Cvičení z optimalizace
Lineární programování 3.

Vojtěch Franc, 2014

Úvod

V praxi se velmi často objevují optimalizační úlohy s mnoha proměnými nebo velkým množstvím omezení. Takovéto úlohy nelze řešit obecnými numerickými algoritmy na běžných počítačích. V těchto případech nezbývá, než se poohlédnout po specializovaných metodách optimalizace. Na tomto cvičení si probereme "algoritmus odsekávajících nadrovin" (cutting plane algorithm), který je jednou z metod, jenž umožňuje efektivní řešení velkých úloh lineárního programování.

Navážeme na úlohu z předchozího cvičení, kdy jsme prokládali lineární funkci množinou bodů. Budeme řešit úlohu se zcela stejným zadáním až na to, že budeme pracovat s mnohem větší množinou bodů. To znamená, že odpovídající úloha lineárního programování bude mít velké množství omezení a stane se pro běžné numerické algoritmy neřešitelná v rozumném čase.

Minimaxní prokládání lineární funkce velkou množinou bodů

Zadání úlohy je stejné jako v předchozím cvičení, tzn. máme dánu množinu bodů $\mathcal{T} = \{(x_1, y_1), \ldots, (x_m, y_m)\} \in (\mathbb{R} \times \mathbb{R})^m$ a chceme nalézt parametry lineární funkce $f_{a,b}(x) = ax + b$, tak aby maximální absolutní odchylka

$$\varepsilon(a, b) = \max_{i=1,\ldots,m} |f_{a,b}(x_i) - y_i| \tag{1}$$

byla minimální. Jinými slovy cheme vyřešit optimalizační úlohu

$$(a^*, b^*) \in \operatorname*{argmin}_{\substack{a\in\mathbb{R},\\ b\in\mathbb{R}}} \max_{i=1,\ldots,m} |ax_i + b - y_i| \,. \tag{2}$$

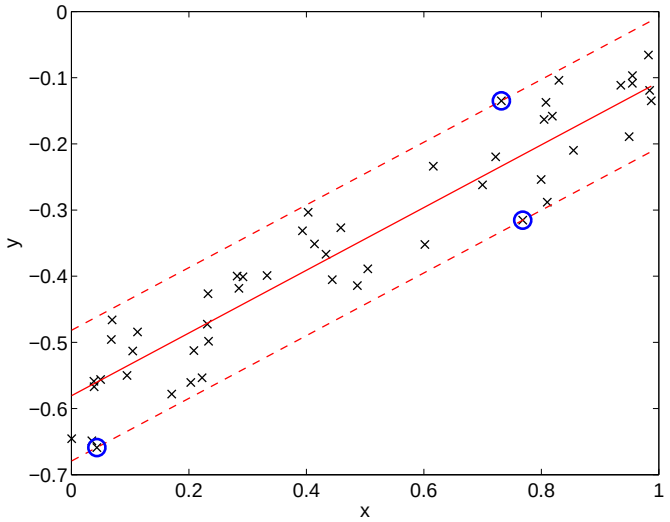
Na minulém cvičení jsme si uvedli grafickou interpretaci této úlohy, a také jsme si odvodili ekvilantní úlohu lineárního programování, která má tvar

$$(a^*, b^*, \lambda^*) \in \operatorname*{argmin}_{a\in\mathbb{R}, b\in\mathbb{R}, \lambda\in\mathbb{R}} \lambda \tag{3a}$$

za podmínky

$$\begin{array}{rcll} ax_i + b - y_i & \leq & \lambda \,, & i = 1, \ldots, m \\ -ax_i - b + y_i & \leq & \lambda \,, & i = 1, \ldots, m \end{array} \tag{3b}$$

Zatímco minule jsme prokládali přímku $m = 50$ body, dnes budeme chtít proložit přímku $m = 1000000$ (slovy miliónem) bodů. To znamená, že odpovídající úloha LP (3) má $2m = 2000000$ omezení. Na běžném počítači s procesorem Intel Centrino @ 1.8GHz a 3GB paměti, trvá výpočet asi 1.5 hodiny. Podotkněme ještě, že i tato úloha je stále velmi zjednodušena, protože uvažujeme pouze body v 1-dimensionální prostoru, tj. obecně můžou být x a a vektory v n-dimenzionálním prostoru. Toto zjednodušení děláme úmyslně proto, abychom mohli problém snadno vizualizovat a tím lépe pochopit.



Obrázek 1: Příklad fitování lineární funkce do množiny bodů. Body odpovídající aktivním omezením jsou označeny modrým kroužkem.

Algoritmus odsekávajících nadrovin

Předpokládejme, že známe optimální řešení problému (3), tj. známe trojici (a^*, b^*, λ^*). Potom si můžeme množinu omezení (3b) rozdělit na prodmnožinu *aktivních* a podmnožinu *neaktivních* omezení podle toho, jestli je po vyřešení úlohy splněna rovnost nebo ostrá nerovnost, tj.:

1. Aktivní omezení:

$$\begin{aligned} a^*x_i + b^* - y_i &= \lambda^*, \quad i \in I_+ \\ -a^*x_i - b^* + y_i &= \lambda^*, \quad i \in I_- \end{aligned} \tag{4}$$

2. Neaktivní omezení:

$$\begin{aligned} a^*x_i + b^* - y_i &< \lambda^*, \quad i \in J_+ \\ -a^*x_i - b^* + y_i &< \lambda^*, \quad i \in J_- \end{aligned}$$

Zavedeme si množinu $I = I_+ \cup I_-$, která obsahuje indexy bodů jenž odpovídají aktivním omezením. Body $\{(x_i, y_i) \mid i \in I\}$ budeme nazývat *aktivní body*. Podobně zavedeme množinu $J = J_+ \cup J_-$ obsahují indexy bodů s neaktivním omezením.

Obrázek 1 ilustruje rozdíl mezi body odpovídající aktivním omezením (označené modrým kroužkem) a body s neaktivním omezením. Je vidět, že body s aktivním omezením leží na kraji pásu, tj. platí $|f_{a^*,b^*}(x_i) - y_i| = \lambda^*$, $i \in I$. Body s neaktivním omezením pak leží uvnitř pásu, tj. $|f_{a^*,b^*}(x_i) - y_i| < \lambda^*$, $i \in J$.

Z obrázku 1 je vidět, že minimální absolutní ochylka ($\frac{1}{2}$ šířky pásu), a tím i řešení celého problému, je určena pravě a jen body odpovídající aktivním omezením. Tudíž pokud odebereme jakýkoliv bod (nebo třeba i všechny) ležící uvnitř pásu, nebude to mít žádný vliv na řešení. Jinými slovy říkáme, že pokud v množině omezení (3b) necháme jen ta, která odpovídají aktivním omezením I, pak řešením takto redukovaného problému dostaneme řešení původního problému. Tj. po vyřešení redukovaného problému

$$(\hat{a}, \hat{b}, \hat{\lambda}) \in \operatorname*{argmin}_{a \in \mathbb{R}, b \in \mathbb{R}, \lambda \in \mathbb{R}} \lambda \tag{5a}$$

za podmínky

$$\begin{array}{rcll} x_i a + b - y_i & \leq & \lambda\,, & i \in I_+ \\ -x_i a - b + y_i & \leq & \lambda\,, & i \in I_- \end{array} \tag{5b}$$

dostaneme $(\hat{a}, \hat{b}, \hat{\lambda})$, která je současně řešením problému (3). Díky nižšímu počtu omezení se dá redukovaný problém (5) řešit jednodušeji v porovnání s původním problémem (3). V konkrétním případě z obrázku 1 je počet bodů $m = 50$ a tudíž původní problém (3) má $2m = 100$ omezení [1]. Naproti tomu redukovaný problém (5), jak je vidět i z obrázku (1), má pouze 3 omezení, tj. více než $16\times$ méně.

K tomu, abychom mohli redukovaný problém (5) řešit, museli bychom znát aktivní omezení (4). Aktivní omezení ovšem nemáme k dispozici, neboť k jejich určení je třeba znát optimální řešení původního problému (3). Naštěstí existuje jednoduchý iterativní algoritmus, který umožňuje množinu aktivních omezení nalézt. Tento algoritmus funguje následovně:

Algorithm 1 Algoritmus odsekvajících nadrovin (Cutting plane algorithm)

1: $t := 1$.
2: Zvol náhodně množinu I_t, např. $I_t = \{1\}$.
3: **repeat**
4: Vyřeš redukovaný problém

$$(a_t, b_t, \lambda_t) \in \operatorname*{argmin}_{a\in\mathbb{R}, b\in\mathbb{R}, \lambda\in\mathbb{R}} \lambda \tag{6a}$$

za podmínky

$$\begin{array}{rcll} a x_i + b - y_i & \leq & \lambda\,, & i \in I_t \\ -a x_i - b + y_i & \leq & \lambda\,, & i \in I_t \end{array} \tag{6b}$$

5: Nalezni bod s největší odchylkou

$$i_t \in \operatorname*{argmax}_{i\in\{1,\ldots,m\}} |a_t x_i + b_t - y_i|$$

a pokud $\varepsilon(a_t, b_t) - \lambda_t >$ Tolerance, pak vlož tento bod mezi aktivní

$$I_{t+1} = I_t \cup \{i_t\}$$

6: $t := t + 1$
7: **until** $\varepsilon(a_t, b_t) - \lambda_t \leq$ Tolerance.

Algoritmus (1) funguje tak, že na počátku zvolí množinu aktivních bodů náhodně (Krok 2). Následně iteruje dva kroky. Nejprve vyřeší redukovaný problém 6, jehož omezení jsou dána aktuálním odhadem množiny aktivních bodů I_t. Dále algoritmus nalezne bod, který nejvíce porušuje omezení. Odchylka tohoto bodu udává hodnotu kriteriální funkce $\varepsilon(a_t, b_t)$, která se porovná s dolní mezí a tím se určí vzdálenost od optimálního řešení. Pokud je tato vzdálenost větší než zadaná tolerance, vloží se nalezený bod do množiny I_t (Krok 5). Tyto dva kroky se opakují dokud není splněna ukončovací podmínka (Krok 7). Splnění ukončovací podmínky zaručí, že kvalita nalezené lineární funkce se neliší od kvality té optimální o více než je požadovaná tolerance, tj. že platí

$$\varepsilon(a_t, b_t) - \varepsilon(a^*, b^*) \leq \text{Tolerance}\,. \tag{7}$$

[1] V ilustračním příkladě nepoužíváme problém s $m = 1000000$ jenž chceme řešit. Důvod je ten, že takovéto množství bodů by se zobrazilo jako jednolitý pás a obrázek by se stal nepřehledným.

Nerovnice (7) vyplyne okamžitě z ukončovací podmínky $\varepsilon(a_t, b_t) - \lambda_t \leq$ Tolerance pokud si uvědomíme, že λ_t je dolní mez optimálního řešení, tj. že platí $\lambda_t \leq \varepsilon(a^*, b^*)$. A skutečně, λ_t je optimální řešení fitovacího problému pro podmnožinu bodů I_t a tudíž nemůže být horší než řešení kdy uvažujeme všechny body.

K tomu abychom dokázali, že se algorimus zastaví v konečném počtu kroků, stačí ukázat, že v kroku 5 se do množiny I_t vloží vždy bod, který tam ještě nebyl. Pokud tento výrok platí, pak se algoritmus zastaví nejdéle po m krocích, kdy I bude obsahovat všechny body a redukovaný problém bude stejný jako problém původní. Důkaz uvedeného výroku není těžký a jeho nalezení je jedním z Vašich úkolů. V mnoha praktických problémech algoritmus naštěstí zkonverguje po několi málo iteracích jak si sami ověříte na naší fitovací úloze.

Název, "algoritmus odsekávajících nadrovin", vyplyne pokud si uvědomíme, že nerovnice, které se v každé iteraci algoritmus přidává k redukovanému problému, odsekávají (separují) aktuální řešení (a_t, b_t) od řešení optimálního (a^*, b^*).

Úkoly k vypracování

1. Implementujte algoritmus 1 a vyzkoušejte ho na datech z minulého cvičení. Pro řešení redukovaného problému pužijte funkci `linprog`, která je součástí optimalizačního toolboxu v Matlabu.

   Výstup: nic.

2. Stáhněte si soubor `http://cmp.felk.cvut.cz/cmp/courses/OPT/cviceni/03/data2.mat`. Soubor obsaduje vektor `x [1000000 x 1]` a vektor `y [1000000 x 1]`, které definují množinu $\mathcal{T}$. V našem případě je tedy $m = 1000000$. Nafitujte lineární funkci pomocí algoritmu implementovaného v bodě 1. Toleranci zvolte Tolerance $= 10^{-4}$.

   Výstup: parametr a a b nalezené přímky.

3. Vykreslete do grafu hodnoty $\varepsilon(a_t, b_t)$ a λ_t jako funkci t.

   Výstup: graf.

4. Jaká je hodnota $\varepsilon(a_t, b_t)$ a λ_t po ukončení algoritmu?

   Výstup: dvě čísla.

5. Dokažte, že v kroku 5 algoritmu 1 se do množiny I_t vždy vkládá bod, který tam ještě nebyl, tj. že vždy platí $i_t \notin I_t$.

   Výstup: stručný důkaz.