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.

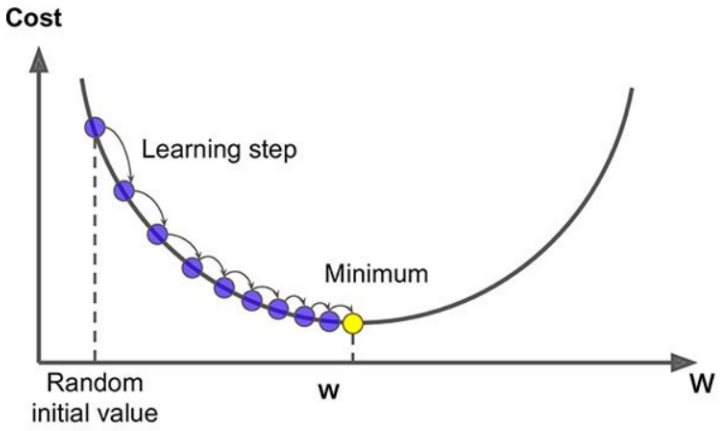
Samo delovanje algoritma je sledeče. Najprej si izberemo začetno točko (initial value) - $x_0 \in \mathbb{R}^n$ in hitrost konvergence (learning rate). Zapišemo stroškovno funkcijo (Loss Function) in jo odvajamo po vseh parametrih. Na ta način dobimo gradient stroškovne funkcije. Izbrano začetku točko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

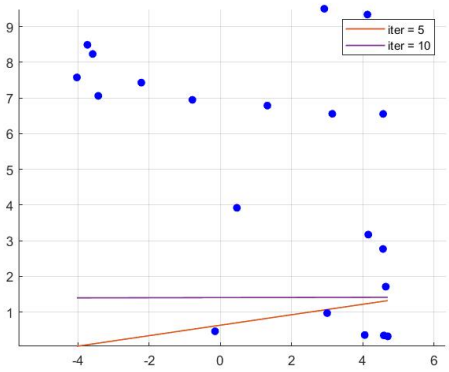
- Izberemo začetno točko $x_0 \in \mathbb{R}^n$.
- Za $t \geq 0$ predpostavimo, da poznamo $x_0, x_1, \ldots, x_t$. Naslednji člen določimo s pomočjo enačbe:
$$x_{t+1} = x_t - \eta \cdot \nabla f(x_t),$$
kjer η predstavlja learning rate, $\nabla f(x_t)$ pa gradient funkcije f v točki x_t.
- Končamo in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

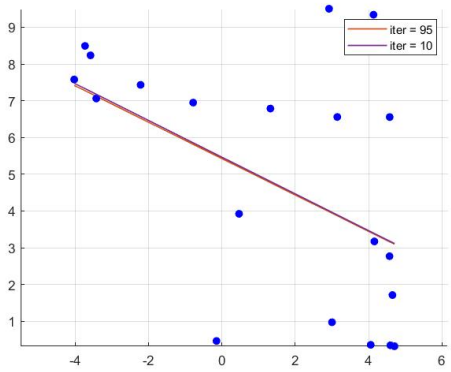
Projekt bom izdelala v programu Matlab.

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.

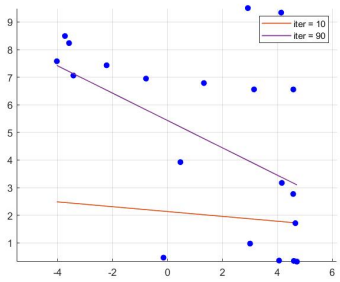


Slika 1: Število iteracij je 5 in 10

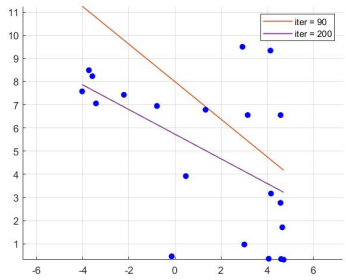


Slika 2: Število iteracij je 95 in 100

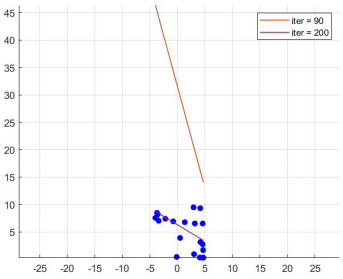
Izvedla sem poskus, kjer sem spreminjala začetne pogoje in opazovala koliko iteracij je potrebno, da se premica približa optimalni rešitvi. V primeru a) sem vzela za parametra naklon in konstanta kar vrednost 1. Ko izvedemo 90 iteraij se zadnja iteracija dobro približa začetni rešitvi. Pri primeru b) sem vzela naklon = -20 in konst = 50. Opzimo, 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 3: Konvergenca pri različnih začetnih pogojih