POLITECHNIKA SLASKA W GLIWICACH
WYDZIAL INZYNIERII BIOMEDYCZNEJ

Projekt

Biocybernetyka
Optymalizacja rojem cząstek – symulacja oraz wizualizacja

Anna Wieczorek

Gliwice, 10 grudnia 2013

Spis treści

Spis rysunków

1. Wprowadzenie

Metody optymalizacji nieliniowej(*gradientowe*) w uczeniu sieci neuronowych, takie jak metoda *największego spadku, newtonowska* czy metody *pseudonewtonowskie* znajdują zastosowanie w optymalizacji parametrów skomplikowanej, nieregularnej, różniczkowalnej funkcji. Dla potrzeb optymalizacji funkcji nieróżniczkowalnej, bądź nieciągłej stosuje się metody meta heurystyczne. Do tych technik zalicza się optymalizacja rojem cząstek(*ang.*Particle Swarm Optimalization), dalej nazywana PSO.
PSO jest metodą sztucznej inteligencji, która może być użyta do szukania przybliżonych rozwiązań w zakresie dynamicznych obliczeń problemów minimalizacji i maksymalizacji. Metoda została wdrożona przez R. Eberharta i J.Kennediego w 1995, zainspirowani zachowaniem stadnym ptaków i ławic ryb. Idea algorytmu bazuje na symulacji zachowania gromady ptaków - ich poruszania się w celu znalezienia pokarmu. Każda cząsteczka roju(ang.*Particle*) porusza się w sposób nieprzewidywalny, mimo swoistej indywidualności poszukiwanie pożywienia przez każdą z cząsteczek przynosi rojowi korzyść.

1.1 Cel projektu

Realizacja projektu "Optymalizacja rojem cząstek – symulacja oraz wizualizacja" obejmowała zaimplementowanie programu, który spełniałby następujące zadania:

- rozwiązywałby problem optymalizacji parametrów funkcji
- optymalizacja parametrów obrazu w odcieniach szarości
- rozwiązanie problemu przedstwione w formie wizualizacji

Do symulacji i wizualizji algorytmu PSO użyto środowiska .NET, sama implementacja w języku C#.

2. Parametry algorytmu oraz jego działanie

2.1 Parametry roju

Rój składa się z n liczby agentów. Zarówno rojowi jak i samym cząsteczkom można przypisać następujące parametry:
Parametry *cząsteczki*, zależne od wymiarowości zadania:

- **Pozycja** - wektor pozycji; Inicjalizowany liczbami pseudolosowymi o rozkładzie jednorodnym od zadanej wartości minimalnej do maksymalnej.
- **Przyspieszenie** - wektor przyśpieszenia
- **Najlepsza pozycja** - wektor najlepszej pozycji; inicjalizowany wektorem pozycji przy tworzeniu cząsteczki. Jego wartość zostaje zmieniona, gdy cząsteczka znalazła lepiej oceniane położenie. Wektor ten staje się później jednym z dwóch atraktorów cząsteczki.

Parametry *roju*:

- **Cząsteczki** - Wektor zawierający wszystkie cząsteczki roju jako obiekty.
- **Najlepsza pozycja** - Wektor najlepszej pozycji; inicjalizowany wektorem pozycji pierwszej stworzonej cząsteczki. Jego wartość zostaje zmieniona, gdy jedna z cząsteczek roju znalazła lepiej oceniane położenie od swojego i jest ono lepiej oceniane niż aktualne najlepsze położenie roju. Wektor ten staje się później jednym z dwóch atraktorów cząsteczek.

2.2 Parametry algorytmu i jego działanie

Parametry algorytmu:

- **Omega** - Parametr odpowiadający za "wygasanie" ruchu cząsteczek.
  Dla Omega =1 cząsteczki nie zwalniają z upływem czasu, dla dla Omega < 1 cząsteczki zwalniają z upływem czasu, dla Omega> 1 cząsteczki przyśpieszają z upływem czasu

- **fiCząsteczki** Parametr odpowiedzialny za indywidualność cząsteczek. Bezpośrednio zależny od fi roju. Gdy jest większy niż fiRoju cząsteczki stają się bardziej samodzielne.

- **fiRoju** Parametr odpowiedzialny za stadność cząsteczek. Bezpośrednio zależny od fiCząsteczek. Gdy jest większy niż fi cząsteczek, cząsteczki stają się bardziej stadne.

Idea działania algorytmu:

1. obliczanie przyspieszenia dla każdej cząsteczki z osobna
2. zmiana jej pozycje zgodnie z wyliczonym przyśpieszeniem
3. wyliczenie wartości funkcji dla jej aktualnej pozycji
4. następnie sprawdzenie czy aktualna pozycja jest lepsza od najlepszej pozycji wg. cząsteczki, jeżeli tak to ją zmieniamy i sprawdzamy czy jest lepsza od najlepszej pozycji wg roju, jeżeli tak, to też ją zmieniamy.

3. Specyfikacja zewnętrzna

Aplikacja PSO została stworzona przy użyciu C# Windows Froms Application. Na aplikację składają się dwa moduły :

1. Function

2. Image

Użytkownik może wybrać z głównego menu(Main Menu), moduł stosujący algorytm PSO dla funkcji bądź obrazka.

3.1 Moduł Function

Moduł Function składa się z głównego okna, w którym znajdują się następujące elementy:

- elementy edytowalne do wprowadzania odpowiednio wartości: liczby agentów, ilość iteracji, omega, przedział, fiRoju, fiCząsteczki

- element statyczny w formie okna wyświetlającego wyniki wizualizacji na wykresie funkcji

- przycisk 'Start' do zainicjalizowania wizualizacji

Domyślną funkcją na której przeprowadzona jest optymalizacja jest funkcja sinus. Optymalnymi funkcjami w której można szukać maksimum bądź minium jest moduł, cosinus i inne funkcje trygonometryczne. Użytkownik ma możliwość wprowadzenia odpowiednich danych potrzebnych do obliczeń takich jak: liczba cząsteczek, ilość iteracji i omega. Możliwe jest sterowanie zachowaniem roju - poprzez dobranie odpowiednich wartość omegi można uzyskać efekt zwalniania lub rozbiegania się cząsteczek. Podobnie jeśli chodzi o kwestie indywidualności cząsteczek - każdy agent, poprzez zmianę wartości parametru fiCząsteczki(zależne od fiRoju), może być mniej lub bardziej nieprzewidywalny. Parametr określający stadność roju fiRoju pozwala uzyskać efekt tworzenia się gromady, bądź mniej zaalokowanych obok siebie cząsteczek; gdy jest większy niż fiCząsteczki to agenci są bardziej stadni. Po wprowadzeniu wszystkich danych należy nacisnąć przycisk "Start". Wyniki można obserwować na bieżąco, z każdą iteracją widoczne jest przesunięcie cząsteczek.

3.2 Moduł Image

Moduł Image obrazuję działanie algorytmu na obrazie w skali szarości. Dzięki optymalizacji można zaobserwować i jednocześnie zrozumieć poszukiwanie optimum przez rój - w tym wypadku odpwiedniego odcienia szarości, recznie wybranego przez użytkownika. Moduł Image składa się z głównego okna oraz elementów edytowanych:

- elementy edytowalne do wprowadzania parametrów roju: liczby agentów, ilość iteracji, omega, przedział, fiRoju, fiCząsteczki
- slider, którym dobierany jest odcień szarości
- przycisk 'Browse' do otwierania eksploratora plików
- przycisk 'Start' do zainicjalizowania działania roju

Użytkownik wybiera z eksploratora plików zdjęcie. Zalecane jest aby obraz był maksymalnie rozdzielczości z rzędu 500x500(z uwagi na rozmiar kontrolki wyświetlającej obraz), a w zakresie skali szarości zawierał szereg odcieni od czarnego(#000000) do białego(#000000)3.1 Następnie wprowadzane są wartości odpowiadające parametrom roju i algorytmu, m.in: omega, fiRoju i fiCząsteczki. Po kliknięciu przycisku 'Start' inicjalizowany jest ruch cząsteczek po obrazie. Poprzez przesunięcie slider'a użytkownik może manipulować ruchem roju i obserwować zmianę ich położenia z kolejną iteracją. Rój dąży do koloru przedstwionego na suwaku.

Gray Shades	HEX	RGB
	#000000	rgb(0,0,0)
	#080808	rgb(8,8,8)
	#101010	rgb(16,16,16)
	#181818	rgb(24,24,24)
	#202020	rgb(32,32,32)
	#282828	rgb(40,40,40)
	#303030	rgb(48,48,48)
	#383838	rgb(56,56,56)
	#404040	rgb(64,64,64)
	#484848	rgb(72,72,72)
	#505050	rgb(80,80,80)
	#585858	rgb(88,88,88)
	#606060	rgb(96,96,96)
	#686868	rgb(104,104,104)
	#707070	rgb(112,112,112)
	#787878	rgb(120,120,120)
	#808080	rgb(128,128,128)
	#888888	rgb(136,136,136)
	#909090	rgb(144,144,144)
	#989898	rgb(152,152,152)
	#A0A0A0	rgb(160,160,160)
	#A8A8A8	rgb(168,168,168)
	#B0B0B0	rgb(176,176,176)
	#B8B8B8	rgb(184,184,184)
	#C0C0C0	rgb(192,192,192)
	#C8C8C8	rgb(200,200,200)
	#D0D0D0	rgb(208,208,208)
	#D8D8D8	rgb(216,216,216)
	#E0E0E0	rgb(224,224,224)
	#E8E8E8	rgb(232,232,232)
	#F0F0F0	rgb(240,240,240)
	#F8F8F8	rgb(248,248,248)
	#FFFFFF	rgb(255,255,255)

Rys. 3.1: Skala szarości

4. Specyfikacja wewnętrzna

4.1 Wstęp

Do implementacji problemu PSO użyto środowisko .NET. Program został stworzony w oparciu o C# Windows Form Application. Windows Form API definiuje metody, dzięki którym możliwa była wizualizacja problemu PSO w formie danych w interfejsie użytkownika(UI), utrzymując przy tym model obiektowy.

4.2 Program

Program.cs to główna klasa w aplikacji, punkt wejściowy programu. Z jej poziomu uruchomiana jest klasa MainMenu.cs

4.3 MainMenu

W tej klasie znajdują się zdarzenia $button_Click$, które wywołują odpowiednio funkcję ImageForm.cs oraz Function.cs. Wywołanie klas ImageForm.cs oraz Function.cs następuję po przekazaniu obiektu do metody Show().

4.4 ImageParticle oraz Particle

Są to klasy przechowujące właściwości cząsteczek. ImageParticle.cs ustawia właściwości dla pikseli w obrazie zaś Particle.cs dla wartości funkcji. Właściwości te przechowują informacje o cząsteczkach takie jak: pozycja startowa, najlepsza pozycja, przyspieszenie.

4.5 Function

Klasa symulująca zachowanie roju w poszukiwaniu maksimum funkcji.

4.5.1 Stałe

Zmienna xStep określa odległość między dwoma sąsiednimi argumentami dziedziny funkcji.

4.5.2 Pola

Obejmują następujące zmienne wykorzystywane w programie:

- minX - wartość minimalna dziedziny
- maxX - wartość maksymalna dziedziny
- omega - wartość wpływająca na przyspieszenie, odpowiada za wygaszanie ruchu
- numberOfParticles - ilość cząsteczek
- iteration - licznik aktualnej iteracji
- maxIterations - maksymalna liczba iteracji
- fiParticle - indywidualność cząsteczki
- fiSwarm - stadność roju
- List¡Particle¿ swarm - lista zawierająca cząsteczki

  functionX - tablica przechowująca dziedzinę funkcji
- bestSwarmPosition - najlepsza pozycja roju

4.5.3 Metody

MoveSwarm

Metoda obliczająca kolejne położenie cząsteczek. W tej metodzie obliczane jest przyspieszenie cząsteczek(*velocity*) wg wzoru:

$$velocity = omega * particle.Velocity + fiParticle * rand.NextDouble() * (particle.BestPosition - particle.Position) + fiSwarm * rand.NextDouble() * (bestSwarmPosition - particle.Position)$$

gdzie: omega - współczynnik przyspieszenia; particle.Velocity - przyspieszenie cząsteczki; fiParticle - fi cząsteczki; particle.BestPosition - najlepsza pozycja cząsteczki; particle.Position - pozycja początkowa cząsteczki; fiSwarm - fiRoju; bestSwarmPosition - najlepsze położenie roju
Kolejno obliczane są najlepsze położenie cząsteczek względem wartości funkcji.

InitializeSwarm

Metoda generująca rój. Do listy dodawane są kolejne cząsteczki wraz z parametrami - pozycja i prędkość

DrawFunction

Metoda rysuje funkcje w okienku. Przyjmuję jako argument dziedzinę funkcji oraz parametry charakterystyczne dla wykresu, po czym generuje wykres.

EvaluateFunction

Zwraca wartość funkcji dla zadanego argumentu.

4.5.4 Zdarzenia

btnStart_Click

Zdarzenie generujące wizualizację ruchu PSO. Startuje całą symulację roju.

t_Tick

Symuluję jedną iterację algorytmu.

4.6 ImageForm

4.6.1 Konstruktory

Inicjalizuje następujące komponenty w programie:

- przedział slider'a - reprezentuje wartość RGB(0-255) skali szarości
- Timer - generuje cykliczność ruchu cząsteczek

4.6.2 Pola

Zmienne wykorzystane w programie:

- bmp - zmienna przechuje obraz w postaci bitmapy
- omega - odpowiada za wygaszanie ruchu
- numberOfParticles - liczba agentow
- iteration - licznik aktualnej iteracji
- maxIterations - maksymalna liczba iteracji
- List¡ImageParticle¿ swarm - lista zawierająca cząsteczki
- bestSwarmPosition - definiuje dwuwymiarowy punkt w którym znajduje się najlepsza pozycja roju(piksel)
- fiParticle - indywidualnosc czasteczki
- fiSwarm - stadnosc roju

4.6.3 Metody

MoveSwarm

Metoda obliczająca prędkość cząsteczek oraz ich kolejne położenie. Nowe położenia cząsteczek przyrównane jest do dwuwymiarowości piksela(wysokość, szerokość), podobnie prędkość przyjmuje składowe x i y.

EvaluateFunction

Metoda ta porównuje aktualnie badany piksel bitmapy z kolorem wzorcowym. Zwraca wartość zmiennoprzecinkową - im wartość bliższa zeru, tym piksele są bardziej podobne(pod względem koloru). Składowe RGB każdego piksela przeliczane są według wzoru:

$$0.11B + 0.59G + 0.30R$$

następnie porównywane ze wzorcem.

DrawSwarm

Metoda nanosi położenie cząsteczek na obraz w postaci pikseli.

InitializeSwarm

Metedowa generująca rój. Na podstawie parametrów wejściowych generuje cząsteczki roju oraz mapuje je na dwuwymiarową przestrzeń bitmamy.

4.6.4 Zdarzenia

trackBar1_Scroll

Funkcja zmieniająca kolor w okienku wyboru koloru podczas poruszania suwakiem. Domyślnie jest to czarny(255).

btnLoadimage_Click

Obsługa zdarzenia otwarcia okna dialogowego. Metoda otwiera plik i wyświetla go w picturebox'ie jako bitmapa.

btnStart_Click

Zczytuje parametry wejściowe wprowadzone przez uzytkownika, inicjalizuje rój i rozpoczyna działanie algorytmu.

t_Tick

Symuluję jedną iterację algorytmu.

4.7 Wizualizacja ruchu cząsteczek

W celu wizualizacji, w obu modułach(Image oraz Function) została użyta klasa Timer dostarczana wraz z frameworkiem .NET. Działanie algorytmu zostało umyślnie zwolnione przy użyciu wyżej wymienionej klasy, w celu przedstawienia na wykresie lub obrazku aktualnej pozycji cząsteczek roju. Ogólnie po upłynięciu ustalonego czasu timera, przeliczoene zostaje nowe położenie roju, następnie jego wyrysowanie na wykresie tudzież obrazku, następnie cykl powtarza się.

5. Wnioski

PSO jest metaheurystyczną metodą optymalizacji skomplikowanych problemów opartą o ideę zachowwań stadnych gromady ptaków. Działanie algorytmu można porównać do swoistego instynktu zwierząt skierowanego na poszukiwanie pożywienia - w algorytmie odpowiednikiem osiągnięcia celu(znalezienie pożywienia) jest znalezienie optimum - w przypadku nieróżniczkowalnej funkcji jest to na przykład maksimum, skomplikowane w obliczeniach do wyznaczenia. Realizacja projektu oparta na implementacji w języku C# pozwoliła przybliżyć w sposób obrazowy zarówno zasadę działania algorytmu jak i nieprzewidywalne, a jednak efektywne zachowania gromad zwierząt. Sama idea algorytmu jest wykorzystywana w wielu dziedzinach - na pozór nie znajdujących odwzorowania w naturze. Przykładem jest przyspieszenie wydajności procesów biznesowych bądź samych aplikacji. Wydaje mi się, że rozwijanie takiego typu usprawnień jest niesamowite - równocześnie można zastosować nowe technologie oraz wpleść w sam proces ich tworzenia odrobinę natury. Doskonale to widać w realizacji projektu - możnaby wykorzystać algorytm PSO w zakresie przetwarzania obrazów bądź obliczeń numerycznych.

Bibliografia

[1] Jacek Łęski *Systemy neuronowo-rozmyte* 2008.

[2] Raymond Chiong *Nature-Inspired Informatics for Intelligent Applications and Knowledge Discovery: Implications in Business, Science, and Engineering* 2010.