Algorytmy ewolucyjne w praktyce

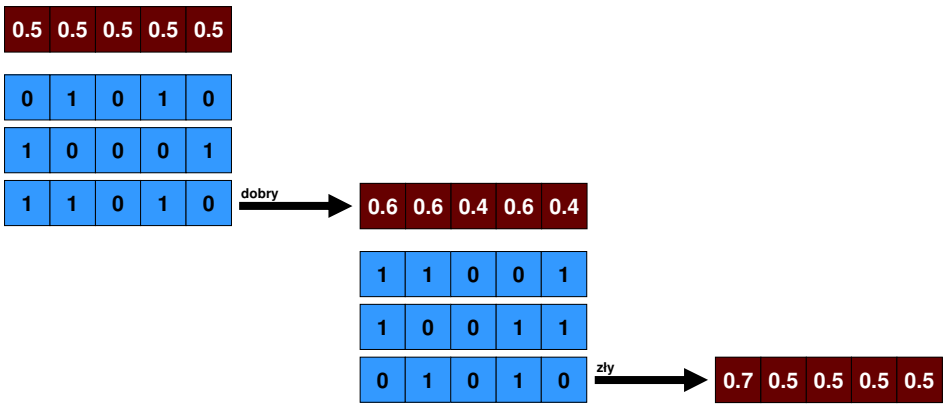
algorytmy Estimation of Distribution Algorithms oparte na rozkładach prawdopodobieństwa wektora niezależnych zmiennych losowych

Piotr Lipiński

Wprowadzenie

- Rozważane dotąd podstawowe algorytmy ewolucyjne nie starały się zdobywać żadnej wiedzy o problemie, a skupiały się jedynie na przeszukiwaniu przestrzeni poszukiwań.
  - Czy wylosowane już elementy przestrzeni poszukiwań nie mogłyby dostarczyć nam jakiejś wiedzy, którą można byłoby wykorzystać przy losowaniu następnych?
  - Czy losowanie z jednostajnym rozkładem prawdopodobieństwa jest najlepszym pomysłem?
- Przykład:
  - Rozpatrzmy przestrzeń poszukiwań $\Omega = \{0, 1\}^d$ i funkcję celu F określoną na niej.
  - Początkowo, element przestrzeni poszukiwań jest losowany z jednostajnym rozkładem prawdopodobieństwa (tzn. prawdopodobieństwo wylosowania każdego wektora binarnego o długości d jest takie samo i wynosi $1/2^d$). Inaczej mówiąc, wektor binarny jest losowany bit po bicie, a dla każdego bitu, prawdopodobieństwo wylosowania 0 i 1 jest takie samo i wynosi 1/2.
  - W kolejnych iteracjach, prawdopodobieństwa wylosowania 0 i 1 dla różnych bitów zmieniają się na podstawie wiedzy uzyskanej z wcześniej wylosowanych elementów:
    - jeśli wylosowany element był dobry (np. lepszy od dotychczas najlepszego) i jeśli miał 1 na pewnym miejscu, to prawdopodobieństwo wylosowania 1 na tym miejscu zostaje zwiększone, a jeśli miał 0 na pewnym miejscu, to prawdopodobieństwo wylosowania 1 na tym miejscu zostaje zmniejszone,
    - jeśli wylosowany element był zły (np. gorszy od dotychczas najgorszego), to odwrotnie.
- Naturalny więc wydaje się pomysł konstruowania rozkładu prawdopodobieństwa, który pozwalałby na losowanie dobrych elementów przestrzeni poszukiwań.

Wprowadzenie



UWAGA: Oczywiście przykład jest bardzo naiwny, w praktyce takie podejście może być bardzo nieefektywne, bo może prowadzić do przedwczesnej zbieżności algorytmu ewolucyjnego. Lepiej byłoby zastosować bardziej delikatne modyfikacje wektora prawdopodobieństw.

Estimation of Distribution Algorithms

- Estimation of Distribution Algorithms (EDA) to rodzina algorytmów ewolucyjnych, które starają się modelować rozkład prawdopodobieństwa opisujący dobre rozwiązania rozpatrywanego problemu optymalizacji.
- Główne cechy EDA:
  - istotne jest nie tyle znalezienie rozwiązania problemu optymalizacji (punktu przestrzeni poszukiwań), co skonstruowanie rozkładu prawdopodobieństwa umożliwiającego generowanie przybliżonych rozwiązań problemu optymalizacji,
  - wyznaczony rozkład prawdopodobieństwa to pewna wiedza o rozpatrywanym problemie, często uniwersalna, którą można ponownie wykorzystać przy rozwiązywaniu podobnych problemów optymalizacji,
  - konstruowanie modelu probabilistycznego w bardziej zaawansowanych algorytmach EDA jest bardzo czasochłonne.
- Dlaczego lepiej jest dostać rozkład prawdopodobieństwa niż konkretny punkt przestrzeni poszukiwań?

Estimation of Distribution Algorithms

- Proste algorytmy EDA oparte na rozkładach prawdopodobieństwa wektora niezależnych zmiennych losowych:
  - dla dyskretnych problemów optymalizacji
    - UMDA
    - PBIL
    - CGA
  - dla ciągłych problemów optymalizacji
    - UMDAc
    - SHCLVND
    - PBILc
- Bardziej zaawansowane algorytmy EDA oparte na rozkładach prawdopodobieństwa wektora skorelowanych zmiennych losowych:
  - dla dyskretnych problemów optymalizacji
    - MIMIC, COMIT, EBNA, BOA, hBOA
  - dla ciągłych problemów optymalizacji
    - EGNA, modyfikacje powyższych

EDA dla dyskretnych problemów optymalizacji

- W tej części wykładu będziemy rozpatrywać dyskretne problemy optymalizacji w których funkcja celu F określona jest na przestrzeni poszukiwań $\Omega = \{0, 1\}^d$.
- Zakładamy niezależność zmiennych losowych w wektorze prawdopodobieństwa modelującym wartości chromosomów.
- Rozkład prawdopodobieństwa można więc opisać przez wektor

$$\mathbf{p} = (p_1, p_2, \ldots, p_d),$$

gdzie p_i określa prawdopodobieństwo wystąpienia 1 na i-tej pozycji chromosomu.

Univariate Marginal Distribution Algorithm (UMDA)

- *Univariate Marginal Distribution Algorithm* (UMDA) jest jednym z najprostszych algorytmów EDA dla dyskretnych problemów optymalizacji.

- UMDA korzysta z prostej estymacji prawdopodobieństw **p** na podstawie próbki danych $\mathbf{P}^{(S)}$ zawierającej najlepsze osobniki z aktualnej populacji **P**:
  - dla każdego i = 1, 2, …, d,

    $$p_i = p(x_i \mid \mathbf{P}^{(S)}),$$

    gdzie x_i to i-ta współrzędna osobnika **x**,
  - zakładając, że populacja $\mathbf{P}^{(S)}$ składa się z osobników $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_M$, to

    $$p_i = \frac{1}{M}\sum_{j=1}^{M} x_{ij}$$

    gdzie x_{ij} to i-ta współrzędna j-tego osobnika w populacji $\mathbf{P}^{(S)}$.

Univariate Marginal Distribution Algorithm (UMDA)

- Algorytm UMDA

```
p = (0.5, 0.5, …, 0.5);
P = Random-Population(p, N);
Population-Evaluation(P, F);
while not Termination-Condition()
    P(S) = Select(P, N, M);
    p = Model-Estimation(P(S))
    P = Random-Population(p, N);
    Population-Evaluation(P, F);
return best of P;
```

- Znaczenie parametrów:
  - F – funkcja celu
  - N – liczba osobników w populacji głównej **P**
  - M – liczba osobników wybieranych z populacji głównej **P** do populacji $\mathbf{P}^{(S)}$ (wielkość próbki)

Univariate Marginal Distribution Algorithm (UMDA)

- Słabą stroną algorytmu UMDA jest każdorazowe wyliczanie wektora prawdopodobieństw **p** od początku, na podstawie nowej próbki pochodzącej z aktualnej populacji, bez wykorzystywania wartości estymowanych w poprzednich iteracjach.
  - pojawienie się w populacji outlierów powoduje duże wypaczenie estymowanego modelu probabilistycznego, zwłaszcza w przypadku niewielkich populacji,
  - stabilne działanie algorytmu UMDA wymaga więc dość dużych populacji,
  - wady tej nie ma algorytm *Population-Based Incremental Learning* (PBIL), w którym model probabilistyczny jest stopniowo dostosowywany do aktualnej populacji, a nie całkowicie zmieniany.

Population-Based Incremental Learning (PBIL)

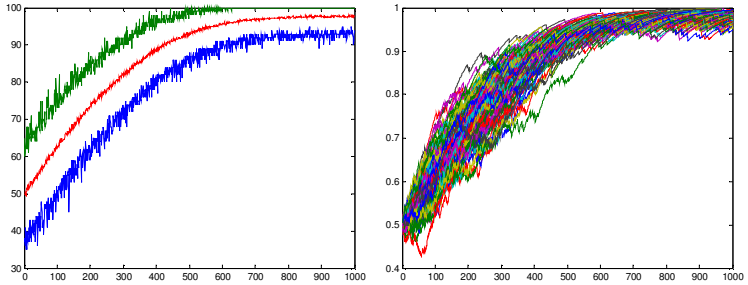
- Algorytm PBIL

```
p = (0.5, 0.5, …, 0.5);
P = Random-Population(p, N);
Population-Evaluation(P, F);
while not Termination-Condition()
    x_i = Best-Individual(P);
    for k = 1 to d do
        p_k = p_k · (1 − θ_1) + x_ik · θ_1;
    for k = 1 to d do
        if Uniform-Random(0, 1) < θ_2 then
            p_k = p_k · (1 − θ_3) + Binary-Random(0.5) · θ_3;
    P = Random-Population(N, p);
    Population-Evaluation(P, F);
return best of P;
```

- Znaczenie parametrów:
  - F – funkcja celu
  - N – liczba osobników w populacji głównej **P**
  - d – długość chromosomu
  - θ_1 – parametr algorytmu, współczynnik uczenia
  - θ_2 – parametr algorytmu, prawdopodobieństwo mutacji
  - θ_3 – parametr algorytmu, współczynnik zaburzenia podczas mutacji

Ilustracja działania PBIL

- PBIL na OneMax o długości d = 100
  - Przestrzeń poszukiwań to wektory binarne długości d = 100.
  - Funkcja celu OneMax zwraca liczbę jedynek w ciągu binarnym. Jej maksimum to wektor samych jedynek.
  - T = 1000 iteracji
  - N = 125 osobników
  - $\theta_1 = 0.01$, $\theta_2 = 0.05$, $\theta_3 = 0.01$
- Algorytm znalazł rozwiązanie problemu po ok. 500 iteracjach.
- Algorytm skonstruował w miarę dobry model probabilistyczny po ok. 800 iteracjach.
- UWAGA: W końcowym modelu probabilistycznym nie wszystkie prawdopodobieństwa są równe 1. Czy to dobrze czy źle?

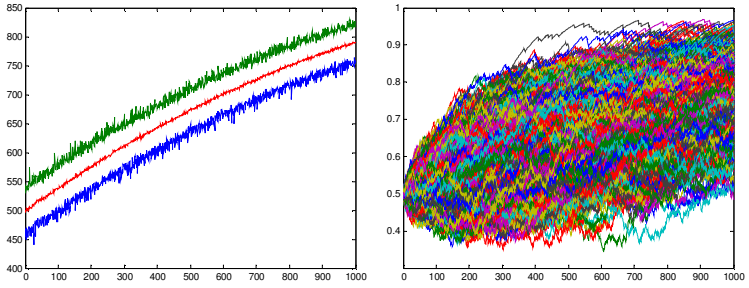


Ilustracja działania PBIL

- Rozpatrywać będziemy uproszczony problem klasyfikacji wielospektralnych obrazów satelitarnych:
  - interesują nas tylko 3 klasy punktów, które chcemy rozpoznawać na obrazie (pozostałe punkty nas nie interesują),
  - dostępne są utworzone wcześniej reguły klasyfikujące,
  - algorytmu PBIL użyjemy do wyboru optymalnego zestawu reguł klasyfikujących.
- Dany jest obraz wielospektralny o 3 spektrach i rozmiarze 106 x 148 punktów.
  - Każdy punkt obrazu można przedstawić jako wektor rozmiaru 3 x 1, a więc cały obraz można przedstawić jako macierz rozmiaru 3 x 15688.
- Dana jest wzorcowa klasyfikacja tego obrazu.
  - Jest to bitmapa o 11 kolorach i rozmiarze 106 x 148 punktów.
  - Można przedstawić ją jako wektor rozmiaru 1 x 15688 o wartościach ze zbioru {1, 2, …, 11}.
- Interesują nas tylko 3 klasy punktów, pozostałe punkty więc usuwamy z danych, otrzymując:
  - macierz X rozmiaru 3 x 9350 (punkty obrazu wielospektralnego),
  - wektor C rozmiaru 1 x 9350 o wartościach ze zbioru {1, 2, 3} (wzorcowa klasyfikacja).
- Dostępnych jest 266 utworzonych wcześniej reguł klasyfikujących.
  - Każda reguła to określona funkcja, która na wejściu dostaje punkt obrazu (wektor rozmiaru 3 x 1), a na wyjściu zwraca etykietę klasy (liczbę ze zbioru {1, 2, 3}).
  - Reguły te zostały utworzone algorytmem programowania genetycznego (będziemy o nim mówić w dalszej części wykładu).
  - Dla wygody dostępne są już policzone wartości każdej reguły klasyfikującej dla każdego punktu obrazu, a nie same definicje tych reguł.
- Dla danego punktu obrazu, każda reguła zwraca proponowaną etykietę klasy. Ostateczna decyzja o klasie punktu może zostać podjęta "większością głosów" wszystkich reguł klasyfikujących.
- Czy warto jednak słuchać wszystkich reguł klasyfikujących? Może wybrać podzbiór zbioru wszystkich reguł i słuchać tylko jego? Jak znaleźć optymalny zestaw reguł klasyfikujących?

Ilustracja działania PBIL

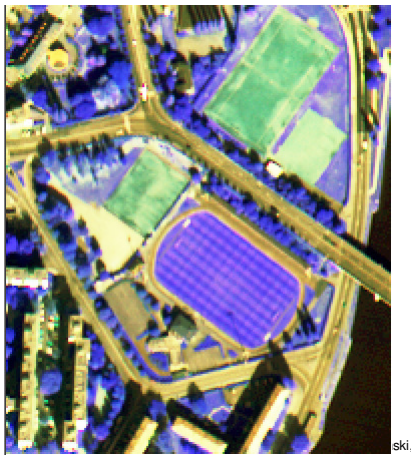
- PBIL na OneMax o długości d = 1000
  - Przestrzeń poszukiwań to wektory binarne długości d = 1000.
  - Funkcja celu OneMax zwraca liczbę jedynek w ciągu binarnym. Jej maksimum to wektor samych jedynek.
  - T = 1000 iteracji
  - N = 125 osobników
  - $\theta_1 = 0.01$, $\theta_2 = 0.05$, $\theta_3 = 0.01$
- Algorytm nie znalazł rozwiązania problemu (potrzeba więcej iteracji lub zmiany parametrów uczenia).
- Algorytm nie skonstruował też właściwego modelu probabilistycznego. Dlaczego?



Przykład – klasyfikacja wielospektralnych obrazów satelitarnych

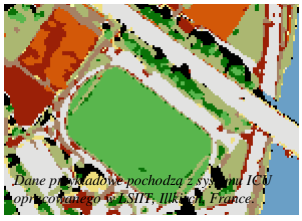
Wielospektralny obraz satelitarny:

- obraz wielospektralny składa się z pewnej liczby n warstw (zazwyczaj n jest duże, np. $n = 115$) (analogicznie jak obraz RGB składa się z 3 warstw: R, G i B)
- każdy punkt obrazu q jest więc opisany przez n parametrów: $q_1, q_2, \ldots, q_n$ (parametry te mogą być liczbami jednobajtowymi lub ogólniej liczbami rzeczywistymi)



Klasyfikacja:

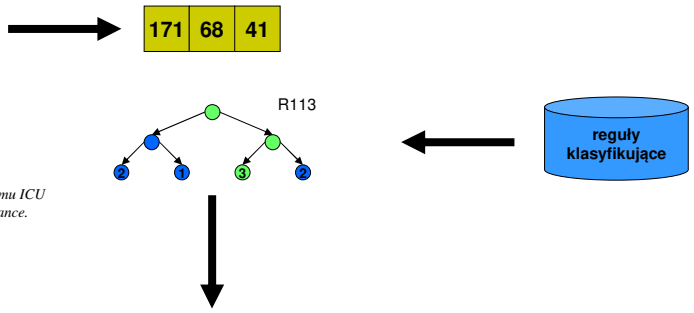
- na obrazie chcemy rozpoznawać pewne klasy punktów, na przykład odpowiadające budynkom, drogom, trawnikom, zbiornikom wodnym, itp.
- pewien fragment obrazu został wcześniej wzorcowo sklasyfikowany, możemy użyć go do utworzenia/nauczenia systemu automatycznej klasyfikacji
- niech K oznacza liczbę rozważanych klas, zaś C1, C2, …, CK będą etykietami tych klas



Dane przykładowe pochodzą z systemu ICU opracowanego w LSIIT, Illkirch, France.

Przykład – klasyfikacja wielospektralnych obrazów satelitarnych

Dane przykładowe pochodzą z systemu ICU opracowanego w LSIIT, Illkirch, France.



reguła klasyfikująca nr R113 uznała, że punkt obrazu jest klasy C3

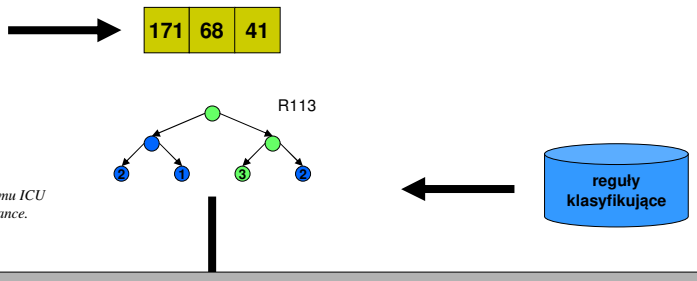
wyniki pozostałych reguł:

3	1	2	1	2	2	3	3	1	3	3	1	3	2	3	3	3	1	…	3

ostateczna decyzja o klasie punktu może być podjęta "większością głosów" wszystkich reguł klasyfikujących

Przykład – klasyfikacja wielospektralnych obrazów satelitarnych

Dane przykładowe pochodzą z systemu ICU opracowanego w LSIIT, Illkirch, France.



wyniki pozostałych reguł:

3	1	2	1	2	2	3	3	1	3	3	1	3	2	3	3	3	1	…	3

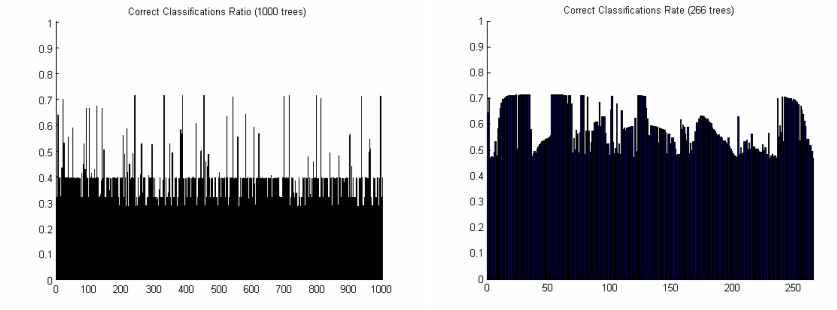
ostateczna decyzja o klasie punktu może być podjęta "większością głosów" wszystkich reguł klasyfikujących

jeśli rozpatrywalibyśmy jedynie pewien podzbiór zbioru wszystkich reguł klasyfikujących, decyzja mogłaby być inna

3	1	2	1	2	2	3	3	1	3	3	1	3	2	3	3	3	1	…	3

Ilustracja działania PBIL

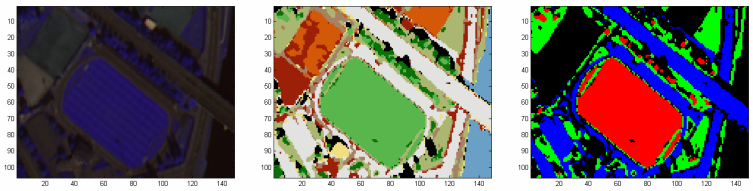
- Decyzja oparta na pojedynczej regule klasyfikującej daje dość słabe wyniki
  - w najlepszych przypadkach poprawnie poklasyfikowanych jest ok. 70% punktów obrazu



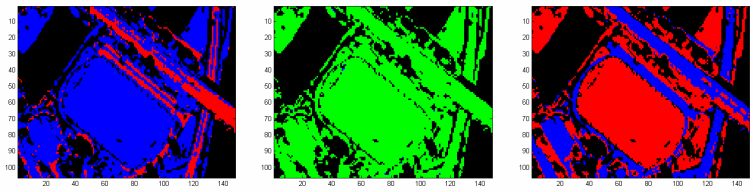
- Decyzja oparta na wszystkich 266 regułach klasyfikujących ("większością głosów") daje tylko 64.53% poprawnie poklasyfikowanych punktów obrazu.
- Znajdując odpowiedni podzbiór reguł klasyfikujących i opierając na nim decyzję można uzyskać nawet ponad 98% poprawnie poklasyfikowanych punktów obrazu.

Ilustracja działania PBIL

- Dane wejściowe: oryginalny obraz wielospektralny, oryginalna klasyfikacja wzorcowa, uproszczona klasyfikacja wzorcowa (tylko 3 klasy punktów).

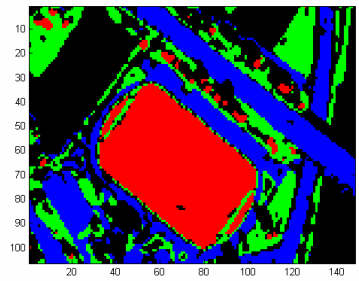


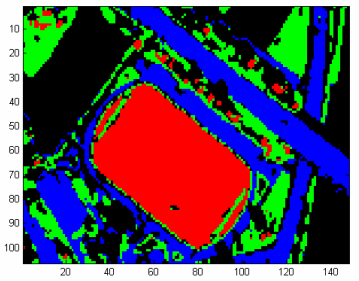
- Przykłady obrazów uzyskiwanych w klasyfikacji opartej na pojedynczej regule klasyfikującej.



Ilustracja działania PBIL

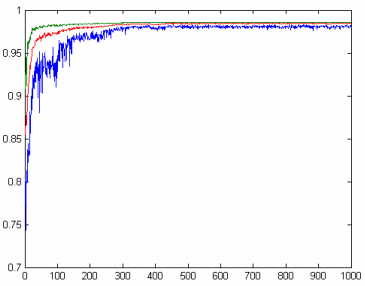
- Po lewej: klasyfikacja wzorcowa.
- Po prawej: obraz uzyskany w klasyfikacji opartej na zbiorze reguł klasyfikujących skonstruowanym algorytmem PBIL (ponad 98% poprawnie poklasyfikowanych punktów obrazu).

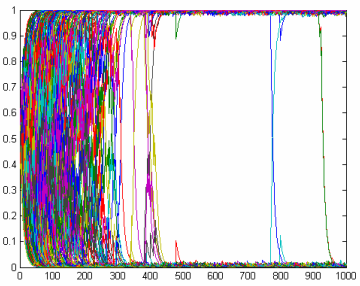




Ilustracja działania PBIL

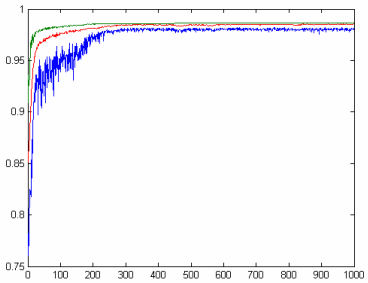
- PBIL
  - Przestrzeń poszukiwań to wektory binarne długości d = 266.
  - Funkcja celu zwraca frakcję poprawnie poklasyfikowanych punktów (zakres wartości od 0.00 do 1.00).
  - T = 1000 iteracji
  - N = 50 osobników
  - czas obliczeń: ok. 2h
  - najlepsze znalezione rozwiązanie: 0.985775

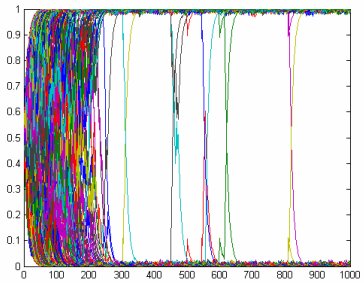




Ilustracja działania PBIL

- PBIL
  - Przestrzeń poszukiwań to wektory binarne długości d = 266.
  - Funkcja celu zwraca frakcję poprawnie poklasyfikowanych punktów (zakres wartości od 0.00 do 1.00).
  - T = 1000 iteracji
  - N = 100 osobników
  - czas obliczeń: ok. 4h
  - najlepsze znalezione rozwiązanie: 0.986631





Compact Genetic Algorithm (CGA)

- Algorytm CGA

```
p = (0.5, 0.5, …, 0.5);
x1 = Random-Individual(p);
x2 = Random-Individual(p);
Individual-Evaluation(x1, F);
Individual-Evaluation(x2, F);
while not Termination-Condition()
    x1 = Best-Individual(x1, x2);
    xj = Worst-Individual(x1, x2);
    for k = 1 to d do
        if xik = 1 and xjk = 0 then
            pk = pk + θ
        if xik = 0 and xjk = 1 then
            pk = pk - θ
x1 = Random-Individual(p);
x2 = Random-Individual(p);
Individual-Evaluation(x1, F);
Individual-Evaluation(x2, F);
return best of x1 and x2;
```

- Znaczenie parametrów:
  - F – funkcja celu
  - d – długość chromosomu
  - θ – parametr algorytmu, współczynnik uczenia,
    - zazwyczaj θ = 1/N, gdzie N odpowiada wielkości populacji w SGA

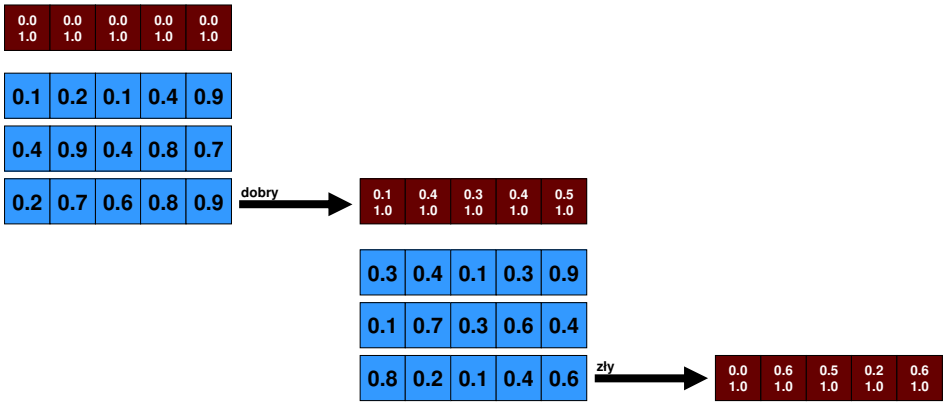
EDA dla ciągłych problemów optymalizacji

- W tej części wykładu będziemy rozpatrywać ciągłe problemy optymalizacji, w których funkcja celu F określona jest na przestrzeni poszukiwań $\Omega = R^d$.
- Zakładamy niezależność zmiennych losowych w wektorze prawdopodobieństwa modelującym chromosom.
- Standardowe podejście polega na modelowaniu rozkładu prawdopodobieństwa przy użyciu rozkładu gaussowskiego, tzn. wartości poszczególnych genów x_i są losowane z rozkładem $N(\mu_i, \sigma_i)$, dla i = 1, 2, …, d.
- Rozkład prawdopodobieństwa można więc opisać przez dwa wektory

$$\boldsymbol{\mu} = (\mu_1, \mu_2, \ldots, \mu_d),$$
$$\boldsymbol{\sigma} = (\sigma_1, \sigma_2, \ldots, \sigma_d),$$

gdzie μ_i określa wartość oczekiwaną używanego rozkładu normalnego, a σ_i jego odchylenie standardowe.

EDA dla ciągłych problemów optymalizacji



UMDAc

- Najprostszy algorytm EDA dla ciągłych problemów optymalizacji to UMDAc – rozszerzenie algorytmu UDMA dla zagadnień dyskretnych.

- UMDAc korzysta z prostej estymacji parametrów rozkładów prawdopodobieństw $N(\mu_i, \sigma_i)$ na podstawie próbki danych $\mathbf{P}^{(S)}$ zawierającej najlepsze osobniki z aktualnej populacji $\mathbf{P}$:
  - estymator wartości oczekiwanej

$$\mu_i = \frac{1}{M}\sum_{j=1}^{M} x_{ij}$$

  - estymator odchylenia standardowego

$$\sigma_i^2 = \frac{1}{M}\sum_{j=1}^{M} (x_{ij} - \mu_i)^2$$

  - początkowe wartości parametrów rozkładu są ustalane na 0 i 1.

UMDAc

- Algorytm UMDAc

```
(μ, σ) = Model-Initialization();
P = Random-Population(μ, σ, N);
Population-Evaluation(P, F);
while not Termination-Condition()
    P(S) = Select(P, N, M);
    (μ, σ) = Model-Estimation(P(S))
    P = Random-Population(p, N);
    Population-Evaluation(P, F);
return best of P;
```

- Znaczenie parametrów:
  - F – funkcja celu
  - N – liczba osobników w populacji głównej $\mathbf{P}$
  - M – liczba osobników wybieranych z populacji głównej $\mathbf{P}$ do populacji $\mathbf{P}^{(S)}$ (wielkość próbki)

UMDAc

- Algorytm UMDAc można łatwo zaadaptować do innych niż gaussowski rozkładów prawdopodobieństwa.
- Estymacja modelu probabilistycznego powinna wówczas działać następująco:
  - stosując testowanie hipotez statystycznych, określ rozkłady prawdopodobieństwa $f(x_1, \theta_1)$, $f(x_2, \theta_2)$, …, $f(x_d, \theta_d)$ opisujące kolejne geny osobników z próbki danych $\mathbf{P}^{(S)}$,
  - metodą największej wiarygodności, estymuj parametry $\theta_1, \theta_2, \ldots, \theta_d$ wybranych rozkładów prawdopodobieństwa na podstawie próbki danych $\mathbf{P}^{(S)}$,
  - łączny rozkład prawdopodobieństwa modelujący osobników będzie miał postać:

$$f(\mathbf{x}, \boldsymbol{\theta}) = \prod_{i=1}^{d} f(x_i, \theta_i)$$

SHCLVND

- Stochastic Hill Climbing with Learning by Vectors of Normal Distributions (SHCLVND) to algorytm modyfikujący parametry rozkładów prawdopodobieństwa stosując regułę Hebba.
  - adaptacja parametrów μ_i przebiega następująco:

$$\mu_i = \mu_i + \alpha \cdot (b_i - \mu_i)$$

gdzie **b** to środek ciężkości ustalonej liczby najlepszych osobników w bieżącej populacji,

  - adaptacja parametrów σ_i przebiega następująco:

$$\sigma_i = \beta \cdot \sigma_i$$

gdzie $0 < \beta < 1$ to stała będąca parametrem algorytmu.

PBILc

- Analogicznie można rozszerzyć algorytm PBIL do PBILc.
  - estymacja parametrów μ_i jest prosta,

$$\mu_i = (1-\alpha)\cdot\mu_i + \alpha\cdot(x_{\text{best1},i} + x_{\text{best2},i} - x_{\text{worst},i})$$

    gdzie $\mathbf{x}_{\text{best1}}$, $_{\text{best1best2}}$ to dwa najlepsze osobniki w bieżącej populacji, zaś $\mathbf{x}_{\text{worst}}$ to najgorszy osobnik w bieżącej populacji,

  - estymacja parametrów σ_i wymaga większej uwagi,
    - można zastosować stopniowe zmniejszanie parametru σ_i przez mnożenie przez stałą (jak w SHCLVND),
    - można zastosować estymację analogiczną do estymacji parametrów μ_i, tzn. stopniowo korygować parametry w zależności od parametrów estymowanych z populacji,
    - można zastosować regułę 1/5 sukcesów Rechenberga ze strategii ewolucyjnych,
    - częstym problemem jest przedwczesna zbieżność algorytmu, która występuje w przypadku radykalnego zmniejszenia się parametrów σ_i.