Logistická regresia – predpoveď srdcovej choroby 104 BPM

Samuel Dokupil $\frac{1}{5}$
Róbert Kendereš $\frac{1}{5}$
Natália Krebesová $\frac{1}{5}$
Pavol Lucina $\frac{1}{5}$
Vanda Vladová $\frac{1}{5}$

11.5.2025

Obsah

Úvod

V tomto projekte sme sa venovali modelovaniu srdcového ochorenia u pacientov na základe výsledkov ich vyšetrení. Hlavným cieľom bolo formulovať a riešiť optimalizačnú úlohu (definovaná v nasledujúcej kapitole), ktorá vzniká pri odhade parametrov logistickej regresie. Úlohu sme riešili pomocou troch rôznych optimalizačných metód:

- gradientná metóda s konštantnou dĺžkou kroku,
- gradientná metóda s približne optimálnou dĺžkou kroku, kde sme dĺžku kroku hľadali pomocou algoritmu `backtracking` a
- gradientná metóda s optimálnou dĺžkou kroku, kde sme využívali metódu `zlatého rezu`.

Dáta ktoré sme používali boli rozdelené na trénovaciu a testovaciu zložku ktoré pozostávali zo šiestich stĺpcov, kde prvých päť (`age, trestbps, chol, thalach, oldpeak`) boli údaje z vyšetrení pacientov a posledný stĺpec (`target`) bola binárna závislá premenná ktorá určovala prítomnosť (1) alebo neprítomnosť (0) srdcovej choroby.

Použité zložky sú v nasledujúcich kapitolách zapísané ako matica $U \in \mathbb{R}^{m\times 6}$, kde každý riadok u^i predstavuje jedného pacienta a u^i_j reprezentuje j-ty nameraný údaj daného pacienta a vektor $v \in \mathbb{R}^m$, kde v^i je jedno pozorovanie závislej premennej i-teho pacienta, kde m je počet pozorovaní v danej zložke, $m = 253$ pre trénovaciu a $m = 50$ pre testovaciu zložku. Matice U sú rozšírené o stĺpec jednotiek kvôli konštantnému členu regresie.

Cieľom optimalizácie bolo teda nájsť vektor parametrov $x \in \mathbb{R}^6$, ktorý minimalizuje účelovú funkciu odvodenú z modelu logistickej regresie, ktorý predpovedá pravdepodobnosť prítomnosti srdcovej choroby podľa vzorca

$$g(x^T u) = \frac{1}{1+e^{-x^T u}},$$

pričom chceme, aby tieto predpovede boli čo najpresnejšie vzhľadom na skutočné hodnoty v. Použité metódy sme nakoniec porovnávali podľa počtu iterácií, času za ktorý sa našlo minimum a podielu správne klasifikovaných pacientov na testovacej zložke, ďalej značené ako presnosť (`accuracy`).

A) Úprava tvaru optimalizačnej úlohy

Máme dané

$$u = \begin{pmatrix} 1 \\ u_1 \\ \vdots \\ u_5 \end{pmatrix} \qquad x = \begin{pmatrix} x_0 \\ ux_1 \\ \vdots \\ x_5 \end{pmatrix} \qquad x^T u = x_0 + x_1 u_1 + \cdots + x_5 u_5$$

$$g(x^T u) = \frac{1}{1 + e^{-x^T u}} \tag{1}$$

a otimalizačnú úlohu

$$\min \left\{ J(x) = -\sum_{i=1}^{m} v^i \ln \Big(g(x^T u^i) \Big) + (1 - v^i) \ln \Big(1 - g(x^T u^i) \Big) \,\middle|\, x \in \mathbb{R}^6 \right\} \tag{2}$$

kde

$$u^i = (1, u_1^i, \ldots, u_5^i)^T$$

—

Máme ukázať, že predpis $J(x)$ sa dá dosadením (1) do (2) zjednodušiť na:

$$\min \left\{ J(x) = \sum_{i=1}^{m} (1 - v^i) x^T u^i + \ln \Big(1 + e^{-x^T u^i} \Big) \,\middle|\, x \in \mathbb{R}^6 \right\} \tag{3}$$

—

Po dosadení a úpravách dostaneme

$$\begin{aligned} J(x) &= -\sum_{i=1}^{m} v^i \ln \left(\frac{1}{1 + e^{-x^T u^i}} \right) + (1 - v^i) \ln \left(1 - \frac{1}{1 + e^{-x^T u^i}} \right) \\ &= -\sum_{i=1}^{m} v^i \ln \left(\frac{1}{1 + e^{-x^T u^i}} \right) + (1 - v^i) \ln \left(\frac{e^{-x^T u^i}}{1 + e^{-x^T u^i}} \right) \\ &= -\sum_{i=1}^{m} v^i \Big(\ln 1 - \ln(1 + e^{-x^T u^i}) \Big) + (1 - v^i) \Big(\ln e^{-x^T u^i} - \ln(1 + e^{-x^T u^i}) \Big) \\ &= -\sum_{i=1}^{m} v^i \ln 1 - v^i \ln(1 + e^{-x^T u^i}) + \ln e^{-x^T u^i} - v^i \ln e^{-x^T u^i} - \ln(1 + e^{-x^T u^i}) + v^i \ln(1 + e^{-x^T u^i}) \\ &= -\sum_{i=1}^{m} \ln e^{-x^T u^i} - v^i \ln e^{-x^T u^i} - \ln(1 + e^{-x^T u^i}) \\ &= -\sum_{i=1}^{m} (1 - v^i)(-x^T u^i) - \ln(1 + e^{-x^T u^i}) \\ &\boxed{= \sum_{i=1}^{m} (1 - v^i)(x^T u^i) + \ln(1 + e^{-x^T u^i}),} \end{aligned} \qquad \square$$

čo sa zhoduje s požadovaným tvarom $J(x)$ v optimalizačnej úlohe

$$\min \left\{ J(x) = \sum_{i=1}^{m} (1 - v^i)(x^T u^i) + \ln(1 + e^{-x^T u^i}) \,\middle|\, x \in \mathbb{R}^6 \right\}.$$

B) Predpis gradientu $\nabla J(x)$

Cieľom tejto úlohy bolo nájsť prvky gradientu $\nabla J(x)$ funkcie $J(x)$ v tvare (3).
Z definície gradientu máme pre jednotlivé prvky gradientu

$$\nabla J(x)_j = \frac{\partial}{\partial x_j} \sum_{i=1}^{m} (1-v^i)(x^T u^i) + \ln(1+e^{-x^T u^i}) \qquad j = 0, \dots 5$$

$$= \sum_{i=1}^{m} \frac{\partial}{\partial x_j}(1-v^i)(x^T u^i) + \frac{\partial}{\partial x_j} \ln(1+e^{-x^T u^i})$$

$$= \sum_{i=1}^{m} \frac{\partial}{\partial x_j}(1-v^i)(x_0 u_0^i + x_1 u_1^i + \cdots + x_5 u_5^i) + \frac{\partial}{\partial x_j} \ln\left(1+e^{-(x_0 u_0^i + x_1 u_1^i + \cdots + x_5 u_5^i)}\right)$$

$$= \sum_{i=1}^{m} (1-v^i) u_j^i + \frac{1}{1+e^{-x^T u^i}}(-u_j^i e^{-x^T u^i})$$

$$= \sum_{i=1}^{m} u_j^i \left((1-v^i) - \frac{e^{-x^T u^i}}{1+e^{-x^T u^i}}\right)$$

$$= \sum_{i=1}^{m} u_j^i \left((1-v^i) - \left(1 - \frac{1}{1+e^{-x^T u^i}}\right)\right)$$

$$= \sum_{i=1}^{m} u_j^i \left(\frac{1}{1+e^{-x^T u^i}} - v^i\right),$$

dosadením (1) dostaneme

$$\nabla J(x)_j = \sum_{i=1}^{m} u_j^i \left(g(x^T u^i) - v^i\right)$$

a teda celkový gradient je

$$\boxed{\nabla J(x) = \sum_{i=1}^{m} u^i \left(g(x^T u^i) - v^i\right) = u^T \left(g(ux) - v\right),} \qquad \square$$

kde u je $m \times 6$ matica pozorovaní prediktorov a v je $m \times 1$ vektor pozorovaní závislej premennej.

C) Gradientná metóda s konštantným krokom

V rámci riešenia optimalizačnej úlohy sme implementovali gradientnú metódu s konštantným krokom. Hlavným cieľom bolo nájsť vhodnú hodnotu konštantného kroku c, ktorá zabezpečí:

- dostatočne nízku hodnotu funkcie $J(x)$,
- rozumný počet iterácií na konvergenciu,
- primeraný čas výpočtu.

Parametre metódy:

- počiatočný bod: $x_0 = [-2.28, 0, 0, 0, 0, 0]^T$,
- účelová funkcia $J(x)$ a jej gradient $\nabla J(x)$
- dĺžka kroku c, ktorú sme nastavovali,
- Tolerancia ε_1 pre kritérium konvergencie: $\|\nabla J(x)\| < \varepsilon_1$ (pre všetky metódy použitá $\varepsilon_1 = 10^{-3}$)
- Tolerancia ε_2 pre alternatívne zastavenie: $\| - c \cdot \nabla J(x)\| < \varepsilon_2$ (pre túto metódu bola vybraná $\varepsilon_1 = 10^{-7}$)
- Maximálny počet iterácií: 10^7

Alternatívne zastavenie bolo zavedené z dôvodu dlhého behu metódy a pomalej konvergencii. Princípom je zastaviť optimalizáciu, ak sa nový odhad x^{k+1} nelíši od predošlého x^k o viac než ε_2, čo sme zapísali ako

$$\|x^k - c \cdot \nabla J(x) - x^k\| = \| - c \cdot \nabla J(x)\| < \varepsilon_2.$$

Postup

Hodnota kroku c bola hľadaná kombináciou:

- hrubého testovania krokov 10^{-5}, 10^{-6}, 10^{-7},
- a binárneho vyhľadávania v okolí najlepších výsledkov.

Výsledky

Tabuľka 1: Výsledky gradientnej metódy s konštantným krokom

Veľkosť kroku c	Čas výpočtu [s]	Iterácie	Hodnota $J(x^*)$
1×10^{-5}	311.0	10 000 000	3793.19
1×10^{-6}	308.8	10 000 000	366.81
1×10^{-7}	22.2	701 051	133.97

Z testovaných hodnôt sa ako najefektívnejšia ukázala voľba $c = 10^{-7}$. Tento krok poskytol výrazne nižšiu hodnotu cieľovej funkcie $J(x)$ a zároveň vyžadoval podstatne kratší čas výpočtu a menej iterácií v porovnaní s väčšími krokmi. Väčšie kroky viedli k pomalšiemu znižovaniu hodnoty $J(x)$ a vyžadovali veľké množstvo iterácií, bez dosiahnutia optimálnej presnosti.

Porovnanie s výsledkami knižnice `scipy.optimize`

- Optimalizované hodnoty x z `scipy.optimize`:

$$[-2.2891, -0.0083, -0.0159, -0.0004, 0.0382, -0.5912],$$

- Optimalizované hodnoty x z gradientnej metódy s konštantným krokom:

$$[-2.2862, -0.0085, -0.0161, -0.0004, 0.0383, -0.5687].$$

Rozdiel medzi výsledkami gradientnej metódy s konštantým krokom $c = 10^{-7}$ a výsledkami získanými pomocou knižnice `scipy.optimize` je veľmi malý. Knižnica `scipy.optimize` však dosiahla riešenie za **17 iterácií**, zatiaľ čo gradientná metóda s konštantným krokom pri optimalizácii s krokom $c = 10^{-7}$ dosiahla výsledky po **701,051 iteráciách**.

Nakoniec sme si povedali, že c ide ešte zlepšiť a preto sme hľadali lepšiu hodnotu. Začali sme od najlepšej doteraz nájdenej, čiže 10^{-7} a postupovali sme nalsedovnými krokmi.

1. Zvolíme si c o 0.5 desatinného miesta väčšie.
2. Overíme, či metóda stále konverguje a či sa zmenšil počet iterácii. Ak áno pokračujeme. Ak nie, zrušíme tento pokus a ideme na krok 3.
3. Zvolíme si c o 0.5 desatinného miesta menšie. Ak sa metóda zlepšila pokračujeme, inak zväčšíme hodnotu späť a ideme krok 1.
4. "Vnoríme"sa o desatinné miesto hlbšie.

Tento jednoduchý algorimtus je niečo ako binárne vyhľadávanie. Vykonávali sme ho ručne a došli sme k $c = 4.18857 \times 10^{-7}$. S týmto krokom zbehne gradientná metóda s konštantným krokom za 408844 iterácii. Vo zvyšku reportu sa už porovnávajú výsledky s použitím tejto hodnoty.

D) Približne optimálna dĺžka kroku (backtracking)

Ako grad_J_eval sme využili analyticky odvodený gradient z časti b). Experimentovanie s parametrami pre gradientnú metódu bolo zdĺhavé, ale nakoniec sme našli alpha=0.1, delta=0.01 a lambda_0=0.3. Pri použití vysokej lambda_0 sa úloha správala nestabilne, naopak pri nízkej sa vôbec nehýbala k optimu.

Model dosahoval presnosť predikcie 0.7 až v približne 20000. iterácii. Optimalizácia skončila v 22902. iterácii a to až vďaka druhému zastavovaciemu kritériu založenom na kontrolovaní rozdielu medzi x^k a x^{k+1}, keďže prvé kritérium by metódu nezastavilo, nakoľko norma gradientu poslednej iterácie bola okolo 0.33 a túto hodnotu sa nám nepodarilo signifikantne zlepšiť.

Alternatívne zastavovacie kritérium pre rozdiel medzi x^k a x^{k+1} pre túto metódu malo tvar

$$||x^k + \lambda^k \nabla J(x^k) - x^k|| < \varepsilon_2,$$

čím sme vlastne iba zastavovali optimalizáciu ak sa nový odhad minima nezmenil o viac než zvolené ε_2, čím sme skrátili čas behu. Ako najvhodnejšia voľba pre alternatívne kritérium ε_2 pre backtracking sa ukázalo $\varepsilon_2 = 10^{-8}$.
Najmenšia a najväčšia použitá lambda sú 3×10^{-8} a 3×10^{-6}.

Funkčná hodnota sa veľmi rýchlo priblížila k optimu 133.961885 (kvôli tomu sme zvolili logaritmickú mierku pre iterácie), ale presnosť predikcie choroby rástla odosť pomalšie, ale nakoniec docielila 0.7.

Norma gradientu sa ani v optime nepriblížila na toleranciu ε_1, čiže sme zastavovaciu podmienku museli odvodiť od niečoho iného. Pravdepodobne to značí nie úplne ideálne parametre, ale nájdené optimum je dostatočne blízko k výsledku z minimize.

E) Optimálna dĺžka kroku (golden section)

Gradientná metóda s hľadaním dĺžky kroku pomocou zlatého rezu bola rýchlejšia a lepšia, dosahujúca presnosť predikcie 0.7. Ako aj pri hľadaním dĺžky kroku backtrackingom, aj tu bola vysoká citlivosť na parametre súvisiace s hľadaním dĺžky kroku. V zadaní bolo spomenuté, že si máme uložiť minimálnu a maximálnu hodnotu λ z predchádzajúcej metódy a rozšíriť tento interval. Na rozšírenie sme použili parameter r, ktorým sme interval rozšírili na $(\frac{1}{r}\lambda_{min}, \lambda_{max} r)$. Pri neideálnom optime $r = 10$ bola presnosť oscilujúca, vybrané lambdy boli buď pravidelne väčšie ako by byť mali alebo šlo o inú nestabilitu. Výsledná norma gradientu sa nikdy nedostala pod hranicu 0.43 pri $r = 1$ a 0.40 pri $r = 10$, čiže metóda skončila na druhé zastavovacie kritérium. ε_2 sme pre r=1 zvolili 10^{-7} a pre r=10 sme zvolili 10^{-6}.

Pokým sme nenašli spoľahlivé optimum v metóde s backtrackingom použitím správnych parametrov, ani výsledné $\lambda_{min}, \lambda_{max}$ neboli ideálne, tým pádom nebolo ideálne ani optimum nájdené metódou so zlatým rezom. Keď sme našli presnejšie optimum pri backtrackingu, zlepšila sa aj stabilita metódy so zlatým rezom s $r = 10$. Vo výsledku mali už všetky vyskúšané hodnoty parametra r veľmi podobné vývoje presnosti, tak sme aj pri porovnaní použili iba $r = 1$ a $r = 10$.

Ako najkľúčovejšie sa však ukázalo zúženie tolerancie metódy zlatého rezu na 10^{-10}, vďaka čomu bola dĺžka kroku dostatočne presná.

Taktiež vývoje funkčných hodnôt účelovej funkcie $J(x)$ boli veľmi podobné. Nájdenie lepšieho optima v časti d) znamenali lepšie lambdy z časti d), čo viedlo k vyššej presnosti, stabilite a rýchlosti konvergencie. Okrem hľadania parametrov optimalizačnými metódami sme skúsili scipy.optimize.minimize a sklearn.linear_model.LogisticRegression. Obe zbehli veľmi rýchlo a našli parametre predikujúce s presnosťou 0.7.

Tabuľka 2: Odhady parametrov logistickej regresie pre rôzne metódy

Metóda	Intercept	x_1	x_2	x_3	x_4	x_5
Const Step	−2.288 96	−0.008 28	−0.015 92	−0.000 39	0.038 23	−0.591 15
Backtracking	−2.286 54	−0.008 30	−0.015 93	−0.000 39	0.038 22	−0.590 53
Golden (r=1)	−2.286 55	−0.008 30	−0.015 92	−0.000 39	0.038 22	−0.590 84
Golden (r=10)	−2.286 56	−0.008 30	−0.015 92	−0.000 39	0.038 22	−0.590 84
sklearn log. reg.	−2.311 80	−0.008 26	−0.015 97	−0.000 41	0.038 35	−0.577 92
scipy minimize	−2.289 12	−0.008 28	−0.015 92	−0.000 39	0.038 23	−0.591 21

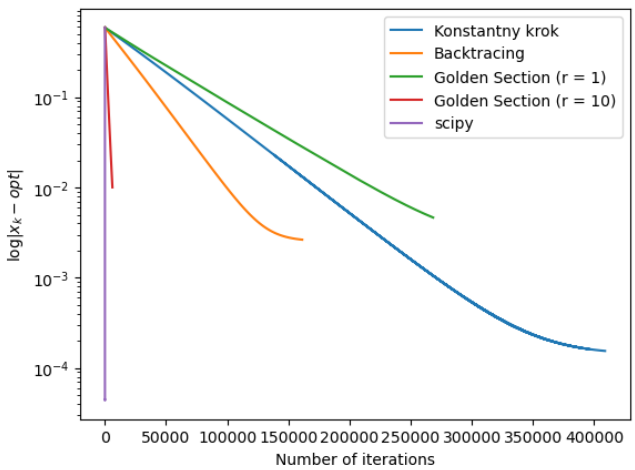
F) Vývoj vzdialenosti od optima

Grafy nižšie zobrazujú vzdialenosť x_k, v závislosti od iterácie k. Líšia sa v použití logaritmickej škály na y-ovej osi. Ako hodnota v opime bola zobraná hodnota, ktorú vypočítalo `scipy`

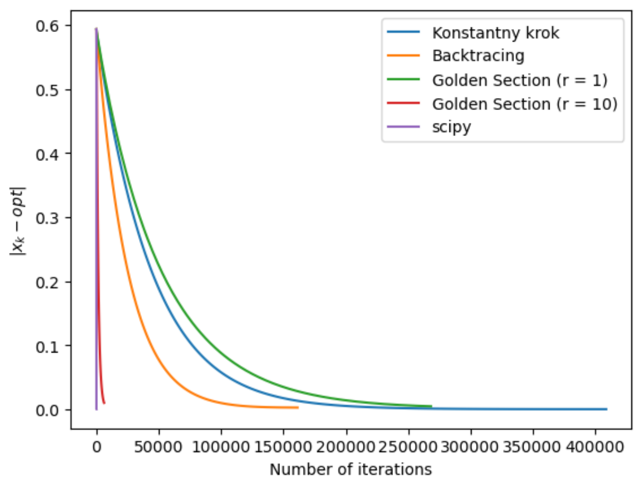
Na prvom grafe je porovnanie naprogramovaných metód pre $\varepsilon_1 = 10^{-3}$ a vysledkov `scipy.optimize.minimize`. Scipy si vybera vhodnú metódu podľa minimalizovanej funkcie a ako je vidno je aj rýchlejšia aj prestnejšia ako zvyšné.

Z grafu sme usúdili:

- gradientna s konštantným krokom síce beží najviac iterácii, ale dostane sa najbližsie k optimu
- gradientnej "so zlatým rezom"klesá rozdiel od optima približne exponenciálne
- gradientna "s backtraking-om"sa v blízkosti optima zlepšuje už len veľmi pomaly



(a) Graf podľa zadania



(b) Graf bez logaritmickej y osi

Obr. 1

Na obrázku 1 vpravo si môžeme všimnúť, že medzi metódou zlatého rezu (r = 10) a scipy je malý rozdiel, čo sa ale o čase výpočtu povedať nedá, ako je vidieť v časti g).

G) Porovnanie optimalizačných metód

V nasledujúcej tabuľke sú zhrnuté základné metriky kvality pre štyri pozorované optimalizačné metódy a dve 'kontrolné' metódy. Tieto metriky zahŕňajú čas behu, počet iterácií, nájdenú optimálnu hodnotu funkcie (3) a normu gradientu v nájdenom optime.

Tabuľka 3: Porovnanie optimalizačných metód

Názov metódy	Čas behu (s)	Počet iterácií	Nájdená optimálna hodnota	Norma gradientu
Konstantný krok	6.3438	408 844	133.961 885	2.3873×10^{-1}
Backtracking	29.3373	161 279	133.961 897	3.3115×10^{-2}
Golden Section (r = 1)	91.6623	268 493	133.962 214	2.4243×10^{-1}
Golden Section (r = 10)	2.5159	6089	133.964 028	4.6907
scipy	0.0028	17	133.961 885	8.8747×10^{-7}
sklearn	0.0089	66	133.961 885	2.7544×10^{-3}

Porovnanie kvality hlavných metód

Pri hodnotení prvých štyroch implementovaných metód ('Konštantný krok', 'Backtracking', 'Golden Section (r = 1)' a 'Golden Section (r = 10)') sa pozeráme na nasledujúce kritériá:

- **Čas behu:** Určuje výpočtovú náročnosť metódy.
- **Počet iterácií:** Indikuje, koľko krokov metóda potrebovala na konvergenciu.
- **Nájdená optimálna hodnota:** Hodnota účelovej funkcie v bode, ktorý daná metóda označila za optimum.
- **Norma gradientu:** Veľkosť gradientu v nájdenom optime. Pre lokálne minimum by mala byť norma gradientu blízka nule. Nižšia hodnota naznačuje lepšiu konvergenciu.

Golden Section (r = 10): Táto metóda bola najrýchlejšia (2.52 s) a vyžadovala najmenej iterácií (6089). Avšak, dosiahla najvyššiu (najhoršiu) optimálnu hodnotu (133.9640) a extrémne vysokú normu gradientu (4.6907).

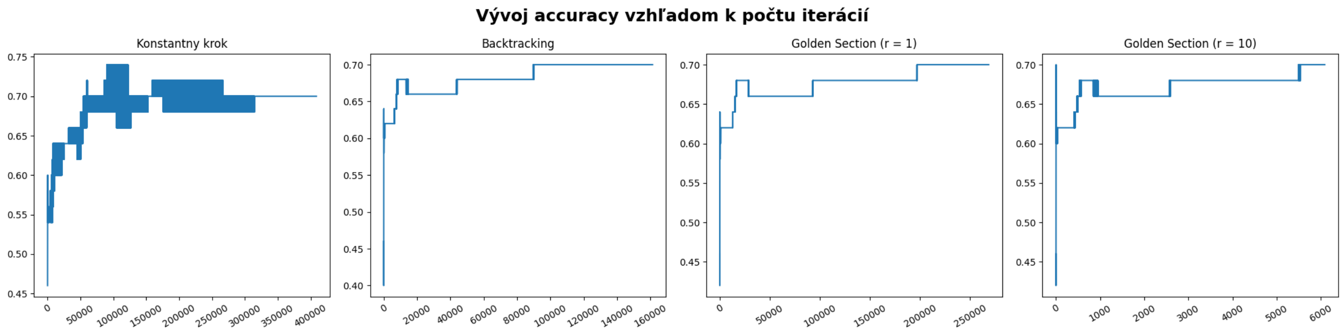
Konstantný krok: Metóda s konštantným krokom bola druhou najrýchlejšou (6.34 s), ale vyžadovala najvyšší počet iterácií (408844). Nájdená optimálna hodnota (133.9619) je najnižšia, ale veľmi blízka výsledku z Backtrackingu. Norma gradientu (0.2387) je však v porovnaní s Backtrackingom vyššia.

Golden Section (r = 1): Táto varianta metódy zlatého rezu bola výrazne najpomalšia (91.66 s) a mala druhý najvyšší počet iterácií (268493). Nájdená optimálna hodnota (133.9622) a norma gradientu (0.2424) sú porovnateľné s metódou konštantného kroku, ale horšie ako pri Backtrackingu. Vzhľadom na dlhý čas behu a nie ideálnu kvalitu riešenia sa javí ako najmenej efektívna z týchto štyroch.

Backtracking: Metóda Backtracking mala stredný čas behu (29.34 s) a druhý najnižší počet iterácií (161279). Táto metóda mala výrazne najnižšiu normu gradientu (0.0331). To znamená, že bod nájdený touto metódou je s najväčšou pravdepodobnosťou najbližšie k skutočnému lokálnemu minimu. Nájdená optimálna hodnota (133.9619) je tiež veľmi dobrá, od výsledku metódy konštantného kroku je iba veľmi málo odchýlená a je veľmi blízko hodnotám dosiahnutým knižnicami `scipy` a `sklearn`.

Grafické znázornenie vývoja presnosti

Nasledujúci obrázok zobrazuje vývoj presnosti (accuracy) v závislosti od počtu iterácií pre hlavné metódy.



Všetky metódy skončili na rovnakej presnosti, a to je 0.7.

Tabuľka 4: Presnosť klasifikácie jednotlivých metód

Metóda	**Presnosť**
Const Step	0.7
Backtracking	0.7
Golden (r=1)	0.7
Golden (r=10)	0.7
sklearn log. reg.	0.7
scipy minimize	0.7

Na výpočet presnosti sme použili logistickú funkciu s rozhodovacou hranicou 0.5. V praxi to znamená, že pacienta klasifikujeme ako chorého ak si je model aspoň na 50% istý, že je pacient chorý.

Záver porovnania hlavných metód

Ako najkvalitnejšiu metódu hodnotíme metódu Backtracking, a to hlavne kvôli tomu, ako blízko sa metóda dostala k skutočnému optimu (indikované nízkou normou gradientu). Aj keď nie je najrýchlejšia, poskytuje najlepší kompromis medzi časom behu a presnosťou konvergencie k minimu. Metóda "Konštantný krok"síce našla o trošku lepšiu optimálnu hodnotu, ale s vyššou normou gradientu. Metóda "Golden Section (r = 1)"mala síce normu podobnú metóde "Konštantný krok", ale nenašla dostatočne presné minimum a čas behu bol výrazne vyšší. "Golden Section (r = 10)"bola síce rýchla, ale jej norma aj minimálna hodnota sú výrazne horšie.

Appendix - Zoznam tolerančných konštánt

Tabuľka 5: Použité hodnoty ε_2 pre rôzne metódy optimalizácie

Metóda	Hodnota ε_2
Konštantný krok	10^{-7}
Backtracking	10^{-8}
Zlatý rez ($r = 1$)	10^{-7}
Zlatý rez ($r = 10$)	10^{-6}