Sprawozdanie - Laboratorium nr 10

Poszukiwanie minimum wartości funkcji metodą

największego spadku w 2D

Hubert Wdowiak, 14.05.2020

1. Wstęp Teoretyczny

Podczas trwania zajęć poznaliśmy sposób na odnajdywanie przybliżonych wartości minimów globalnych funkcji. Korzystaliśmy przy tym z iteracyjnej metody największego spadku, która bazuje na :

- obliczaniu gradientu analizowanej funkcji,
- aktualizacji wartości poszukiwanego przybliżenia,
- a następnie powtórzeniu procesu, do momentu, aż wykonana zostanie maksymalna liczba iteracji, bądź spełniony zostanie warunek zadany przez nas.

Przebieg każdej z iteracji wygląda następująco:

Zaczynając od początkowych wartości $x_1, x_2, \ldots, x_n$, dla funkcji przyjmującej n argumentów, obliczane są pochodne funkcji f dla każdej, z w/w wartości, które wspólnie tworzą gradient ∇ .
Wzór na pochodne cząstkowe:

$$\frac{df}{dx_i} = \frac{f(x_1,\ldots,x_i+d\Delta, \ldots, x_n) - f(x_1,\ldots,x_i - \Delta x_i, \ldots, x_n)}{2\Delta x_i}, \tag{1}$$

gdzie Δx_i oznacza dowolnie ustaloną niewielką wartość kroku przestrzennego.

Posiadając pochodne składowe gradientu, jesteśmy w stanie zaktualizować nasze początkowe wartości argumentów funkcji f, przy pomocy wzoru:

$$x_{i\ zaktualizowane} = x_i - h \cdot \frac{df}{dx_i}, \tag{2}$$

gdzie h oznacza współczynnik odpowiadający za szybkość zmian wartości poszukiwanych argumentów, w każdej z iteracji.

Metoda ta jest podstawą w większości algorytmów uczenia maszynowego i doczekała się wielu udoskonaleń. Głównie wiążą się one z pytaniem: Jak szybko powinny się zmieniać wartości naszego przybliżenia, tak aby nasz program osiągnął zbieżność/dokładny wynik jak najszybciej, a przy tym nie utknął w minimum lokalnym. W tym celu zaprojektowano rozwiązania takie jak Adagrad czy Momentum, które manipulują wagą zmian i pomagają rozwiązywać te problemy.

2. Opis problemu

W trakcie laboratorium naszym zadaniem było odnalezienie minimum lokalnego funkcji określonej wzorem:

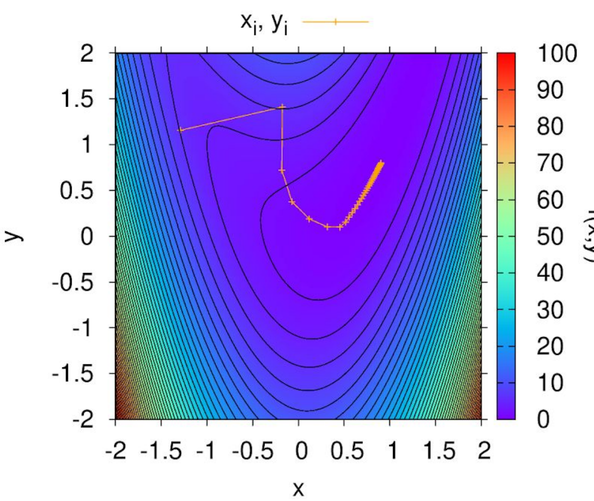
$$f(\vec{r}) = f(x, y) = \frac{5}{2}(x^2 - y)^2 + (1 - x)^2 \quad (3)$$

Jak widać, we wzorze, funkcja ta przyjmuje 2 argumenty, dlatego mówimy, że poszukiwanie zostało przeprowadzone w 2D - w każdej z iteracji obliczane były wartości dwóch pochodnych cząstkowych. Treść zadania mówiła, aby wykonać poszukiwanie dwukrotnie, dla dwóch różnych wartości kroku przestrzennego: 10^{-2} oraz 10^{-3}.

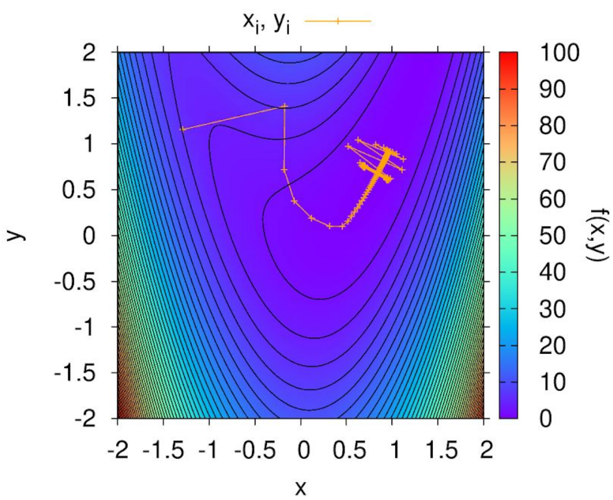
W każdym z powtórzeń, wartość *h* z wzoru (2), pozostała stała równa 0.1, a wartości początkowe współczynników *x* i *y*, w funkcji *f* zostały ustalone na (-0.75, 1.75). Maksymalna wartość iteracji wyniosła 1000.

Aby przeprowadzić aproksymację, napisaliśmy program w języku C++, w którym nie potrzebne było wykorzystywanie żadnych specjalistycznych bibliotek numerycznych. Na koniec zwizualizowaliśmy przebieg iteracji przy pomocy programu *gnuplot*.

3. Wyniki



Wykres 1. Przybliżenia minimum globalnego, dla ε = 1e-2



Wykres 2. Przybliżenia minimum globalnego, dla ε = 1e-3

Na powyższym *Wykresie 1* widać, że ustalona wartość ε = 1e-2, okazała się być stosunkowo dobra, gdyż algorytm przerwał swoje działanie już po 36 obrotach pętli, a pomarańczowe punkty na wykresie zdają się zmierzać do punktu zbieżności (x = 1, y = 1). Mimo wszystko nasze wartości otrzymane w wyniku działania programu wyniosły odpowiednio: *x* = 0.900038, *y* = 0.793695. Zatem zmniejszając zakres marginesu błędu jesteśmy w stanie otrzymać wyniki bardziej zbliżone do oczekiwanych.

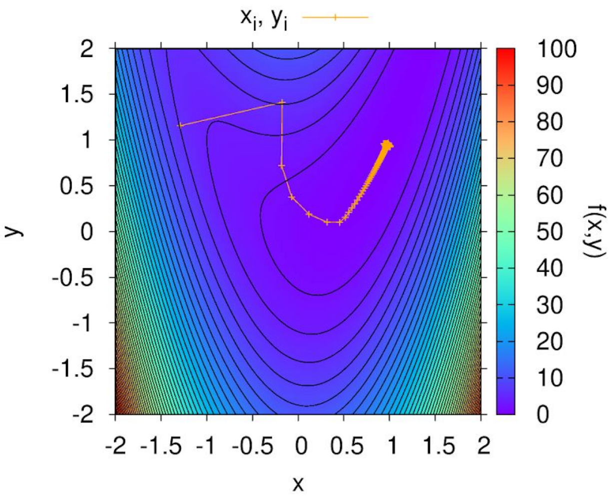
Wykres 2, pokazał jednak efekt zawężenia warunku stopu. W wyniku zmniejszenia ε do 1e-3, nasz program nie zatrzymał się i wykonał pełen zakres iteracji równy 1000. Oprócz tego na wykresie zobaczyliśmy, że około 60 przejścia pętli, program zaczyna krążyć wokół prawdopodobnego minimum globalnego, a wynik końcowy to x = 0.94205 i y = 0.621386. Z jednej strony otrzymany x jest bliższy prawdziwemu x, jednak otrzymany y jest jeszcze mniej dokładny, niż ten który uzyskaliśmy dla ε= 1e-2. Przyczyną takiego stanu rzeczy, może być fakt "przeskakiwania minimum globalnego" i wkraczania w okolice minimów lokalnych. Zwiększając liczbę iteracji, wynik nie ulega poprawie i program wciąż przeskakuje nad znalezionym hipotetycznym minimum globalnym.

Aby uzyskać jak najlepszy efekt, należy dobrać odpowiednio współczynnik *h*, który jeśli będzie odpowiedni, pozwoli pominąć minima lokalne, a finalnie nie będzie przeskakiwał i krążył około minimum globalnego. Stała wartość *h* nie jest najlepszym rozwiązaniem, gdyż prawdopodobnie uda się spełnić tylko jeden z powyższych warunków (w zależności od charakterystyki zadania). Ów kompromis, może być osiągnięty poprzez prowadzenie dynamicznego współczynnika *h*, który będzie zmieniał swoją wartość w trakcie iteracji: Na początku będzie większy, aby pominąć minima lokalne, a następnie stopniowo zmniejszy swoją wartość, aby znaleziona para (x,y), była możliwie bliska do punktu zbieżności.

Aby spróbować poprawić wynik, wprowadziliśmy usprawnienie:
w każdej iteracji poczynając od 50, *h*, które na początku wyniosło 0.1, zostało zmodyfikowane jako 0.99**h*. W ten sposób, dla ε= 1e-2, i dynamicznie zmieniającego się *h*, otrzymaliśmy wynik końcowy:

x = 0.984751,
y = 0.96654,

a sama pętla osiągnęła warunek stopu, w 96 przejściu. Dla ε= 1e-3, nie zaszły żadne zmiany, gdyż dla takiego warunku stopu, współczynnik *h*, nie miał znaczenia (do zatrzymania dochodziło, przy stałej wartości *h* i program nie krążył wokół wybranego punktu).



Wykres 3. Przybliżenia minimum globalnego, dla ε= 1e-2
i stopniowo zmniejszanego współczynnika *h*

4. Wnioski

Na podstawie prób odnalezienia minimum globalnego funkcji *f*, jesteśmy w stanie powiedzieć, że metoda największego spadku jest efektywna i dla przypadku 2D, okazała się być bardzo szybka. W zależności od dobranych parametrów warunku stopu oraz współczynnika *h*, metoda może się okazać bardzo dokładna, bądź nieefektywna. W przypadku dobrania zbyt małego ε, takiego jak 1e-2, w stosunku do stałej wartości $h = 0.1$, warunek zbieżności może zostać nigdy nie osiągnięty, a dalsza praca algorytmu nie przyniesie wymiernych efektów. Zatem możemy jednoznacznie stwierdzić, że owe połączenie parametrów było złe. Jednakże, odpowiednio manipulując wartością *h*, jesteśmy w stanie odnaleźć kompromis pomiędzy pomijaniem minimów lokalnych, a zbliżeniem do minimum globalnego.

Index of comments

1.1 Sama nabla to tylko operator, który sam w sobie nie ma związku z żadną konkretną funkcją. Przy symbolu nabli powinna stać funkcja f, żeby była jasność.

3.1 Bardzo dobry pomysł!
Można jeszcze zauważyć, że przy tym usprawnieniu wciąż widzimy pewne oscylacje na wykresie (tym razem już mniejsze), które są związane z wydłużonym kształtem funkcji i dynamicznie zmieniającym się gradientem. Tutaj jednak algorytm jest w stanie wyjść z tych oscylacji dzięki zmiennemu krokowi h.

3.2 Powinno być: 1e-3.
Dla 1e-2 program nie dociera do 50. iteracji.

3.3 Za to tutaj: 1e-2.

4.1 Warunek zbieżności nie został osiągnięty przy stałym h=0.1 dla epsilon = 1e-3.