Cvičení z RPZ – Support Vector Machines

Vojtěch Franc 2005
(rozšířil Tomáš Werner 2008)

1 Úvod

Cílem tohoto cvičení je implementace učícího algoritmu Support Vector Machines (SVM). SVM je algoritmus realizující princip minimalizace strukturálního risku. Problém minimalizace strukturálního risku je převeden na problém maximalizace vzdálenosti rozdělující nadroviny klasifikátoru k bodům z trénovací množiny - tato vzdálenost se nazývá margin. A konečně problém maximalizace margin je převeden na problém minimalizace kvadratického kritéria. Ve vhodném vyjádření tohoto kritéria se body z trénovací množiny vyskytují pouze ve formě skalárních součinů, což nám umožňuje použít jádrových funkcí k zobecnění rozdělující nadroviny na nelineární klasifikátor (nelineární rozdělující nadplochu).

2 SVM pro separovatelný případ

2.1 Přímá úloha

Lineární SVM hledají nadrovinu $\langle w, x\rangle + b = 0$, která maximalizuje vzdálenost od bodů z lineárně separovatelné trénovací množiny $\{(x_1, y_1), (x_2, y_2), ..., (x_l, y_l)\}$, kde $x_i \in \mathbb{R}^n$ je trénovací vzor a $y_i \in \{+1, -1\}$ je jeho identifikátor třídy (label). Problém hledání nadroviny maximalizující vzdálenost od tříd je ekvivalentní vyřešení optimalizační úlohy

$$(w, b) = \operatorname*{argmin}_{w,b} \frac{1}{2}\|w\|^2 \tag{1}$$

za podmínek

$$\begin{aligned} \langle w, x_i\rangle + b &\geq +1\,, \quad y_i = +1\,, \\ \langle w, x_i\rangle + b &\leq -1\,, \quad y_i = -1\,. \end{aligned} \tag{2}$$

Pokud vyřešíme úlohu (1) dostaneme vektor w a práh b jenž nám určují hledanou nadrovinu.

2.2 Duální úloha

Avšak je výhodné neřešit tuto úlohu přímo, ale převést ji na tzv. duální úlohu. Cílem duální úlohy je nalézt čísla α_i, $i = 1, ..., l$, která jsou řešením úlohy

$$\vec{\alpha} = \underset{\vec{\alpha}}{\operatorname{argmax}} \left(\sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle \right) , \tag{3}$$

za podmínky

$$\begin{aligned} \alpha_i &\geq 0 , \quad i = 1, 2, ..., l , \\ \sum_{i=1}^{l} \alpha_i y_i &= 0 . \end{aligned}$$

Výhodou duální úlohy je to, že zde již nemusíme optimalizovat vektor w, ale pouze reálná čísla α_i za podstatně jednodušších omezení, než tomu bylo v primární úloze.

Poznámka o dualitě v kvadratické optimalizaci Přechod od přímé k duální úloze stojí na teorii *lagrangeovské duality* v optimalizaci, kterou jste možná neprobírali. Viz např. knihu [1] (která je volně na www) nebo skripta Prof. Štechy. Jedná se o to, že ke každé úloze kvadratické (ale i lineární i jiné) optimalizace můžeme sestrojit úlohu *duální*, která je v úzkém vztahu s původní úlohou (nazývanou potom *primární* nebo *přímou*). Jak se toto sestrojení provádí zde nebudeme vysvětlovat. Ale naznačíme tři věty, které specifikují vztah primární a duální úlohy:

- Optima obou úloh se rovnají (tzv. *věta o silné dualitě*).
- Jsou-li obě úlohy v optimu, nemůže se stát, že by sobě odpovídající omezující podmínky primární a duální úlohy byly najednou splněny s ostrou nerovností – tj. alespoň jedna je splněná s přesnou rovností (tzv. *věta o komplementaritě*).
- Pokud je jedna úloha *nesplnitelná* (tj. její omezující podmínky nelze splnit), druhá úloha je *neomezená* (tj. její optimum se může zlepšovat nade všechny meze). A naopak.

2.3 Od duálního řešení zpět k přímému

Po vyřešení duální úlohy je nutné z optimálních duálních proměnných α nalézt optimální proměnné přímé úlohy w a b, které určují rozdělující nadrovinu. Vektor

w určíme pomocí minimalizace Lagrangeovy funkce za znalosti α (podrobnosti vynechváváme):

$$w = \sum_{i=1}^{l} \alpha_i y_i x_i \tag{4}$$

Posunutí b zjistíme pomocí věty o komplementaritě mezi duálními úlohami, podle které může v přímé podmínce 2 nastat rovnost jen tehdy, když $\alpha_i > 0$. Definujme tedy množinu

$$I = \{i : 0 < \alpha_i\}$$

Tato množina vlastně obsahuje indexy trénovacích bodů jež leží ve vzdálenosti $+1$ nebo -1 od rozdelující nadroviny – jsou to vlastně indexy *support vectors*. Pak se posunutí b určí jako

$$b = \frac{1}{|I|} \sum_{i \in I} \left(y_i - \sum_{j=1}^{l} \alpha_j y_j \langle x_j, x_i \rangle \right) \tag{5}$$

2.4 Nelineární SVM

Další velkou výhodou je fakt, že data x_i se v duální úloze vyskytují pouze ve formě skalárních součinu. To nám umožňuje použít jádrových funkcí pro hledání nelineárního klasifikátoru místo rozdělující nadroviny. Celá modifikace spočívá pouze v zaměnění všech skalárních součinů $\langle x_i, x_j \rangle$ za příslušnou jádrovou funkci $k(x_i, x_j)$. Klasifikaci bodu x provedeme podle znaménka funkce

$$f(x) = \sum_{i=1}^{l} \alpha_i y_i k(x_i, x_j) + b \,,$$

což není nic jiného než určení na jaké straně rozdělující nadplochy $\langle w, \phi(x) \rangle + b = 0$ ve vyrovnaném příznakovém prostoru klasifikovaný bod x leží. Funkci $f(x)$ jsme získali dosazením rovnice (4) do $f(x) = \langle w, x \rangle + b$ a nahrazením skalárního součinu jádrovými funkcemi.

3 SVM pro neseparovatelný případ

Při definování primární úlohy (1) jsme předpokládali, že data z trénovací množiny jsou separovatelná, což znamená, že lze nalézt nadroviny jenž rozdělí trénovací body s nulovou chybou. Jinými slovy, body z první třídy $\{x_i \colon y_i = +1\}$ mají mít kladnou vzdálenost od rozdělující nadroviny a body z druhé třídy $\{x_i \colon y_i = -1\}$ mají mít zápornou vzdálenost od nadroviny. Uvedený požadavek je vyjádřen soustavou nerovnic (2) primární úlohy. Pokud jsou data z trénovací množiny neseparabilní pak neexistuje nadrovina, která by je oddělila s nulovou chybou a tudíž ani soustava omezení (2) není splnitelná. Řešením uvedeného problému

jsou SVM pro neseparabilní případ, které jednoduše uvolní soustavu omezení přidáním nových relaxačních proměnných ξ_i. Primární úloha má potom tvar

$$(w, b, \xi_i) = \underset{w,b,\xi_i}{\operatorname{argmin}} \left(\frac{1}{2}\|w\|^2 + C \sum_{i=1}^{l} \xi_i \right) , \tag{6}$$

za omezení

$$\begin{aligned} \langle w, x_i \rangle + b &\geq +1 - \xi_i , \quad y_i = +1 , \\ \langle w, x_i \rangle + b &\leq -1 + \xi_i , \quad y_i = -1 , \end{aligned} \tag{7}$$

Vidíme že v kritérium (6) přibyl nový člen $C \sum_{i=1}^{l} \xi_i$, který vyjadřuje náš požadavek na co nejmenší počet bodů z trénovací množiny porušujících podmínky (7). Konstanta C, kterou je nutno zvolit před optimalizací, určuje zdali upřednostňujeme minimalizaci prvního členu (odpovídá maximalizaci margin) nebo minimalizaci druhého členu, jenž odpovídá počtu špatných bodů porušujících soustavu omezení. Poznamenejme, že pro separovatelný případ můžeme nastavit $C = \infty$, protože lze najít takovou nadroviny, že žádný bod nebude přesahovat. Opět pro tento případ je lepší hledat řešení v jeho duálním vyjádření, které se od duální úlohy pro separovatelný případ liší pouze v přidání nové omezující podmínky $0 \leq \alpha_i \leq C$. A tedy duální úloha SVM pro neseparovatelný případ je

$$\vec{\alpha} = \underset{\vec{\alpha}}{\operatorname{argmax}} \left(\sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle \right) , \tag{8}$$

za podmínky

$$\begin{aligned} C \geq \alpha_i &\geq 0 , \quad i = 1, 2, ..., l , \\ \sum_{i=1}^{l} \alpha_i y_i &= 0 . \end{aligned}$$

Zpět k přímému řešení se přejde podobně jako v neseparabilním případě – pouze množina I musí být dána jako

$$I = \{i : 0 < \alpha_i < C\}$$

Dodatečná podmínka $\alpha_i < C$ zaručí (dle komplementarity), že $\xi_i = 0$.

4 Numerické řešení

Pro řešení kvadratického optimalizačního kritéria duální úlohy použijeme funkci na řešení kvadratického programování. Tato funkce typicky vyžaduje maticový zápis kvadratického kritéria, který snadno získáme. Maticově lze duální úlohu (8) zapsat ve tvaru

$$\vec{\alpha} = \underset{\vec{\alpha}}{\operatorname{argmax}} \left(\vec{\alpha}^T \cdot \vec{1} - \frac{1}{2} \vec{\alpha}^T \cdot H \cdot \vec{\alpha} \right) ,$$

za podmínek

$$\begin{aligned} \vec{\alpha}^T \cdot \vec{y} &= 0\,, \\ \vec{0} \leq \vec{\alpha} &\leq \vec{C}\,, \end{aligned}$$

kde

$\vec{\alpha}$	je vektor $[l \times 1]$ hledaných čísel,
H	je matice $[l \times l]$ a její prvky jsou $H(i,j) = y_i y_j k(x_i, x_j)$
$\vec{C}$	je vektor $[l \times 1]$ obsahující jehož prvky jsou rovny C,
$\vec{y}$	je vektor $[l \times 1]$ obsahující identifikátory tříd y_i,
$\vec{1}$	je vektor $[l \times 1]$ obsahující jedničky,
$\vec{0}$	je vektor $[l \times 1]$ obsahující nuly.

Konkrétně použijeme funkci `gsmo` z rozpoznávacího toolboxu. Nezapomeňte nastavit maximální počet iterací – jinak (hlavně v neseparabilním případě a velkém C) funkce poběží velmi (nekonečně) dlouho. Také nezapomeňte kontrolovat výstupní příznak říkající, zda optimalizace skončila úspěšně či neúspěšně a proč.

Alternativnou je použít funkci `quadprog` z Optimalizačního toolboxu v Matlabu. To však nedoporučujeme, protože tato funkce je pomalejší a jsou s ní i jiné problémy. V případě jejího použití musíte k H přičíst malou jednotkovou matici, např. $10^{-12}\delta(i,j)$, protože jinak je algoritmus nestabilní.

4.1 Kdy skončí optimalizace úspěšně?

Co se stane, když jsou třídy neseparovatelné a my se snažíme řešit SVM pro separovatelný případ (nebo, což je stejné, řešíme SVM pro neseparovatelný případ, ale nastavíme $C = \infty$)? V tom případě bude primární úloha nesplnitelná – tj. buď podmínky (2) nebudou splnitelné nebo bude minimum optimalizační úlohy (6)+(7) rovné $+\infty$. Podle věty o dualitě v kvadratické optimalizaci z toho vyplývá, že duální úloha bude neomezená – tj. optimum úlohy (3) nebo (8) bude rovné $+\infty$. To je vlastně jasné – protože podle duality se musejí optima navzájem duálních úloh rovnat (zde se obě rovnají $+\infty$).

5 Zadání úlohy

Postupujte podle následujících bodů:

1. Seznamte se se SVM v demonstračním programu `demo_svm` ze Statistical Pattern Recognition Toolboxu.

2. Implementujte SVM popsaný v sekci (2). Pro řešení problému kvadratického programování použijte funkci `quadprog` z optimalizačního toolboxu v Matlabu.

3. Ověřte funkčnost vašeho algoritmu na jednoduchých synteticky vygenerovaných datech. Pro generování dat použijte funkci `createdata`. Trénovací

data zobrazte funkcí `ppatterns` a nalezenou rozdělující nadplochu pomocí funkce `psvm`.

Reference

[1] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.