UNIVERZA V LJUBLJANI
FAKULTETA ZA MATEMATIKO IN FIZIKO

Finančna matematika – 2. stopnja

Anja Plesec

Gradientni spust

Končno poročilo pri predmetu Matematika z računalnikom

Mentor: prof. dr. Sergio Cabello Justo,
asist. Gašper Domen Romih

Ljubljana, 2022

1 Gradientni spust

Gradientni spust je iterativna metoda za iskanje lokalnega minimuma funkcije. To je algoritem, ki se lahko uporablja na različnih področjih. Cilj pa je imeti algoritem s čim manj iteracij.

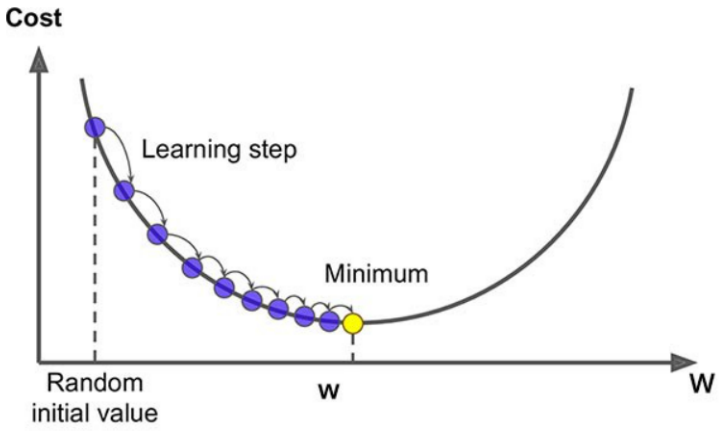
Samo delovanje algoritma je sledeče. Najprej si izberemo začetno točko (initial value) - $x_0 \in \mathbb{R}^n$ in hitrost konvergence (learning rate). Zapišemo stroškovno funkcijo (Loss Function) in jo odvajamo po vseh parametrih. Na ta način dobimo gradient stroškovne funkcije. Izbrano začetku točko nato vstavimo v ta gradient in s pomočjo tega poračunamo velikost naslednjega koraka:

$$\text{velikost koraka} = \nabla f(x_t) \cdot \text{learning rate}$$

Vrednost novega parametra pa dobimo s pomočjo spodnje formule:

$$x_1 = x_0 - \text{velikost koraka} \in \mathbb{R}^n$$

Ko izračunamo vrednost x_1 postopek ponovimo.



Splošen algoritem:

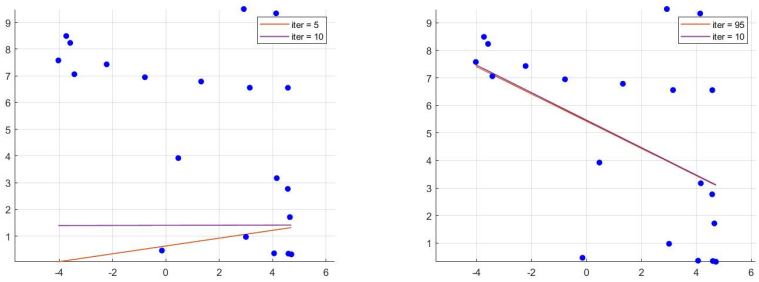
- Izberemo začetno točko $x_0 \in \mathbb{R}^n$.
- Za $t \geq 0$ predpostavimo, da poznamo $x_0, x_1, \ldots, x_t$. Naslednji člen določimo s pomočjo enačbe:
$$x_{t+1} = x_t - \eta \cdot \nabla f(x_t),$$
kjer η predstavlja learning rate, $\nabla f(x_t)$ pa gradient funkcije f v točki x_t.
- Končamo in vrnemo zadnjo iteracijo.

Težave, ki se pri tem algoritmu pojavijo, so ustrezna izbira vhodnih podatkov. Eden izmed teh je "learning rate". Radi bi delali velike korake, kar bi pomenilo manjše število iteracij, vendar se nam lahko zgodi, da s tem zgrešimo našo rešitev in sam algoritem ne vrne prave rešitve. Naslednji izmed vhodnih podatkov je začetna točka x_0. Želimo si jo izbrati tako, da je čim bližje optimalni rešitvi, saj bomo zaradi tega potrebovali manj iteracij.

Projekt bom izdelala v programu Matlab.

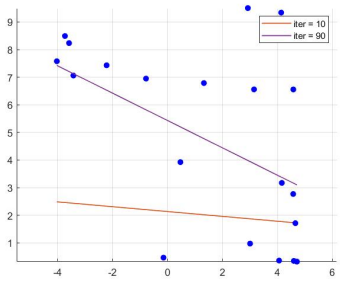
2 Primer uporabe pri linearni regresiji

Gradientni spust doseže optimalno vrednost tako, da na začetku algoritma dela velike korake, ko pa se bliža optimalni vrednosti (minimumu) pa so koraki vedno manjši. To lastnost algoritma lahko potrdita tudi slika 1 in 2. Na prvi sliki torej vidimo, da pet dodatnih iteracij zelo vpliva na premico (razlika med pet in deset iteracij), med tem ko na drugi sliki, kjer gledamo razliko med 95 in 100 iteracij, skoraj ni razlike, saj so koraki, ki jih delamo vedno manjši.

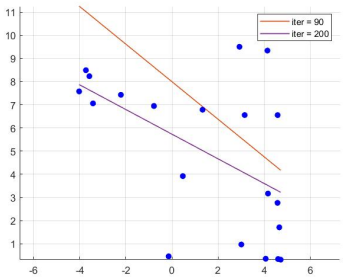


Slika 1: Razlike v velikosti koraka

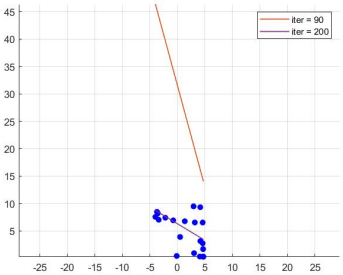
Izvedla sem poskus, kjer sem spreminjala začetne pogoje in opazovala koliko iteracij je potrebno, da se premica približa optimalni rešitvi. V primeru a) sem vzela za parametra naklon in konstanta kar vrednost 1. Ko izvedemo 90 iteracij se zadnja iteracija dobro približa začetni rešitvi. Pri primeru b) sem vzela naklon = -20 in konst = 50. Opazimo, da je v tem primeru potrebnih 200 iteracij. Prav toliko jih je potrebnih v c) primeru, kjer sem vzela naklon = -200 in konst = 500.



(a) naklon = 1, konst = 1



(b) naklon = -20, konst = 50



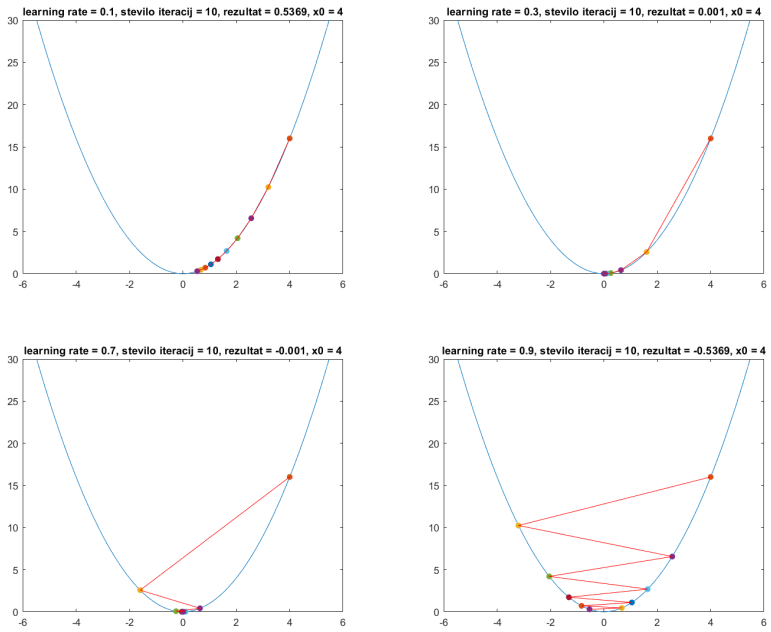
(c) naklon = -200, konst = 500

Slika 2: Konvergenca pri različnih začetnih pogojih

3 Iskanje minimuma podane funkcije

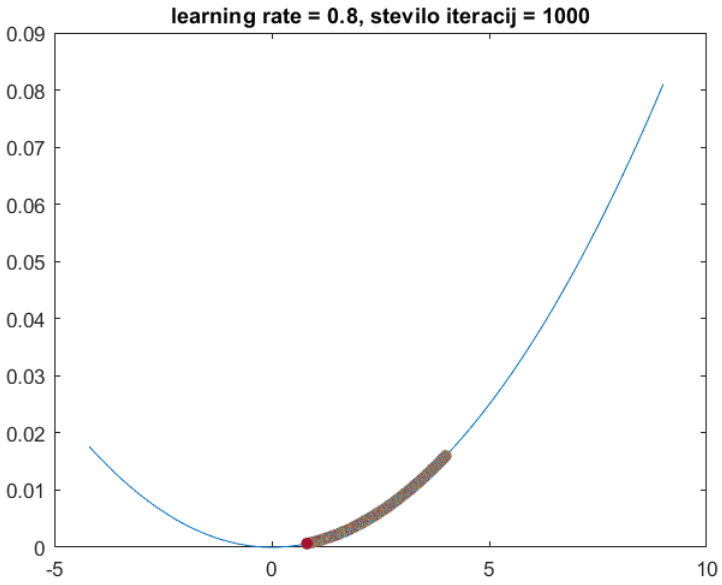
3.1 Iskanje minimuma funkcije ene spremenljivke

Iz spodnjih slik lahko opazimo, da če izberemo preveliko učno stopnjo nam približki skačejo iz ene strani minimuma na drugo stran, med tem ko če izberemo manjšo nam približki ne skačejo. Opazimo, da pri preveliki in premajhni učni stopnji algoritem potrebuje več iteracij. Torej moremo izbrati ravno pravšnjo za našo metodo. Za začetni približek vzamemo $x_0 = 4$



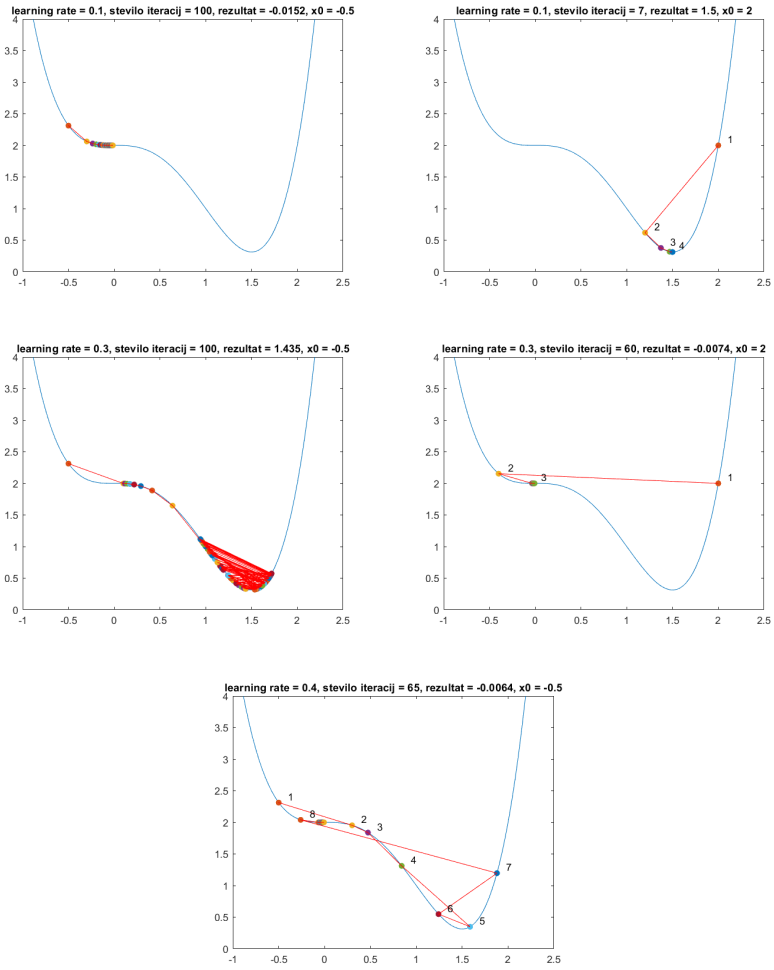
Slika 3: Razlike glede na izbrano učno stopnjo

Sam algoritem ni primeren za funkcije, ki je nekaj časa skoraj konstantna oz je naklon tangente skoraj ničelni. Pri takih funkcijah sam algoritem potrebuje veliko več iteracij, da pride do minimuma. Na spodnjem grafu funkcije $\frac{x^2}{1000}$ vidimo, da je kljub dobro izbrani učni stopnji končna vrednost približka za minimum 0.8066, ker je še vedno daleč stan od samega minimuma naredili pa smo 1000 iteracij.



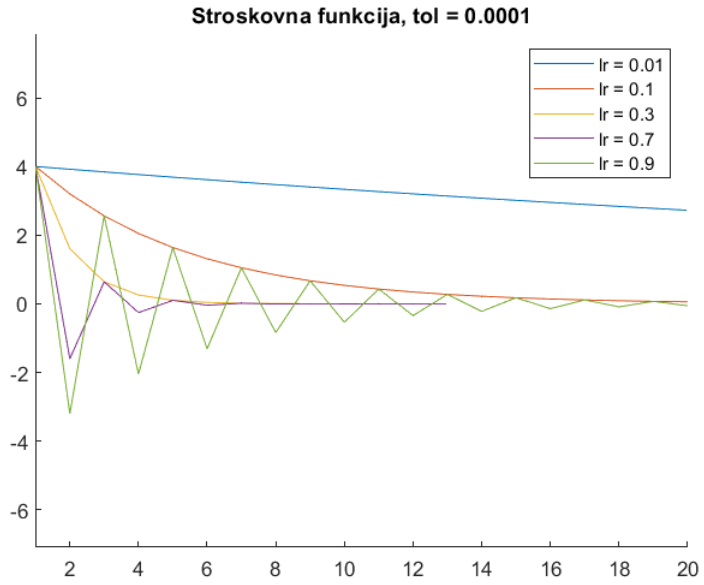
Slika 4: Izravnana funkcija

Zanimiv primer je naslednja funkcija $x^4 - 2x^3 + 2$. Težave algoritmu povzroči sedlo. Lahko vidimo kako izbira začetne točke in učne stopnje vplivata na rešitev algoritma. Seveda v primerih sedelne točke ne moremo trditi, da nam algoritem res vrne globalni minimum.



Slika 5: Razlike glede na izbrano učno stopnjo in začetni približek

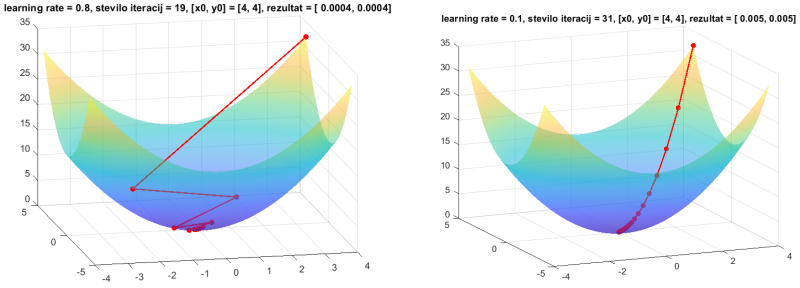
Na spodnjem grafu lahko vidimo, katera učna stopnja je najboljša za funkcijo x^2. Opazim, da sta to vrednosti 0.3 in 0.7, ki se dovolj približata ničli v trinajstih iteracijah. Manjša kot je učna stopnja dlje časa potrebuje da skonvergira, prav tako lahko to opazimo za velike učne stopnje.



Slika 6: Izravnana funkcija

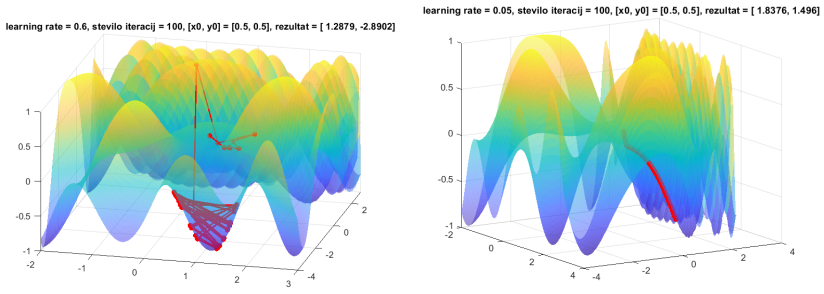
3.2 Iskanje minimuma funkcije več spremenljivk

Podobno kot pri iskanju minimuma funkcije ene spremenljivke si lahko za začetek pogledamo enostaven primer, torej funkcijo $x^2 + y^2$. V primeru, ko je učna stopnja enaka 0.8 lahko opazimo, da tako x koordinate kot y koordinate skačejo iz pozitivnih v negativna števila in dosežejo minimum v 19 iteracijah, med tem ko pri učni stopnji 0.1 so vrednosti obeh koordinat pozitivne in lepo padajo k minimumu. V tem primeru pa je potrebnih 31 iteracij.



Slika 7: Razlike glede na izbran learning rate

Če si pogledamo funkcijo, ki ima sedla in več lokalnih ter globalnih minimumov opazimo podobne stvari, kot v primeru funkcij ene spremenljivke. Samo iskanje minimuma je zelo odvisno od izbire začetnega približka in učne stopnje. Na spodnjih grafih opazimo, da pri različnih učnih stopnjah metoda najde različne minimume. Prav tako je razlika v konvergenci, nekje lepo pada proti minimumu, nekje pa skače iz ene strani minimuma na drugo stran.



Slika 8: Razlike glede na izbran learning rate

4 Zaključek

....

Literatura

[1] Nisheeth K. Vishnoi (2020) *Algorithms for Convex Optimization*. Pridobljeno 24. 4. 2022 iz `https://convex-optimization.github.io/`.

[2] Robert Kwiatkowski (2021) *Gradient Descent Algorithm — a deep dive*. Pridobljeno 30. 4. 2022 iz `https://towardsdatascience.com/gradient-descent-algorithm-a-deep-dive-cf04e8115f21`.