NLLS: Kružnice

Tento úkol navazuje na úkol prokládání bodů kružnicí [/wiki/courses/b0b33opt/cviceni/hw/kruznice_lin/start], kde jsme se snažili proložit body tak, aby minimalizovali součet čtverců jejich algebraických vzdáleností a doufali jsme, že řešení bude zhruba podobné, jako kdybychom minimalizovali součet čtverců vzdáleností bodů od kružnice. V tomto úkolu už budete hledat parametry kružnice, které minimalizují součet čtverců vzdáleností

$$f(x_0, y_0, r) = \sum_{i=1}^{n} \mathrm{dist}(x_i, y_i, x_0, y_0, r)^2, \tag{1}$$

kde (x_i, y_i), $i, \ldots, n$ jsou body kterými prokládáme kružnici, (x_0, y_0) je její střed a r její poloměr.

Budeme minimalizovat f pomocí iteračních metod. Použijeme gradientní metodu a protože je f ve tvaru nelineárních nejmenších čtverců, můžeme použít i Gauss-Newtonovu a Levenberg-Marquardtovu metodu.

Implementační úkoly

Napište funkce vykonávající jednu iteraci Levenberg-Marquardtovy metody, Gauss-Newtonovy metody a gradientní metody na optimalizaci funkce f. U gradientní metody musíme volit délku kroku. K tomu můžeme použít line-search. Tomu se v této úloze věnovat nebudeme a délku kroku dostane vaše funkce jako argument. Vyzkoušejte si sami, jaký vliv má tato délka kroku na konvergenci metody.

Pokud budete prakticky chtít řešit nějakou optimalizační úlohu, typicky na to použijete nějaký solver (z tohoto kurzu třeba na nejmenší čtverce, svd rozklad, lineární programování). Čím specializovanější je to solver, tím větší instance úlohy umí řešit v rozumném čase. Pokud ale máte malou úlohu, často stačí obecný solver, kterému dáte jako argument účelovou funkci a vrátí optimální řešní. Vyzkoušejte i tento přístup.

Jako **úkoly** napište tyto funkce:

- `[x y r success] = LM_iter(X, x0, y0, r0, mu)` , kde `success` indikuje jestli došlo ke zlepšení účelové funkce touto iterací.
- `[x y r] = GN_iter(X, x0, y0, r0)` .
- `[x y r] = grad_iter(X, x0, y0, r0, a)` .

- `f = get_objective_function(X)` , kde `f` je funkce (případně handle na funkci), která přijímá vektor parametrů `c = [x0, y0, r]` a hodnota `f( c )` odpovídá hodnotě $f(x_0, y_0, r)$ pro body X, jak jsme ji definovali v (1). Můžeme tedy například provést volání `f([1,2,3])` a bude platit `f([1,2,3]) == sum(dist(X, 1, 2, 3).^2)` , případně `f([1,2,3]) == sum(dist(X, 1, 2, 3)**2)` .

Budete potřebovat funkci `d = dist(X, x0, y0, r)` , kterou jste už napsali, kde `X` je matice která má v řádcích body, `x0, y0, r0` jsou parametry kružnice před iterací, `x, y, r` jsou parametry po vykonání iterace, `mu` je parameter LM-metody a `a` je délka kroku pro gradientní metody.

Templaty

Templaty, včetně skriptu pro testování si stáhněte zde: template pro matlab [/wiki/_media/courses/b0b33opt/cviceni/hw/kruznice/matlab_template_nlls.zip], template pro python [/wiki/_media/courses/b0b33opt/cviceni/hw/kruznice/python_template_nlls.zip]. Pro python spouštějte skript `nlls.py` , pro matlab `nlls.m` . Animace se pauzuje po každém kroku, musíte klikat pro další iterace.

Implementační poznámka

V řídícím skriptu se přijme krok gradientní metody pouze pokud se zlepší účelová funkce a na základě toho se upraví délka kroku, podobně jako u LM. Nenechte se tím zmást, tímto zaručujeme konvergenci metody. Více to diskutujeme v úkolech pro radost.

Dobrovolné úkoly pro radost

1. Z teorie plyne, že gradientní metoda asymptoticky konverguje do lokálního minima, pokud $\sum_{i=1}^{\infty} \alpha_i = \infty$, $\sum_{i=1}^{\infty} \alpha_i^2 < \infty$, kde α_i je délka kroku v i-té iteraci, což splňuje třeba $\alpha_i = 1/i$. Ověřte, že prakticky se zdaleka neblížíme asymptotickému chování této posloupnosti. Spočítejte $\sum_{i=1}^{k} \alpha_i$, pokud bychom s výpočtem začali v době velkého třesku (před 14 miliardami let), měli bychom tak výkonný počítač, který by dokázal udělat 10^{12} iterací merody za sekundu a dnes bychom dělali k-tou iteraci. Mělo by vám vyjít, $\sum_{i=1}^{k} \alpha_i < 70$.
2. Délka kroku (learning rate u neuronových sítí) je velice choulostivý parametr u gradientní metody a při jeho špatné volbě metoda nekonverguje. V úkolu jsme to v řídícím skriptu vyřešili tak, že přijímáme pouze iterace, které zlepší účelovou funkci a adaptivně nastavujeme délku kroku, tohle ale prakticky není žádoucí. Experimentálně zjistěte jak nastavit délku kroku, aby metoda konvergovala, i když přijmeme každou iteraci. Speciálně si uvědomte, co se stane s gradientem, pokud zduplikujeme body a jak se změní lokální minima. Zkuste délku kroku $\alpha_i = \frac{1}{n+i}$, kde n je počet bodů a nechte metodu (opakovaně) běžet. Uvidíte, že někdy bude zprvu utíkat pryč, ale až bude i dostatečně velké, metoda se 'vzpamatuje' a dokonverguje do lokálního minima. Typicky se to stane když $n \sim i$. To by nás mohlo navést na délku kroku $\alpha_i = \frac{1}{2n+i}$. Zkuste i tu. Kdybychom volili moc malou délku kroku, konvergence bude stabilnější, ale o to pomalejší.

3. Pro $n \geq 3$ a body $\mathbf{x}_1, \ldots, \mathbf{x}_n$ v obecné poloze, zkuste nalézt co nejobecnější podmínku charakterizující, ve kterých bodech je funkce f diferencovatelná.
4. Může se zdát, že iterační metody na nelineární nejmenší čtverce bez omezení (např. Gauss-Newtonovu a Levenberg-Marquardtovu metodu) nejde použít, protože máme omezení $r \geq 0$. Vadí to? Co se stane, budeme-li toto omezení ignorovat? Můžou metody konvergovat k řešení se záporným r?
5. Rozmyslete, že pro nějaké body $\mathbf{x}_1, \ldots, \mathbf{x}_m$ má funkce f více lokálních minim s různou funkční hodnotou. Zkuste najít minimální takovou konfiguraci.
6. Zatím jsme prokládání kružnice formulovali jako minimalizaci funkce (1) přes tři proměnné x_0, y_0, r, tedy

$$\min_{(x_0,y_0)\in\mathbb{R}^2,\, r\geq 0} f(x_0, y_0, r) = \min_{(x_0,y_0)\in\mathbb{R}^2} \overbrace{\min_{r\geq 0} f(x_0, y_0, r)}^{g(x_0,y_0)}.$$

   Ale pokud je (x_0, y_0) pevné, vnitřní minimalizaci přes r lze vyřešit explicitně a napsat tak vzorec pro funkci $g(x_0, y_0)$. Tedy úlohu můžeme vlastně formulovat jako minimalizaci funkce g přes proměnné $(x_0, y_0) \in \mathbb{R}^2$. Rozpracujte tuto myšlenku: odvoďte vzorec pro funkci g, vykreslete si její graf a vrstevnice (což je zde možné, protože je to funkce pouze dvou proměnných), implementujte iterační metodu (Gauss-Newton a/nebo Levenberg-Marquardt) pro minimalizaci funkce g a diskutujte, která formulace (dvě vs. tři proměnné) je výhodnější (např. porovnáním rychlosti/přesnosti iteračních metod příp. pracnosti implementace).

courses/b0b33opt/cviceni/hw/kruznice/start.txt · Last modified: 2021/05/13 11:36 by voracva1