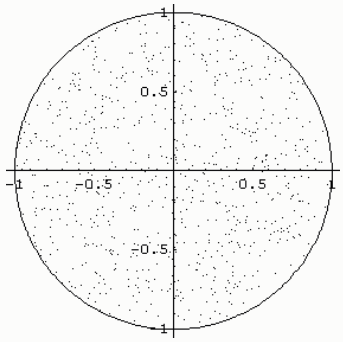

```
repeat
      u1=random(D)
      u2=random(0,M)
until u2<f(u1)
```

PRZYKŁAD - LOSOWANIE PUNKTÓW Z KOŁA



Metoda "biegunowa"

Losujemy współrzędne w układzie biegunowym (kąt i promień) z rozkładu jednostajnego.



Metoda eliminacji

Losujemy punkty z kwadratu i eliminujemy te, które leżą poza kołem.

LAS VEGAS - PRZYKŁADY

RandomQuickSort:

1. Mamy posortować zbiór S. Wybieramy z S losowy x.
2. Dzielimy S na zbiory S_1 - elementów mniejszych, niż x i S_2 - większych.
3. Rekurencyjnie sortujemy S_1 i S_2.

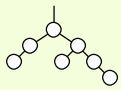
Średnia złożoność jest taka sama, jak w przypadku deterministycznym. Losowanie zapobiega zbyt częstemu pojawianiu się"złośliwych" danych.

Przykład pokrewny: budowa drzewa binarnego

Przed budową drzewa mieszamy losowo dane wejściowe.

Przykład: **ciąg liczb rosnących**

zbudowane drzewo



drzewo zbudowane po losowym przemieszaniu ciągu

MONTE CARLO

- Metody, które dają z pewnym prawdopodobieństwem dobry wynik
- Prawdopodobieństwo można zwiększać powtarzając obliczenia wielokrotnie

  (tej cechy nie mają alg. deterministyczne)
- Algorytmy numeryczne
  - całkowanie

    (liczymy średnią wartość funkcji w losowych punktach)
  - sprawdzanie, czy AB=C dla macierzy A,B,C

    (zamiast mnożyć A i B, co jest kosztowne, losujemy wektor *x* i badamy, czy A(Bx) = Cx, dla wielu różnych x)

PRZESZUKIWANIE LOSOWE

Najprostsza metoda: z przestrzeni stanów *S* losujemy wiele obiektów *x* i wybieramy ten, dla którego *f(x)* ma wartość największą.

Metoda bardziej złożona: po wylosowaniu np. 1000 obiektów zapamiętujemy 100 najlepszych i kolejnych losowań dokonujemy z "sąsiedztwa" tych 100.

(Przykład - problem komiwojażera: losujemy 1000 dróg, wybieramy 100 najkrótszych, następnie losowo je modyfikujemy, otrzymując kolejne 1000 przykładów itd.)

Metoda hybrydowa: wybieramy 100 najlepszych z 1000 losowych, następnie dla każdego z nich uruchamiamy algorytm wspinaczki.

Metody (często) oparte na sąsiedztwie.