Wilno, dnia ……………….. 20..… r.

……………………………………………………..

Imię i nazwisko studenta/studentki

……………………………………………………..

Nr indeksu

……………………………………………..............

Kierunek i forma studiów (stacjonarne/niestacjonarne)

…………………………………………….............

Imię i nazwisko Promotora pracy dyplomowej

**O**ś**wiadczenie**

Przedkładając w roku akademickim 20..… /20..…. Promotorowi pracę dyplomową pod

tytułem

………………………………………………………………………………………………...

………………………………………………………………………………………………...

niniejszym oświadczam iż:

1. praca dyplomowa została napisana przeze mnie samodzielnie,

2. w swojej pracy korzystałem/am z materiałów źródłowych wymieniając autora, tytuł

pozycji i źródło jej publikacji,

3. zamieszczałem/am krótkie fragmenty prac innych autorów w cudzysłowiu, a w

przypisie/bibliografii podawałem/am źródło tego cytatu,

4. praca nie zawiera żadnych danych, informacji i materiałów, których publikacja nie jest prawnie dozwolona,

5. treśćpracy dyplomowej dostarczonej do dziekanatu w wersji elektronicznej pokrywa się z treścią pracy złożonej w formie pisemnej.

*Składaj*ą*c niniejsze o*ś*wiadczenie jestem* ś*wiadomy/a,* ż*e:*

- przywłaszczenie sobie autorstwa lub wprowadzenie w błąd co do autorstwa całości lub części cudzego utworu jest przestępstwem - zagrożonym odpowiedzialnością karną, na podstawie ustawy z dnia 4 lutego 1994 roku o prawie autorskim i prawach

pokrewnych (tekst jednolity Dz. U. Nr 80 z 2000 roku poz. 904 z pózn. zm. )

- w przypadku ujawnienia naruszenia ustawy o prawie autorskim i prawach pokrewnych praca dyplomowa może być unieważniona, nawet po przeprowadzeniu obrony.

……………………………………

Podpis czytelny studenta/studentki

Pusta strona w przypadku obustronnego drukowania

UNIWERSYTET W BIAŁYMSTOKU

WYDZIAŁ EKONOMICZNO – INFORMATYCZNY

KATEDRA INFORMATYKI

EDGAR NOVOGRODSKI

Nr albumu: 71253

WYBRANE METODY OPTYMALIZACJI GLOBALNEJ: STUDIUM PORÓWNAWCZE

Praca licencjacka

napisana pod kierunkiem

prof. dr hab. Władysława Homendy

Wilno 2017

Pusta strona w przypadku obustronnego drukowania

SPIS TREŚCI

WSTĘP 4

ROZDZIAŁ 1. OPTYMALIZACJA 5

1.1 Czym jest optymalizacja? 5

rozwinnąć…

ROZDZIAL 2. OPIS ALGORYTMÓW

ROZDZIAŁ 3. PROCES PROGRAMOWANIA 9

2.1 Strona oraz wybór podstrony 9

2.2 Strona główna(Aktualności) 10

2.3 Galerie 11

2.3 Pozostałe działy strony 12

2.3 Oprawa graficzna 13

ROZDZIAŁ 4. TESTOWANIE 14

3.1 Wygląd strony 14

ZAKOŃCZENIE 17

SPIS RYSUNKÓW 18

BIBLIOGRAFIA 19

ŹRÓDŁA INTERNETOWE 20

WSTĘP

W dzisiejszych czasach coraz częściej słyszymy o stucznej inteligencji. Inteligentne samochody, inteligentne domy oraz mnóstwo innych inteligentnych rzeczy

**Celem pracy jest** przybliżenie procesu wykorzystania algorytmów służących do optymalizacji wybranej funkcji, oraz zaprezentowanie i porównanie ich efektywności.

Praca składa się z trzech rozdziałów.

**W pierwszym rozdziale** będą przedstawione algorytmy optymalizacyjne oraz zasady ich działania.

**Rozdział drugi** zostanie poświęcony procesowi oprogramowania algorytmów

**W ostatnim rozdziale** zademonstrowane zostaną wyniki wykorzystanych algorytmów, oraz porównanie ich skuteczności.

Optymalizacja, ALGORYTMY METAHEURYSTYCZNE

Czym jest optymalizacja?

Podczas rozwiązywania różnych problemów optymalizacyjnych mamy kandydatów na rozwiązanie problemu. Jednak nie możemy określić jednoznacznie czy kandydat jest prawidłowym, czy nieprawidłowym rozwiązaniem. Możemy natomiast określić jakość tego rozwiązania, a służą ku temu funkcje oceniające. Istnieją przypadki, gdy naszym zadaniem jest zmaksymalizowanie funkcji oceniającej, czyli kandydat z oceną największą jest najlepszym rozwiązaniem. W naszym przypadku celem jest zminimalizowanie funkcji, znaczy to, że kandydat z najniższą oceną będzie najlepszym kandydatem.

Formalna definicja dla optymalizacji wygląda następująco:

Załóżmy, że ***X*** jest zbiorem kandydatów do rozwiązania problemu optymalizacyjnego. Zazwyczaj ***X*** należy do jakiejś dziedziny *n*-wymiarowej dla przykładu ***X***⊆ *Nn*. Dziedzina ***X*** jest często nazywana przestrzenią wyszukiwania (ang. search-space). Niech problem optymalizacyjny będzie określony funkcją ***f,*** jest ona także nazywana funkcją oceniającą (ang. fitness function) i pokazuje, jak dobre jest rozwiązanie kandydata ze zbioru X dla danego problemu optymalizacyjnego:

*f* : *X* → R

Ponieważ naszym zadaniem jest zminimalizowanie funkcji oceniającej ***f***  musimy znaleźć takie rozwiązanie ***x*** ∈ ***X*** aby spełnić warunek:

∀***y*** ∈ ***X* : *f(x)* ≤ *f(y)***

Takie rozwiązanie ***x*** jest nazywane minimum globalnym dla funkcji ***f.*** Jednak nie możemy określić czy rzeczywiście jest to minimum globalne, dlatego też rozwiązania z wystarczająco dobrymi wynikami są określane jako możliwe do zaakceptowania.

Proces optymalizowania posiada także kilka problemów. Pierwszym z nich jest charakter zadania optymalizacyjnego, a mianowicie to, że może ono posiadać kilka optimum lokalnych, w których może zostać uwięzione nasze rozwiązanie. Mogą występować także przypadki w których kandydaci rozwiązania będą dawali różne wyniki za każdym razem. Kolejnym problemem w optymalizowaniu jest wiedza lub też niewiedza na temat problemu. Jeżeli wiemy bardzo mało na temat tego co chcemy zoptymalizować to nasze wyniki będą prawidłowe tylko dla pewnej klasy problemów

OPISY Algorytmów

Problem plecakowy

Do przedstawienia zasad działania algorytmów zostanie wykorzystany problem plecakowy (ang. Knapsack problem). Problem polega na tym, że mamy zbiór elementów o różnej wielkości i wartości oraz plecak o ograniczonej pojemności. Naszym zadaniem jest wybranie elementów, dla których suma wartości będzie największa, a których suma wielkości nie przekroczy pojemności plecaka. Istnieje kilka wariacji tego problemu, natomiast w naszym przypadku zostanie rozpatrzony Dyskretny problem plecakowy (ang. 0-1 knapsack problem).

Definicja formalna dla danego problemu wygląda następująco: mamy plecak o maksymalnej pojemności ***V*** oraz zbiór ***N*** elementowy *{x1, x2, xi,…, xN}*, każdy z elementów ma określoną wartość ***pi*** oraz wielkość ***wi***. Naszym zadaniem jest zmaksymalizowanie funkcji oceniającej.

Zakładając przy tym, że

Do demonstracji działania algorytmów zostały wybrane następujące wartości:

***V=9;***

***N=7;***

***P={6, 5, 8, 9, 6, 7, 3};***

***W={2, 3, 6, 7, 5, 9, 4};***

Algorytm genetyczny

**Algorytm genetyczny (ang. Genetic Algorithm, GA)** – rodzaj algorytmu metaheurystycznego zaliczany do algorytmów ewolucyjnych. Stworzony w latach 70 XX wieku, przypominający swoim działaniem proces ewolucji biologicznej. Twórcą algorytmu jest amerykański naukowiec John Henry Holland, który inspiracje do swoich prac czerpał właśnie z biologii.

Zasada działania algorytmu:

1. Problem definiuje środowisko.
2. Środowisko zawiera pewną populację początkową (zbiór możliwych rozwiązań).
3. Każdy z osobników populacji posiada pewien zbiór informacji, z którego składa się jego genotyp.
4. Istnieje funkcja oceniająca, która ocenia każdego z osobników i pokazuje jak dobre jest rozwiązanie.
5. Najlepsi przedstawiciele generacji są wybierani do procesu reprodukcji.
6. Osobnicy są ze sobą lączeni poprzez proces krzyżowania.
7. Przeprowadzana jest mutacja.
8. W ten sposób powstaje pokolenie potomków.
9. Pokolenie przodków i potomków jest łączone w jedno według ocen z funkcji oceniającej. Najlepsi przeżywają, najsłabsi giną.
10. Proces jest powtarzany aż do momentu gdy zostanie osiągnięty potomek z najlepszą oceną lub przez wskazaną ilość iteracji.

Ewolucja różniczkowa

**Ewolucja różniczkowa** **(ang. Differential Evolution, (DE))** – kolejny algorytm ewolucyjny wprowadzony w roku 1996. Swoją podstawą przypomina algorytm genetyczny. Posiada takie elementy jak łączenie poprzez krzyżowanie, a także proces mutacji tak jak w przypadku GA.

Zasada działania algorytmu:

1. **Problem definiuje środowisko.**
2. **Problem zawiera pewną populację początkową (zbiór możliwych rozwiązań).**

*W przypadku problemu plecakowego są to zbiory N elementowe składające się z 0 i 1. Populacja początkowa jest generowana losowo. Przykładowo:*

*{0, 1, 0, 0, 1, 0, 0}*

*{0, 1, 0, 0, 0, 1, 1}*

*…*

*{0, 0, 1, 1, 1, 0, 0}*

*{1, 1, 1, 0, 0, 0, 1}*

1. **Dla każdego osobnika *I* z populacji są wybierani 3 różni od siebie osobnicy *A, B, C* z tej samej populacji, dlatego też wielkość populacji musi być większa niż 3.**

*Przykładowo:*

*I = {0, 1, 0, 0, 1, 0, 0}*

*A = {0, 1, 0, 0, 0, 1, 1}*

*B = {0, 0, 1, 1, 1, 0, 0}*

*C = {1, 1, 1, 0, 0, 0, 1}*

1. **Stosowany jest wzór do stworzenia nowego osobnika „dawcy” *D = A + F(B – C)***

*F ∈ [0, 2] jest parametrem behawioralnym nazywanym różnicą wagi. W przypadku problemu plecakowego stosujemy F=1.*

*Przykładowo:*

*D = {0, 1, 0, 0, 0, 1, 1} +1\* ( {0, 0, 1, 1, 1, 0, 0} - {1, 1, 1, 0, 0, 0, 1} );*

*D = {0, 1, 0, 0, 0, 1, 1} + {-1, -1, 0, 1, 1, 0, -1};*

*D = {-1, 0 ,0, 1, 1, 1, 0};*

*W tym wypadku widzimy, że wartości elementów dawcy nie wykraczają poza dozwolone granice. W takich wypadkach jest przeprowadzana operacja normalizacji. Nieodpowiedni element zbioru jest wówczas poddawany funkcji normalizującej, może to być funkcja prosta np.: sin, cos, tanh. Także możliwe jest ograniczenie do dolnej lub górnej granicy dozwolonego przedziału, albo wygenerowanie nowej wartości z dozwolonego przedziału. Dla danego przykładu wykorzystamy ograniczenie dolnej granicy przedziału, ponieważ -1 < 0, zamieniamy -1 na 0.*

*Po normalizacji D = D = {0, 0 ,0, 1, 1, 1, 0};*

1. Za pomocą metody mutacji „dawca” jest łączony z osobnikiem ***I***.
2. Nowo powstały osobnik jest poddawany ocenie
3. Jeżeli wynik nowego osobnika jest lepszy niż wynik początkowego osbnika I, osobnik I jest zamieniany na osobnika nowopowstałego.
4. Proces jest powtarzany aż do momentu gdy zostanie osiągnięty potomek z najlepszą oceną lub przez wskazaną ilość iteracji.

Optymalizacja Roju Cząstek

**Optymalizacja Roju Cząstek (ang. Particle Swarm Optimization (PSO))** algorytm służący do optymalizacji problemów poprzez iteracyjne ulepszanie położenia kandydata. Nie jest to algorytm ewolucyjny jak poprzednie, ale był napisany jako reprezentacja ruchu stada ptaków, a także ławicy ryb. Jego autorami są James Kennedy oraz Russell C. Eberhart. Algorytm również posiada populacje (rój) kandydatów na rozwiązanie (cząstek). Cząstki te poruszają się po przestrzeni rozwiązań zgodnie z prostymi równaniami. Ruchy cząstek są kierowane względem ich najlepszej (optymalnej) pozycji w przestrzeni rozwiązań. Proces ten jest powtarzany aż do momentu, gdy zostanie znalezione rozwiązanie lub przez zadaną ilość iteracji. Jednak algorytm PSO nie zapewnia, że znalezione rozwiązanie będzie rozwiązaniem satysfakcjonującym.

Zasada działania algorytmu:

**Faza początkowa:**

1. Dla każdej cząsteczki w roju generujemy pozycję początkową ***p***
2. Wygenerowaną pozycję ustawiamy jako najlepszą dotychczasową pozycję ***b*** dla danej cząstki.
3. Jeżeli pozycja cząstki ***b*** jest lepsza niż dotychczasowa najlepsza pozycja globalna ***g*** wśród wszystkich cząstek, ustawiamy ją jako najlepszą globalną.
4. Generujemy prędkość początkową ***v*** jako 0.

**Faza iteracji:**

1. Dla każdego wymiaru cząstki generujemy dwie losowe liczby ***r1, r2*** z przedziału [0; 1]
2. Aktualizujemy prędkość cząstki zgodnie ze wzorem:
3. Aktualizujemy pozycję cząsteczki zgodnie ze wzorem:
4. Jeżeli pozycja ***p*** jest lepsza niż pozycja ***b:*** ***b = p***
5. Jeżeli pozycja ***b*** jest lepsza niż pozycja ***g: g = b***

PROCes Programowania

Problematyka

Problemem wybranym do optymalizacji jest problem przewidywania wyników. Posiadając wektory ***v1, v2, v3, ..... vn-1, vn*** o rozmiarze ***N*** i przemnażając wektor ***vi*** przez macierz wag o rozmiarach ***N****x****N*** powinniśmy otrzymać wektor jak najbardziej podobny do ***vi+1***. Po przeprowadzeniu optymalizacji i przemnożeniu wektora ***vn*** przez macierz otrzymamy wektor **vn+1,** czyli nasz wektor przewidywalny.

Język programowania

Do realizacji zagadnień wybrany został język programowania Python w wersji 3.4. Python jest językiem bardzo elastycznym, prostym w użytku. Posiada szerokie spektrum przeróżnych bibliotek przydatnych w programowaniu sztucznej inteligencji.

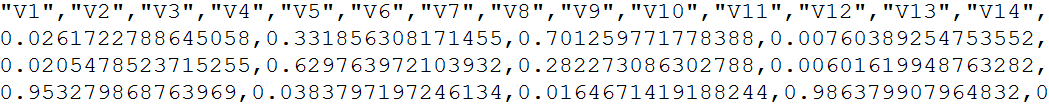
Wspólne cechy algorytmów

Wszystkie oprogramowane algorytmy mają wspólne cechy takie jak np:

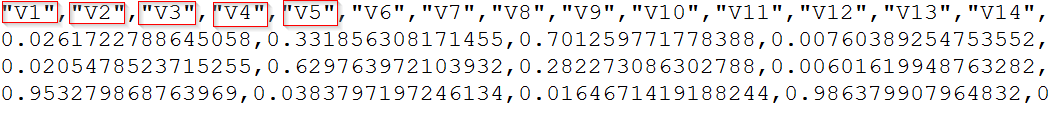
* Wczytywanie danych treningowych oraz danych do testowania z pliku csv.
* Generowanie macierzy wag.
* Obliczanie wyjścia.
* Funkcja oceniająca
* Wyświetlanie wyników przy pomocy wykresu.

W tym podrozdziale przyjrzymy się tym funkcjom bliżej.

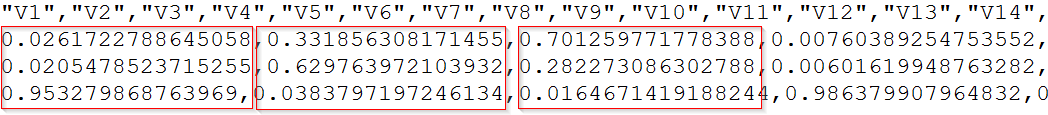
**Wczytywanie danych z pliku csv:**

Pliki csv z wektorami treningowymi oraz wektorami do testowania mają postać następującą.

Znakiem oddzielającym w naszym przypadku jest **`,`** (przecinek), a jedna kolumna odpowiada jednemu wektorowi. W pierwszym wierszu mamy numery wektorów, w sumie jest ich 2100 w zestawie treningowym i 900 w zestawie testującym.



W następnych wierszach mamy same wektory, w zależności od tego jakich wymiarów są wektory, tyle wierszy będzie miał plik + 1 wiersz z numerami wektorów. Czyli wektory jednowymiarowe miałyby dwa wiersze, dwu wymiarowe trzy itd.. W tym wypadku są przedstawione wektory o wymiarze 3.



Do wczytywania plików została wykorzystana biblioteka ***csv.*** W Pythonie biblioteki są załączane na początku kodu przy użyciu słowa kluczowego **import.**

* Gdy mamy załączoną bibliotekę do obsługi plików csv pierwsze co robimy to otwieramy nasz plik przy pomocy funkcji standardowej ***open().*** Jako argument tej funkcji podajemy ścieżkę do naszego pliku csv.
* Następnie z wykorzystaniem funkcji ***reader()*** z biblioteki ***csv***odczytujemy plik. Jako argumenty funkcji reader() podajemy nasz wcześniej otworzony plik, dodatkowo możemy podać typ separatora (znaku oddzielającego), który domyślnie jest ustawiany jako **`;`** (średnik).
* Kolejny krokiem jest wczytanie wszystkich danych. W naszym przypadku opuszczamy pierszy wiersz, w którym są nazwy wektorów (nie będą one nam potrzebne). Następnie dla każdego wiersza są sczytywane i zapisywane do listy elementy wektorów. Pierwsza kolumna będzie przypisana do wektora o indeksie 0, następna do indeksu 1 itd.. Dla kolejnych wierszy proces jest taki sam.
* Po wczytaniu danych przypisujemy do zmiennej ***N***  liczbę kolumn, zmienna ***N*** będzie nas informowała ile mamy wektorów. Do zmiennej ***K*** przypisujemy ilość wierszy-1, zmienna ***K*** wskazuje jakiego rozmiaru są wektory, a na jej podstawie będą budowane macierze wag.

**Generowanie macierzy wag:**

Macierze wag są generowane na podstawie rozmiarów wektorów z pliku csv. Jeżeli w pliku są wektory o rozmiarze 2 będą generowane macierze o rozmiarach 2x2, jeżeli wektory mają rozmiar 3 macierze będą rozmiaru 3x3 itd.. Elementy macierzy (wagi) muszą być z przedziału [-1; 1]. Do generowania takich macierzy wykorzystana została biblioteka ***NumPy.*** Przy użyciu klasy ***random*** z tej biblioteki oraz funkcji ***uniform()*** generowanie macierzy to tylko jedna linijka kodu. Wystarczy, że jako parametry funkcji ***uniform()*** podamy dolną i górną granicę generowanych liczb oraz rozmiar macierzy.

**Obliczanie wyjścia:**

Dla każdego wektora ***v***, ***v*** = [***v1, v2, .., vc***]T,***vi*** ∈ [0, 1] na wyjściu otrzymujemy wektor ***y*** = [***y1, y2,...,yc***] *T* , ***yi*** ∈ [0, 1]. Dla każdego i-tego elementu wyjście jest obliczane w sposób następujący:

dla ***i*** = 1, 2, ..., c. Ponieważ wyjściowe wartości muszą być z przedziału [0;1] funkcja ***f*** jest funkcją sigomidalną z parametrem ***β,*  *β*** > 0:

W naszym przypadku funkcja realizująca powyższe założenia nazywa się ***computeOutput()*** jako parametry tej funkcji musimy podać macierz wag oraz wektor wejściowy, opcjonalne jest podawanie wartości ***β,***  domyślnie ***β*** = 5.

**Funkcja oceniająca (fitness):**

Funkcja oceniająca jest najważniejszą funkcją we wszystkich algorytmach, to właśnie od jej wyników zależą dalsze losy programu. Macierze są oceniane na podstawie Błędu średniokwadratowego (MSE, ang. Mean Squared Error). Określmy **Y** jako zbiór wektorów otrzymany w wyniku funkcji obliczającej wyjście, gdzie **Y** = [***y1, y2, ..., yi***], ***yi***=[***yi1, yi2, .., yij***]T, ***yij***∈ [0, 1]. Następnie określmy zbiór wektorów wyjściowych treningowych **T** = [***t1, t2, ..., ti***], ***ti*** =[***ti1, ti2, .., yij***]T, ***tij***∈ [0, 1]

W tym przypadku ***N*** oraz ***K*** to zmienne pozyskane podczas sczytywania danych odpowiednio ilość wektorów, oraz ich rozmiar.

**Wyświetlanie wyników przy pomocy wykresu:**

Najbardziej efektownie oraz wyraźnie przedstawić efektywność naszych algorytmów możemy za pomocą wykresów. Do rysowania wykresów 2D w Pythonie wykorzystywana jest biblioteka ***Matplotlib*** w tym wypadku wykożystana została jej klasa ***pyplot.*** Aby móc narysować wykres musimy posiadać listę z danymi, w tym wypadku są to wyniki funkcji oceniającej. Po zakończeniu każdej iteracji dodajemy do listy wyników najlepszy wynik w danej iteracji. Po zakończeniu cyklu iteracji algorytmu naszą listę z wynikami wystarczy przekazać jako argument funkcji ***plot***() lub ***semilogx***() z wyżej wymienionej klasy oraz biblioteki.

Algorytm genetyczny

…

…

…

Ewolucja różniczkowa

…

…

…

Optymalizacja Roju Cząstek

…

…

…

TESTOWANIE

Dane

Przedstawione wcześniej algorytmy służą do pracy z szeregami czasowymi. W naszym wypadku zostały wykorzystane dane syntetyczne. Do utworzenia danych zostały użyte cykliczne powtórzenia trzech liczb 1, 5, 7 oraz ich zaszumienie mnożąc przez przypadkowe liczby z przedziału [0,5; 1,5]. W ten sposób zostało wygenerowane ponad 3000 przypadków, z czego 2100 do procesu treningowego, oraz 900 do testowania. Dane zostały wygenerowane za pomocą programu Microsoft Excel.

Wyniki

**Algorytm genetyczny:**

**Ewolucja różniczkowa:**

**Optymalizacja Roju Cząstek**

**ZAKOŃCZENIE**

...

….

.

.

.

.

.

.

..

SPIS TABLIC

SPIS RYSUNKÓW

Rysunek 1. Kod strony pozwalający na wybór podstrony 9

Rysunek 2. Kod strony ładujący odpowiednią podstronę 9

Rysunek 3. Kod połączenia z bazą danych 10

Rysunek 4. Kod pracy z bazą danych 10

Rysunek 5. Kod wyświetlania wszystkich galerii 11

Rysunek 6. FancyBox 12

Rysunek 7. Kod wyświetlania konkretnej galerii 12

Rysunek 8. Strona główna 14

Rysunek 9. Wszystkie galerie 14

Rysunek 10. Konkretna galeria 15

Rysunek 11. Kontakt 15

Rysunek 12. O nas 16

BIBLIOGRAFIA

ŹRÓDŁA INTERNETOWE

1. HTML https://pl.wikipedia.org/wiki/HTML [dostęp: 25.05.2017].
2. CSS https://pl.wikipedia.org/wiki/Kaskadowe\_arkusze\_stylów [dostęp: 25.05.2017].
3. JavaScript https://pl.wikipedia.org/wiki/JavaScript [dostęp: 25.05.2017].
4. PHP https://pl.wikipedia.org/wiki/PHP [dostęp: 25.05.2017].
5. MySQL https://pl.wikipedia.org/wiki/MySQL [dostęp: 25.05.2017].