Wstęp do sztucznej inteligencji

Przeszukiwanie przestrzeni

Jakub Robaczewski

Algorytm:

Metoda Newtona wykorzystuje gradient i hesjan funkcji w punkcie do wyliczenia następnego punktu, do którego powinien przejść algorytm. Odpowiednie wartości zostają podstawione do wzoru:

$$d = H_f^{-1}(x)\nabla q(x)$$
$$x_n = x_{n-1} + \beta d$$

Implementacja algorytmu:

Algorytm znajdowania maksimum metodą Newtona zaimplementowałem jako bazową klasę `NewtonMethod`, która przyjmuje wymiar przestrzeni pomniejszony o 1, a zatem wymiar danego mu punktu początkowego. Stworzyłem również klasy pochodne `NewtonMethod_A` i `NewtonMethod_B`, które dodają odpowiednie metody odpowiedzialne za obliczanie funkcji, gradientu i hesjanu na potrzeby obu funkcji. Tym sposobem można zastosować ten algorytm do dowolnej funkcji. Do testowania funkcji wykorzystuję metodę `test_B`, która bada algorytm dla równych wartości β, wypisując wartości do pliku. Do stworzenia wykresów wykorzystuję metodę `make_plot`.

Uwagi ogólne:

Badania dla rozmiaru kroku prowadziłem dla dim(x)=2, ale mój algorytm jest uniwersalny i można by je prowadzić przy dowolnym rozmiarze.

Problem stopu:

Problem zatrzymania algorytmu rozwiązałem przyjmując, że zatrzymuje się, gdy odległość 2 ostatnich punktów przez niego wyznaczonych jest mniejsza niż wartość `stop_acc` (domyślnie 0.000001) lub gdy ilość iteracji wyniesie `stop_n` (domyślnie 1000). Drugi warunek jest konieczny, ponieważ przy dużych wzmocnieniach β, program może nigdy nie dotrzeć do wyznaczonej dokładności.

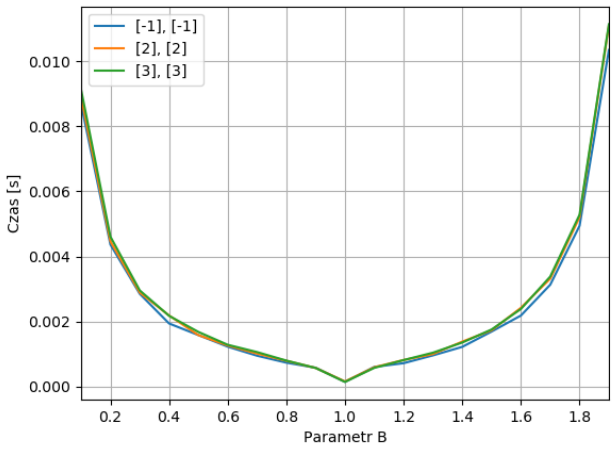
Wzór 1: $f(x) = -x^T x$

Funkcja 1 posiada tylko jedno maksimum, więc algorytm Newtona nie ma żadnego problemu z jego znalezieniem (punkt 0.0 i wynosi 0), niezależnie od punktu początkowego.

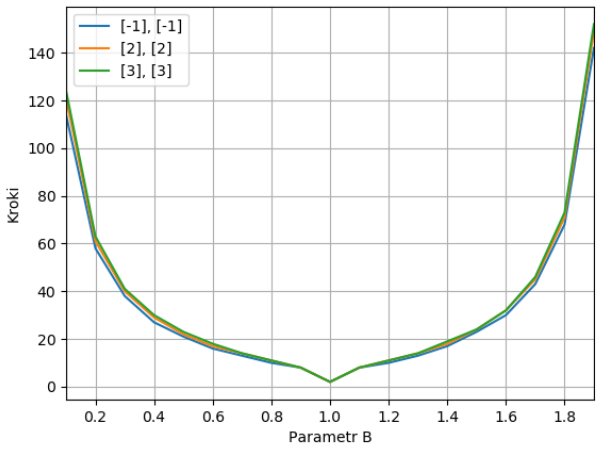
Czas działania algorytmu:

Czas algorytmu jest proporcjonalny do liczby wykonanych kroków, która zależy przede wszystkim od wybranej wartości β oraz w mniejszym stopniu, od odległości do maksimum globalnego. Gdy β<1, szybkość algorytmu rośnie wraz z wzrostem β, jednak, gdy β>1 algorytm ponownie zwalnia. Jest to spowodowane tym, że gdy β>1, algorytm zbliża się do maksimum obustronnie, „przeskakując" nad punktem końcowym.

Zależność czasu działania algorytmu od wartości B i punktu początkowego

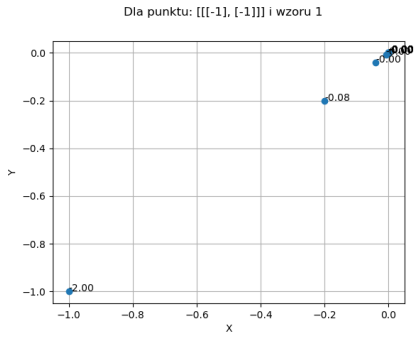


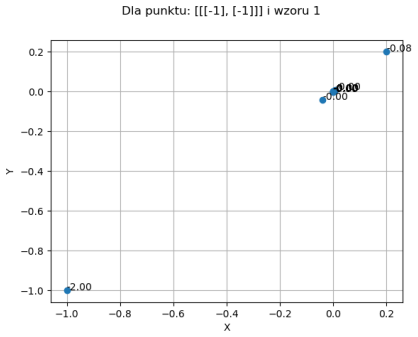
Zależność liczby kroków algorytmu od wartości B i puntku początkowego

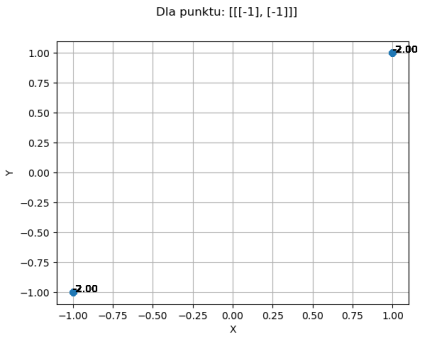


β	Czas działania algorytmu [s]		
	[-1,-1]	[2,2]	[3,3]
0.10	0,0091	0,0094	0,0096
0.20	0,0044	0,0046	0,0053
0.30	0,0028	0,0034	0,0031
0.40	0,0020	0,0023	0,0023
0.50	0,0015	0,0016	0,0017
0.60	0,0012	0,0013	0,0013
0.70	0,0010	0,0010	0,0010
0.80	0,0009	0,0008	0,0008
0.90	0,0007	0,0006	0,0006
1.00	0,0002	0,0002	0,0002
1.10	0,0006	0,0006	0,0006
1.20	0,0007	0,0008	0,0009
1.30	0,0010	0,0010	0,0011
1.40	0,0013	0,0014	0,0016
1.50	0,0018	0,0019	0,0019
1.60	0,0022	0,0024	0,0024
1.70	0,0032	0,0035	0,0034
1.80	0,0053	0,0053	0,0055
1.90	0,0106	0,0112	0,0116

Różnice w zbliżaniu się do punktu końcowego, gdy β<1, 1<β<2 i β=2







β=0.8		β=1,2	
0	[[-1], [-1]]	0	[[-1], [-1]]
1	[[-0.2], [-0.2]]	1	[[0.2], [0.2]]
2	[[-0.04], [-0.04]]	2	[[-0.04], [-0.4]]
3	[[-0.008], [-0.008]]	3	[[0.008], [0.008]]
4	[[-0.002], [-0.002]]	4	[[-0.002], [-0.002]]
5	[[-0.0003], [-0.0003]]		

Wyniki algorytmu:

Zauważamy, że dokładność algorytmu rośnie logarytmicznie wraz z parametrem β, ale po przekroczeniu β=1, dokładność spada ze względu na oscylację algorytmu wokół punktu docelowego.

β	Wyniki algorytmu [f(x) * 10^{11}]		
	[-1,-1]	[2,2]	[3,3]
0.10	-7,3845	-6,7574	-8,0801
0.20	-1,1467	-1,2025	-1,1082
0.30	-0,3377	-0,3243	-0,3575
0.40	-0,2095	-0,1086	-0,0880
0.50	-0,0455	-0,0455	-0,0256
0.60	-0,0369	-0,0236	-0,0085
0.70	-0,0051	-0,0018	-0,0041
0.80	-0,0021	-0,0003	-0,0008
0.90	0,0000	-0,0001	-0,0002
1.00	0,0000	0,0000	0,0000
1.10	0,0000	-0,0001	-0,0002
1.20	-0,0021	-0,0003	-0,0008

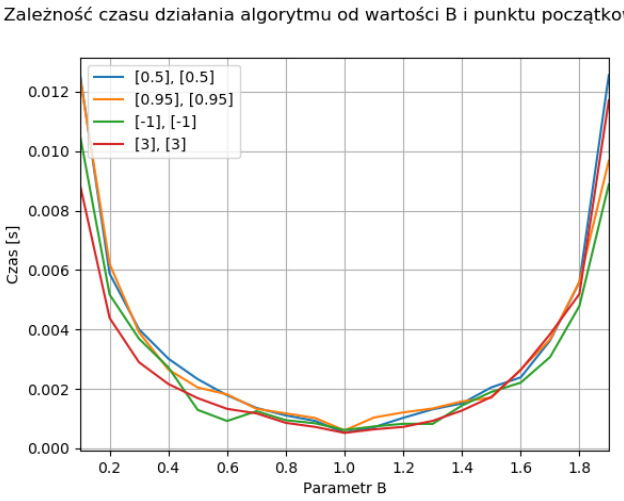
1.30	-0,0051	-0,0018	-0,0041
1.40	-0,0059	-0,0038	-0,0014
1.50	-0,0028	-0,0028	-0,0064
1.60	-0,0098	-0,0051	-0,0114
1.70	-0,0095	-0,0092	-0,0101
1.80	-0,0132	-0,0139	-0,0128
1.90	-0,0202	-0,0185	-0,0221

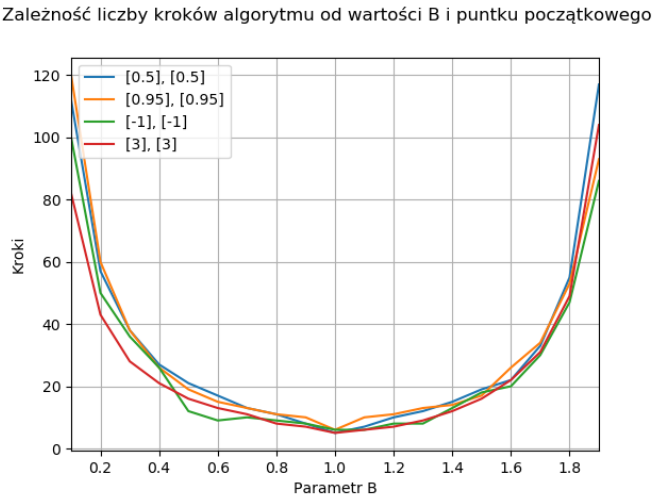
Wzór 2: $f(x) = -x^T x + 1.1\cos(x^T x)$

Funkcja 2 posiada nieskończenie wiele maksimów lokalnych, dlatego algorytm może nie znaleźć maksimum globalnego funkcji, a jedynie inne maksimum lokalne. Zależy to od wartości β oraz punktu początkowego; im bliżej jest punkt do maksimum globalnego tym większa jest szansa na jego poprawne znalezienie.

Czas działania algorytmu:

Czasy działania algorytmu dla funkcji 2, są podobne do czasów algorytmu dla funkcji 1, ale charakteryzuje je dużo większa nieregularność.

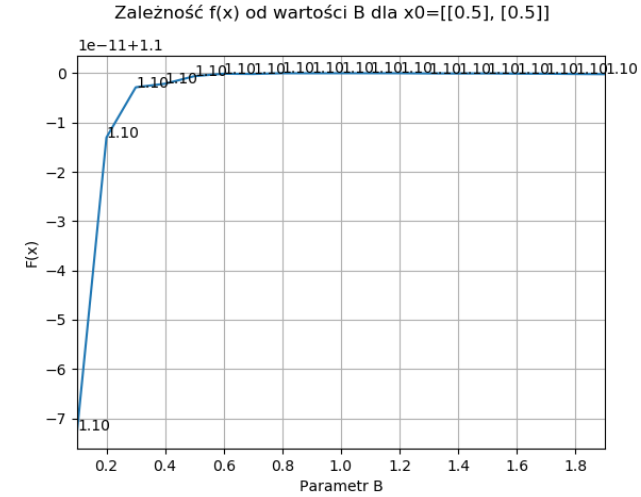


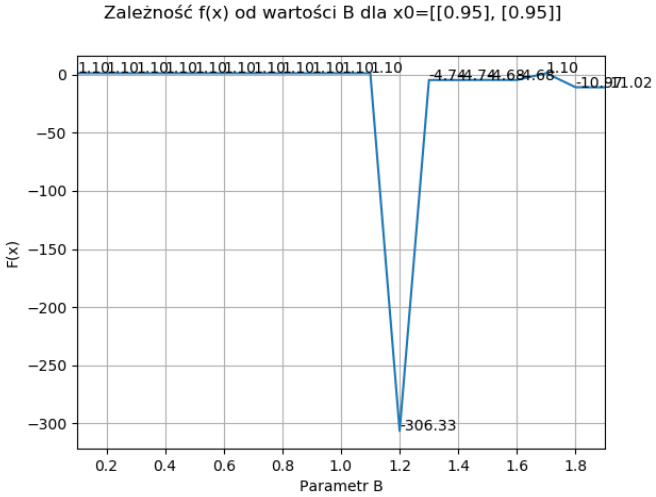


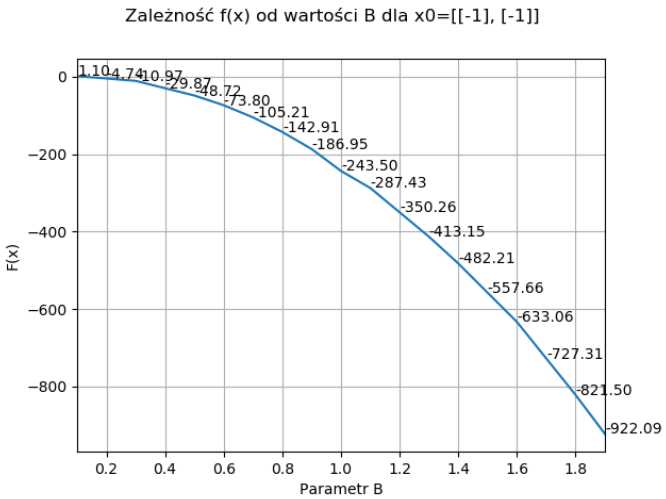
β	Czas działania algorytmu [s]			
	[0.5,0.5]	[0.95,0.95]	[-1,-1]	[3,3]
0.10	0,0118	0,0124	0,0088	0,0092
0.20	0,0059	0,0061	0,0044	0,0047
0.30	0,0039	0,0039	0,0029	0,0031
0.40	0,0027	0,0027	0,0020	0,0023
0.50	0,0021	0,0019	0,0015	0,0017
0.60	0,0017	0,0015	0,0012	0,0014
0.70	0,0013	0,0013	0,0012	0,0010
0.80	0,0011	0,0011	0,0007	0,0008
0.90	0,0008	0,0011	0,0006	0,0006
1.00	0,0005	0,0006	0,0001	0,0001
1.10	0,0007	0,0010	0,0006	0,0008
1.20	0,0010	0,0011	0,0009	0,0008
1.30	0,0012	0,0013	0,0009	0,0010
1.40	0,0015	0,0014	0,0012	0,0015
1.50	0,0020	0,0018	0,0017	0,0018
1.60	0,0023	0,0027	0,0022	0,0023
1.70	0,0034	0,0035	0,0031	0,0034
1.80	0,0056	0,0054	0,0050	0,0053
1.90	0,0120	0,0098	0,0105	0,0113

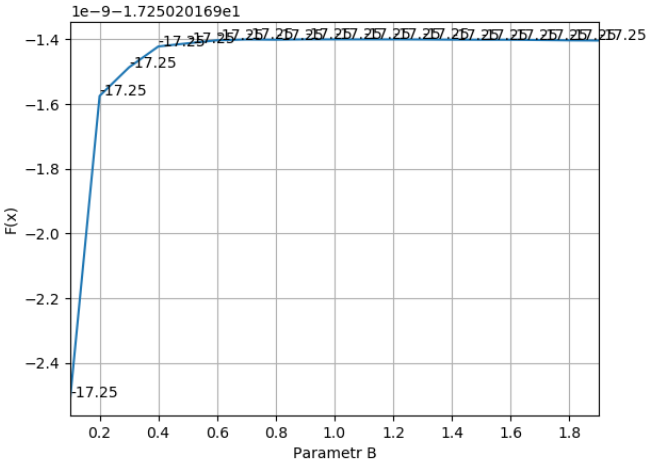
Wyniki algorytmu:

Wyniki algorytmu dla funkcji 2 są znacząco inne niż dla funkcji 1. Dla punktów bardzo blisko maksimum globalnego, algorytm zachowuje się tak, jak dla funkcji 1: jego dokładność rośnie do β=1, a potem maleje. W miarę oddalania się od maksimum globalnego dokładność spada, pojawiają się też przeskoki na inne maksima lokalne funkcji. W pewnej odległości algorytm może już zupełnie nie znaleźć maksimum globalnego, a jedynie inne maksima lokalne.









β	Wyniki algorytmu [f(x)]			
	[0.5,0.5]	[0.95,0.95]	[-1,-1]	[3,3]
0.10	1.10	1.10	1,10	-17,25
0.20	1.10	1.10	-4,74	-17,25
0.30	1.10	1.10	-10,97	-17,25
0.40	1.10	1.10	-29,87	-17,25
0.50	1.10	1.10	-48,72	-17,25
0.60	1.10	1.10	-73,80	-17,25
0.70	1.10	1.10	-105,21	-17,25
0.80	1.10	1.10	-142,91	-17,25
0.90	1.10	1.10	-186,95	-17,25
1.00	1.10	1.10	-243,50	-17,25
1.10	1.10	1.10	-287,43	-17,25
1.20	1.10	-306.33	-350,26	-17,25
1.30	1.10	-4.74	-413,15	-17,25
1.40	1.10	-4.74	-482,21	-17,25
1.50	1.10	-4.68	-557,66	-17,25
1.60	1.10	-4.68	-633,06	-17,25
1.70	1.10	1.10	-727,31	-17,25
1.80	1.10	-10.97	-821,50	-17,25
1.90	1.10	-11.02	-922,09	-17,25

Wnioski:

Metoda Newtona pozwala na osiągnięcie bardzo dobrego przybliżenia dla maksimum, jednak tylko w przypadkach, gdy maksimum znajduje się bardzo blisko punktu początkowego lub funkcja posiada wyłącznie jedno maksimum. W pozostałych przypadkach wyniki uzyskane za jej pomocą mogą być innymi maksimami lokalnymi. Dlatego w przypadku punktów początkowych oddalonych w znaczny sposób od maksimum powinno się na początku używać innego algorytmu np. najszybszego wzrostu, a dopiero później przełączyć się na metodę Newtona, która pozwala na szybsze osiągnięcie celu (przy $\beta=1$, można go osiągnąć już w 1 lub 2 krokach).