Poročilo - domača naloga 4

Urh Primožič

17. junij 2023

1 Odkrivanje enačb in uporaba predzanja

1.1 Predznanje

Uporabljeno predznanje sem vključil v ustvarjanje novih značilk. Namesto značilk T_w in T_a sem uvedel značilko $\Delta T = T_w - T_a$. Poleg značilke θ sem uvedel $\sin\theta$, saj sledi domenskemu predzanju (doseže največjo vrednost za $\theta = \frac{\pi}{2}$)). Domensko znanje sicer ne nodi nobenega zagotovila za uporabnost spremenljivke $\sin\theta$, a iz prakse fizikalnih enačb pričakujem prisotnost kotnih funkcij. Prav tako sem uvedel spremenljivko η^{-1}.

Analiza enot v tem primeru ni koristila, saj so v nalogi dane le enote temperature.

1.2 Metode

Naloge sem se sprva lotil s ProGEDom in preprosto gramatiko oblike

$$\begin{array}{lcl} V & \to & VF \ \ [0.6] \mid cF \ \ [0.4] \\ F & \to & \theta \ \ [0.1] \mid \sin\theta \ \ [0.3] \mid \Delta T \ \ [0.3] \mid \eta^{-1} \ \ [0.3]. \end{array}$$

, vendar rezultati niso bili najboljši. Najboljša enačba, ki jo je tvorila zgornja gramatika je bila

$$0.00209 * deltaT * *2$$

, njena povprečna kvadratna napaka pa . Razlog je verjetno v strukturi gramatik; namreč že majhne gramatike tvorijo velik prostor funkcij in posledično veliko slabih enačb. Rezultati bi se verjetno izboljšali, če bi število tvorjenih enačb povečali za nekaj redov velikosti.

Prav tako sem poizkusil z redko linearno regresijo (laso), vendar je bila dobljena enačba predolga z obupno napako 71820990853669.36. Boljše rezultate bi lahko dosegel, če bi v linearno regresijo nesel manj značilk. (metoda Laso ni analitična, zato je dimenzija prostora možnih parametrov zelo pomembna. Večja kot je, manj natančno lahko izberemo parametre v fiksnem času.) Glede na ostale rezultate bi bile verjetno dovolj le polinomske kombinacije stopnje 4.

Na značilkah $DeltaT$, $\sin\theta$ in η^{-1} sem pognal še metodo PySR. Ta je vrnila enačbo

$$0.00502\Delta T^2 \sin\theta(0.94987 - 0.63199\eta)$$

s povprečno kvadratno napako enako 0.013. Očizno se je intuicija za uporabo značilke $\sin\theta$ splačala. Obratno sorazmernost η in Q pa enačba izrazi z razliko, torej je bila uvedba spremenljivke η^{-1} v tem primeru odveč.

Toplotni tok se v osnovnih fizikalnih enačbah večkrat izrazi s produktom drugih količin. (Recimo $\frac{Q}{\Delta t} = -kA\frac{\Delta T}{\Delta x}$. Zato se mi je zdelo smiselno poiskati enačbo v taki obliki. S pomočjo metode `sklearn.optimize.minimize` sem poiskal optimalne parametre enačbe $A\Delta T^B \sin\theta^C \eta^{-D}$ in dobil enačbo

$$0.002\Delta T^{2.052}\ sin\theta^{1.027}\eta^{-0.226}$$

s kvadratno napako 0.038.

Očitno je najboljšo enačbo glede na napako vrnila metoda PySR, kar je pričakovano, saj je metoda plod večletnega razvoja.

Zaupanje enačbam Zadnji dve enačbi imata podobno obliko, le obratna sorazmernost z η je predstavljena na drugačen način. Kljub

2 Verjetnostne gramatike in posodabljanje verjetnosti

Za fiksen podatkovi nabor in fiksno gramatiko vektor verjetnosti prepisovalnih pravil $p = (p_1, \ldots, p_N)$ parametrizira porazdelitev $P = P_p$ nad funkcijami, ki jih tvori gramatika. Dobra porazdelitev funkcij bo funckijam z veliko napako dodelila majhno verjetnost.

Primer - racionalna gramatika Racionalna gramatika

$$\begin{array}{lcl} E & \to & E + F \ \ [p_1] \mid E - F \ \ [p_2] \mid F \ \ [p_3] \\ F & \to & T \ \ [p_4] \mid FT \ \ [p_5] \mid F/T \ \ [p_6] \\ T & \to & V \ \ [p_7] \mid E \ \ [p_8] \mid \sin E \ \ [p_9] \\ V & \to & x_1 \ \ [p_1 0] \mid \cdots \mid x_n \ \ [p_{9+n}] \end{array}$$

je parametrizirana z verjetnostnim vektorjem $p = (p_1, \ldots, p_{9+n})$.

2.1 Funckija izgube porazdelitve

Naj bodo $f_1, f_2, \ldots$ funkcije, ki jih tvori fiksna gramatika G. Definiramo funkcijo izgube porazdelitve

$$L(p) = \sum_{i=1}^{\infty} \mathrm{ERROR}(f_i) P_p(f_i)$$

kjer je $ERROR(f)$ smiselna napaka funckije f na fiksnih podatkih, $P_p(f)$ pa verjetnost, da gramatika z verjetnostnim vektorjem p tvori funkcijo f. Vrednosti p, za katere je $L(p)$ majhna, parametrizirajo porazdelitve, ki vračajo slabe funkcije z majhno verjetnostjo. Naš cilj bo iterativno minimalizirati funkcijo L.

Izračun funkcije L ni enostaven. Namesto L bomo v algoritmu računali njen približek

$$\hat{L}(p) = \sum_{f \in S} \text{ERROR}(f) P_p(f)$$

kjer je S končna množica funkcij iz jezika gramatike G, vzorčenih glede na porazdelitev $P(p)$.

2.2 Algoritem

Algoritem iterativno implementira gradienti spust za minimalizacijo funckije L.

1. Najprej vzorčimo množico $S \subset L(G)$ glede na porazdelitev $P(p)$.
2. Poračunamo napake funkcij f iz množice S na podatkih. Če je katerakoli izmed napak dovolj majhna, iteracijo prekinemo in vrnemo funkcijo s to napako in gramatiko s trenutno verjetnostno porazdelitvijo.
3. Posodobimo verjetnostni vektor p na $p \leftarrow p - \gamma \nabla L(p)$ in se vrnemo na prvi korak.

Računanje $\nabla L(p)$ Verjetnosti niso popolnoma poljubna števila. Veljajo zveze $p_3 = 1 - p_1 - p_2, p_3 + 6 = 1 - p_5 - p_4, p_9 = 1 - p_8 - p_7, p_n = 1 - \sum_{i=10}^{9+n-1} p_i$. Funkcija L je torej odvisna le od vrednosti $p_1, p2, p_4, \ldots, p_{n-1}$. Velja zveza

$$\frac{\partial L}{\partial p_i} = \sum_{f \in S} \text{ERROR}(f) P_p(f) \left(\frac{G_i(f)}{p_i} - \frac{G_j(f)}{p_j}\right)$$

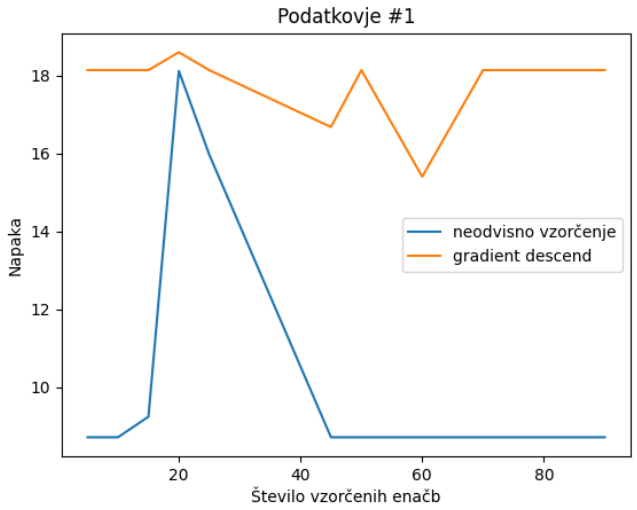
kjer je $P(f) = p_1^{G_1(f)} \cdots p_n^{G_n(f)}$, j pa najmanjši izmed indeksov $3, 6, 9, n$, da velja $i < j$. Implementacija algoritma je v skripti `ProGEDescend.py`.

2.3 Delovanje

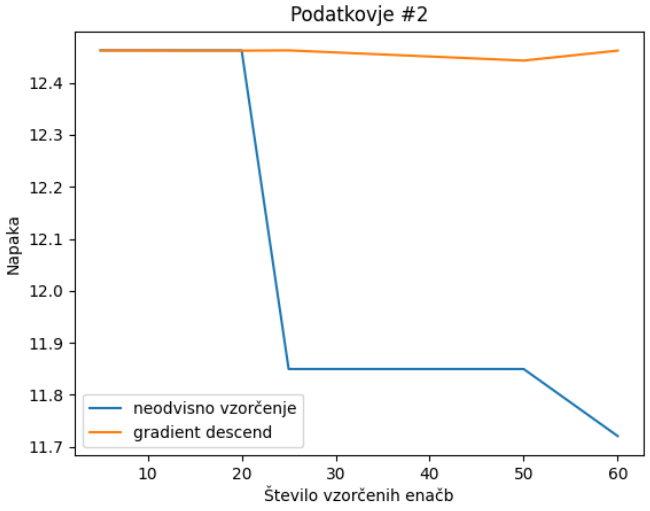
Nastavitve eksperimenta Za namene risanja grafa sem primerjal klasični ProGED z gradientim spustom za $|S| = 1$ in $|S| = 10$. Nekaj rezultatov za $|S| = 100$ je v `naloga2.ipynb`.

Algoritem verjetnosti spreminja in se umika prej zaznanim slabim funkcijam. Vseeno pa to ne pomeni, da zagotovo vrača dobre funkcije z veliko verjetnostjo. ProGED sam ne upošteva struktur enačb in deluje na sistemu masovnega preiskovanja prostora. Tega preiskovanja se ne moremo ogniti.

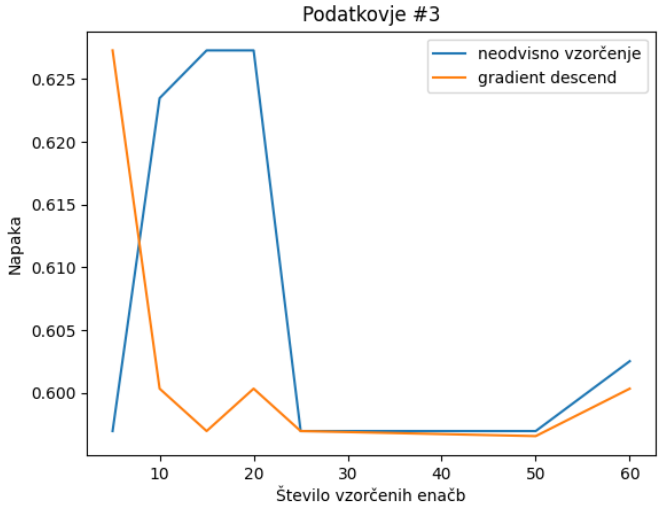
Izboljšave Očitna izboljšava gradiente metode bi bila matematična izpeljava verjetnosti, da bo $\hat{L}$ blizu L. Prostor za izboljšave je predvsem v analizi. Primerjava glede na število zgeneriranih enačb mogoče ni optimalna, saj klasičen ProGed naredi eno vzorčenje, gradient descent pa novo vzorčenje vsako iteracijo. Rešitev z $|S| = 1$ je bila enostavna za implementacijo, vendar se za male



Slika 1: Enačba $y = x_1 - 3x_2 - x_3 - x_5$. Vidna premoč klasične metode.



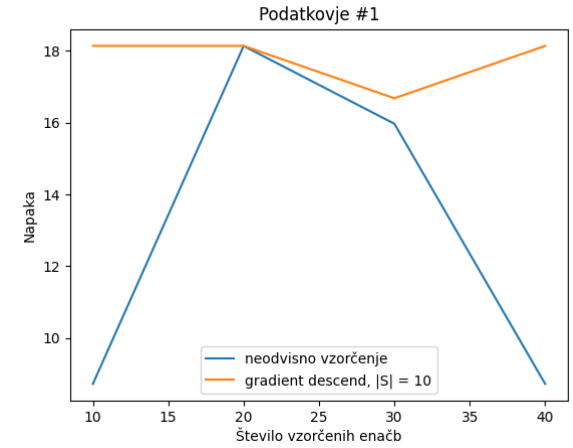
Slika 2: Enačba $y = x_1^5 x_2^3$. Neodvisno vzorčenje je pri dovolj velikih vzorcih postalo boljše.

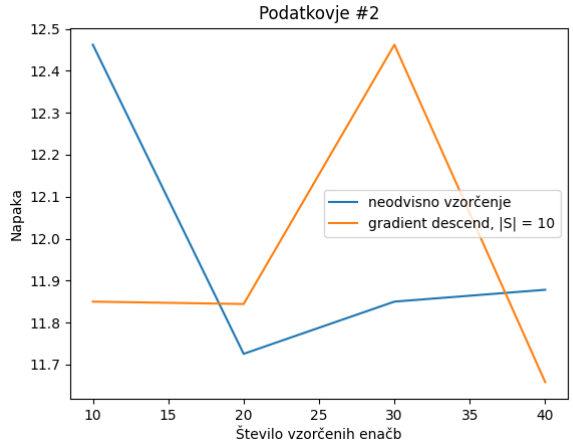


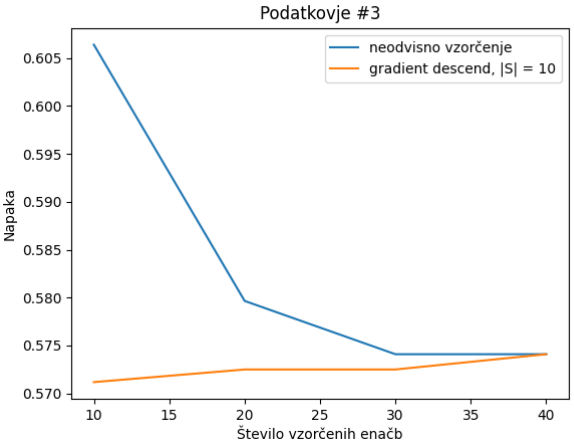
Slika 3: Enačba $y = \sin x_2 + \sin(x_2/x_1^2)$. Premoč gradiente metode.

vrednosti $|S|$ $\hat{L}$ lahko poljubno oddaljuje od L. Boljša bi bila analiza, kjer bi konstano $|S|$ nastavili na veliko vrednost.

Zanimiva bi bila tudi primerjava z drugimi metodami adaptacije verjetnosti. Poleg tega je primerjava na prenosniku omejena s strojno močjo računalnika. Zanimiva bi bila analiza na večji skali; zdi se namreč, da se na grafih gradientega spusta pojavljajo neki trendi padanja.







Slika 4: Rezultati za $|S| = 10$