Support Vector Machines

Počítačové zpracování mluveného jazyka

Josef Michálek

Západočeská univerzita

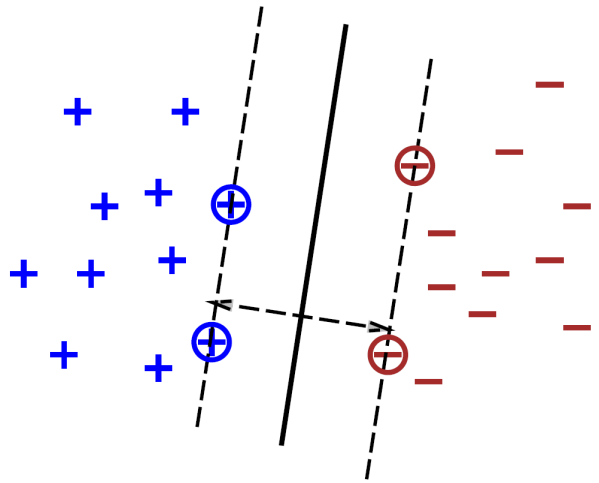
Osnova

Co je SVM

- SVM je metoda strojového učení s učitelem
- Používá se ke klasifikaci i k regresi
- Trénovací set obsahuje prvky náležející do první nebo druhé třídy. Trénovací algoritmus SVM pak vytvoří model, který predikuje, do které třídy patří nové prvky.

Lineární SVM - separabilní případ

Předpokládejme, že trénovací množina je lineárně separabilní. Trénovací data označíme $\{x_i, y_i\}$, $i = 1, \ldots, l$, $y_i \in \{-1, 1\}$, $x_i \in \mathbb{R}^d$, kde x_i je prvek z trénovací množiny a y_i je třída, do které prvek patří. Předpokládejme, že existuje nadrovina oddělující pozitivní prvky od negativních ($y_i = 1$ od $y_i = -1$). Body ležící na nadrovině splňují:



$$w \cdot x + b = 0,$$

kde w je normála nadroviny, $||w||$ je její euklidovská norma a $|b|/||w||$ je kolmá vzdálenost od nadroviny k počátku souřadnic.
Nejkratší vzdálenost od prvků na každé straně roviny označíme d_+ a d_-.
Součet $d_+ + d_-$ se nazývá šířka hraničního pásma (*margin*).
Cíl trénovacího algoritmu SVM je nalézt takovou rozdělující nadrovinu, která má největší šířku hraničního pásma.

Formulace problému

Předpokládejme, že trénovací množina splňuje

$$x_i \cdot w + b \geq +1 \quad \text{pro } y_i = +1 \tag{1}$$

$$x_i \cdot w + b \leq -1 \quad \text{pro } y_i = -1 \tag{2}$$

Rovnice lze sloučit do

$$y_i(x_i \cdot w + b) - 1 \geq 0 \quad \forall i \tag{3}$$

Prvky splňující rovnost z rovnice 1 leží na nadrovině $H_1 : x_i \cdot w + b = +1$ Prvky splňující rovnost z rovnice 2 leží na nadrovině $H_2 : x_i \cdot w + b = -1$
Nadroviny H_1 a H_2 jsou rovnoběžné a d_+ i d_- jsou rovné $1/||w||$. Šířka hraničního pásma je tedy $2/||w||$. Rozdělující nadrovinu s nejširším pásmem najdeme minimalizací $||w||$ vzhledem k podmínkám 3. V praxi se minimalizuje $||w||^2/2$, protože výsledek je stejný a vede to ke zjednoduššení vztahů.
Trénovací prvky pro které platí rovnost v rovnici 3 se nazývají podpůrnými vektory (*support vectors*). Pokud se odstraní, řešení problému se změní, zatímco ostatní prvky se mohou odstranit bez změny řešení.

Lagrangeovská formulace

Převést formulaci problému do Lagrangeovské je výhodné zejména ze dvou důvodů:

1. Omezení z rovnice 3 se změní na omezení Lagrangeovských multiplikátorů, s kterými se lépe pracuje
2. Prvky z trénovací množiny se v Lagrangeovské formulaci problému objevují pouze ve formě skalárního součinu. To nám umožní provést tzv. kernel trick, který bude později důležitý pro nelineární SVM

Pro každé omezení z 3 zavedeme Lagrangeovský multiplikátor $\alpha_i, \quad i = 1, \ldots, l$. Lagrangián pro omezení ve tvaru $c_i \geq 0$ se vytvoří vynásobením omezení 3 Lagrangeovskými multiplikátory a jeho odečtením od ztrátové funkce.

Lagrangeovská formulace problému

$$\text{Minimalizace } L_P = \frac{1}{2}||\boldsymbol{w}||^2 - \sum_{i=1}^{l} \alpha_i y_i (\boldsymbol{x}_i \cdot \boldsymbol{w} + b) + \sum_{i=1}^{l} \alpha_i \tag{4}$$

vzhledem k w a b, derivace L_P vzhledem ke všem α_i musejí být nulové a $\alpha_i \geq 0 \quad \forall i$

Lagrangeovská formulace

L_P je konvexní a body splňující omezení tvoří konvexní množinu, jedná se tedy o problém kvadratického programování a lze řešit duální problém hledání maxima L_P za podmínek, že gradient L_P vzhledem k w a b je nulový a $\alpha_i \geq 0 \quad \forall i$. Tato duální formulace se nazývá Wolfeho dualita.
Aby byl gradient L_P vzhledem k w a b nulový, musí platit

$$w = \sum_i \alpha_i y_i x_i \tag{5}$$

$$\sum_i \alpha_i y_i = 0 \tag{6}$$

Po dosazení těchto omezení do 4 získáme:

Duální Lagrangeovská formulace problému

$$\text{Maximalizace } L_D = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \boldsymbol{x}_i \cdot \boldsymbol{x}_j \tag{7}$$

vzhledem k α_i při omezení $\alpha_i \geq 0 \; \forall i$ a $\sum_i \alpha_i y_i = 0$

Výsledné řešení je pak dáno vztahem 5. Body s $\alpha_i > 0$ jsou podpůrné vektory a leží v nadrovinách H_1 nebo H_2.

Karush-Kuhn-Tucker podmínky

Karush-Kuhn-Tucker (KKT) podmínky jsou nutné podmínky pro optimální řešení v úloze nelineárního programování za předpokladu splnění určitých podmínek regularity, které jsou pro SVM splněny vždy. Pro SVM jsou zároveň i postačující, protože SVM je konvexní problém. Hledání řešení SVM problému je ekvivalentní hledání řešení KKT podmínek.

KKT podmínky pro primární problém SVM

$$\frac{\partial L_P}{\partial w_v} = w_v - \sum_i \alpha_i y_i x_{iv} = 0 \quad v = 1, \ldots, d \tag{8}$$

$$\frac{\partial L_P}{\partial b} = -\sum_i \alpha_i y_i = 0 \tag{9}$$

$$y_i(\boldsymbol{x}_i \cdot \boldsymbol{w} + b) - 1 \geq 0 \quad i = 1, \ldots, l \tag{10}$$

$$\alpha_i \geq 0 \quad \forall i \tag{11}$$

$$\alpha_i(y_i(\boldsymbol{x}_i \cdot \boldsymbol{w} + b) - 1) = 0 \quad \forall i \tag{12}$$

kde d je počet rozměrů a l je počet prvků v trénovací množině.

Karush-Kuhn-Tucker podmínky

Zatímco w může být spočteno během trénování vztahem 5, b takle spočítat nelze. Pro jeho spočtení lze použít vztah 12 při zvolení jakéhokoliv i pro které $\alpha_i \neq 0$. Pro vyšší numerickou přesnost ho lze také spočítat jako aritmetický průměr pro všechna vhodná i.
Klasifikace se provede určením, na které straně rozdělující nadroviny prvek leží pomocí vztahu:

Klasifikace SVM

$$y^* = \operatorname{sgn}(f(\boldsymbol{x})) = \operatorname{sgn}(\boldsymbol{x} \cdot \boldsymbol{w} + b) \tag{13}$$

Lineární SVM - neseparabilní případ

Tento algoritmus nenajde vhodné řešení pro lineárně neseparabilní trénovací množinu. Tento problém se řeší úpravou omezení 1 a 2 a ztrátové funkce, která se optimalizuje. Zavede se cena za porušení omezení označená ξ_i, $i = 1, \ldots, l$. Nová omezení pak mají tvar

$$\boldsymbol{x}_i \cdot \boldsymbol{w} + b \geq +1 - \xi_i \quad \text{for } y_i = +1 \tag{14}$$

$$\boldsymbol{x}_i \cdot \boldsymbol{w} + b \leq -1 + \xi_i \quad \text{for } y_i = -1 \tag{15}$$

$$\xi_i \geq 0 \quad \forall i \tag{16}$$

Sloučené omezení je pak

$$y_i(\boldsymbol{x}_i \cdot \boldsymbol{w} + b) - 1 + \xi_i \geq 0, \quad \xi_i \geq 0 \quad \forall i \tag{17}$$

Aby došlo k chybě pro trénovací prvek $\{x_i, y_i\}$, odpovídající ξ_i musí překročit hodnotu 1. $\sum_i \xi_i$ je pak horní hranice počtu trénovacích chyb.

Lagrangeovská formulace

Ztrátová funkce se změní z $||\boldsymbol{w}||^2/2$ na $||\boldsymbol{w}||^2/2 + C(\sum_i \xi_i)^k$, kde C je parametr zvolený uživatelem a představuje kompromis mezi minimalizací trénovacích chyb a maximalizací šířky hraničního pásma.
Jedná se o problém konvexního programování pro jakékoliv kladné přirozené k.
Pro hodnoty $k = 2$ nebo $k = 1$ se také jedná o problém kvadratického programování a při volbě $k = 1$ se v duálním problému neobjeví ani ξ_i ani odpovídající Lagrangeovy multiplikátory, proto použijeme $k = 1$.
Pro vyjádření nového Lagrangiánu pro primární problém zavedeme kladné Lagrangeovy multiplikátory μ_i pro každé ξ_i z 16.

Lagrangián primární formulace problému

$$L_P = \frac{1}{2}||\boldsymbol{w}||^2 + C\sum_i \xi_i - \sum_{i=1}^{l} \alpha_i(y_i(\boldsymbol{x}_i \cdot \boldsymbol{w} + b) - 1 + \xi_i) - \sum_{i=1}^{l} \mu_i \xi_i \qquad (18)$$

Lagrangeovská formulace

Duální Lagrangeovská formulace problému

$$\text{Maximalizace } L_D = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \boldsymbol{x}_i \cdot \boldsymbol{x}_j \tag{19}$$

vzhledem k α_i při omezení $0 \leq \alpha_i \leq C \ \forall i$ a $\sum_i \alpha_i y_i = 0$

Výsledné řešení je opět dáno vztahem 5. Jediný rozdíl oproti separabilnímu případu je, že hodnoty α_i teď mají horní mez C.

Karush-Kuhn-Tucker podmínky

KKT podmínky pro primární problém SVM

$$\frac{\partial L_P}{\partial w_v} = w_v - \sum_i \alpha_i y_i x_{iv} = 0 \quad v = 1, \ldots, d \tag{20}$$

$$\frac{\partial L_P}{\partial b} = -\sum_i \alpha_i y_i = 0 \tag{21}$$

$$\frac{\partial L_P}{\partial \xi_i} = C - \alpha_i - \mu_i = 0 \quad i = 1, \ldots, l \tag{22}$$

$$y_i(\boldsymbol{x}_i \cdot \boldsymbol{w} + b) - 1 + \xi_i \geq 0 \quad i = 1, \ldots, l \tag{23}$$

$$\xi_i \geq 0 \quad \forall i \tag{24}$$

$$\alpha_i \geq 0 \quad \forall i \tag{25}$$

$$\mu_i \geq 0 \quad \forall i \tag{26}$$

$$\alpha_i(y_i(\boldsymbol{x}_i \cdot \boldsymbol{w} + b) - 1 + \xi_i) = 0 \quad \forall i \tag{27}$$

$$\mu_i \xi_i = 0 \quad \forall i \tag{28}$$

Karush-Kuhn-Tucker podmínky

Stejně jako v separabilním případě můžeme použít KKT podmínek k nalezení b, v tomto případě podmínky 27 a 28.
Z podmínky 22 vyplývá, že pro $\alpha_i < C$ musí být odpovídající μ_i kladné a z podmínky 28 víme, že $\xi_i = 0$.
To znamená, že můžeme vzít jakýkoliv prvek z trénovací množiny s $0 < \alpha_i < C$ a použít pouze podmínku 27 k určení hodnoty b.
Podle hodnoty α_i řadíme každý trénovací vektor do jedné ze 3 skupin:

- **Lower bound** $\alpha_i = 0, \quad f(\boldsymbol{x}) > 1$
- **Free (volné)** $0 < \alpha_i < C, \quad f(\boldsymbol{x}) = 1$
- **Upper bound** $\alpha_i = C, \quad f(\boldsymbol{x}) < 1$

$$\text{kde } f(x) = w \cdot x + b$$

Nelineární SVM

Diskriminační funkce $f(x)$ byla zatím lineární funkce, ale lze ji rozšířit i pro nelineární případ.
V problému trénování SVM se trénovací data objevují pouze ve vztahu L_D (7) ve formě skalárního součinu $x_i \cdot x_j$. Trénovací data můžeme transformovat z původního prostoru L do jiného Euklidovského prostoru H použitím transformace

$$\Phi : \boldsymbol{R}^d \rightarrow H \tag{29}$$

Trénovací algoritmus teď závisí pouze na skalárním součinu trénovacích dat v prostoru H, tzn. $\Phi(\boldsymbol{x}_i) \cdot \Phi(\boldsymbol{x}_j)$. Definujeme jádrovou funkci (*kernel function*) K jako

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \Phi(\boldsymbol{x}_i) \cdot \Phi(\boldsymbol{x}_j) \tag{30}$$

V trénovacím algoritmu teď můžeme použít pouze hodnoty jádrové funkce K, aniž bychom znali vztah pro výpočet Φ. Tomu se říká jádrový trik (*kernel trick*).

Nelineární SVM

Prostor H má obvykle o mnoho více dimenzí než L, někdy dokonce nekonečno. To znamená, že pro neseparabilní data můžeme najít transformaci Φ takovou, že data v novém prostoru separabilní jsou. SVM pak můžeme natrénovat v přibližně stejném čase, jako při použití netransformovaných dat. Jedná se stále o lineární klasifikátor, pouze v jiném prostoru.

Duální Lagrangeovská formulace problému

$$\text{Maximalizace } L_D = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j K(\boldsymbol{x}_i, \boldsymbol{x}_j) \tag{31}$$

vzhledem k α_i při omezení

$$0 \leq \alpha_i \leq C \ \forall i \tag{32}$$

$$\sum_i \alpha_i y_i = 0 \tag{33}$$

Protože SVM je problém konvexního programování, každé lokální řešení je zároveň řešením globálním.

Klasifikace nelineárního SVM

Pro klasifikaci prvku x můžeme použít vztah 13, to ale vyžaduje transformaci x do prostoru H. Tomu se lze vyvarovat dosazením vztahu pro výpočet w (5) do 13. Ve vztahu pro klasifikaci pak můžeme použít jádrovou funkci 30 místo explicitního spočtení $\Phi(x)$.

Klasifikace

$$\begin{aligned} y^* &= \operatorname{sgn}(\boldsymbol{\Phi(x)} \cdot \boldsymbol{w} + b) \\ &= \operatorname{sgn}(\sum_i \alpha_i y_i \Phi(s_i) \cdot \Phi(x_i) + b) \\ &= \operatorname{sgn}(\sum_i \alpha_i y_i K(s_i, x_i) + b) \end{aligned} \tag{34}$$

kde s_i jsou podpůrné vektory.

Jádrové funkce

Příklady jádrových funkcí

Lineární	$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{x}_i \cdot \boldsymbol{x}_j$
Polynomiální	$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = (a\boldsymbol{x}_i \cdot \boldsymbol{x}_j + b)^d$
RBF (Radial basis function)	$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp(-\gamma \lvert\lvert \boldsymbol{x}_i - \boldsymbol{x}_j \rvert\rvert^2)$
Sigmoid	$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \tanh(a\boldsymbol{x}_i \cdot \boldsymbol{x}_j + r)$

Kritérium optimality

Nechť α^* je řešení duálního problému 45 a g^* jsou derivace L_D v α^*.

$$g_i^* = \frac{\partial L_D(\alpha^*)}{\partial \alpha_i} = 1 - y_i \sum_j y_j \alpha_j^* K_{ij} \tag{35}$$

kde $K_{ij} = K(\boldsymbol{x}_i, \boldsymbol{x}_j)$.
Omezení 32 upravíme na omezení $y_i \alpha_i$:

$$y_i \alpha_i \in [A_i, B_i] = \begin{cases} [0, C] & \text{if } y = +1 \\ [-C, 0] & \text{if } y = -1 \end{cases} \tag{36}$$

Dále uvažujme pár indexů (i, j) takový, že platí $y_i \alpha_i^* < B_i$ a $A_j < y_j \alpha_j^*$.
α^ϵ je pak definováno jako

$$\alpha_k^\epsilon = \alpha_k + \begin{cases} +\epsilon y_k & \text{když } k = i \\ -\epsilon y_k & \text{když } k = j \\ 0 & \text{jinak} \end{cases} \tag{37}$$

α^ϵ splňuje omezení, pokud je ϵ kladné a dostatečně malé. A protože α^* je řešení duálního problému, platí $L_D(\alpha^\epsilon) \leq L_D(\alpha^*)$.

Kritérium optimality

Rozvoj prvního řádu je

$$L_D(\alpha^\epsilon) - L_D(\alpha^*) = \epsilon(y_i g_i^* - y_j g_j^*) + o(\epsilon) \tag{38}$$

Rozdíl $y_i g_i^* - y_j g_j^*$ musí být záporný. To platí pro všechny (i, j) splňující $y_i \alpha_i^* < B_i$ a $A_j < y_j \alpha_j^*$, takže můžeme stanovit nutnou podmínku optimality:

$$\exists \rho \in \boldsymbol{R} \text{ takové, že } \max_{i \in I_{\mathbf{up}}} y_i g_i^* \leq \rho \leq \min_{j \in I_{\mathbf{down}}} y_j g_j^* \tag{39}$$

kde $I_{\mathbf{up}} = \{i | y_i \alpha_i < B_i\}$ a $I_{\mathbf{down}} = \{j | A_j < y_j \alpha_j\}$.
Obvykle existují koeficienty α_k^*, které leží mezi svými spodními i horními mezemi. A protože taková k jsou v $I_{\mathbf{up}}$ i $I_{\mathbf{down}}$, odpovídající $y_k g_k^*$ je pak na obou stranách nerovnice a ρ pak může nabývat jediné možné hodnoty. Kritérium optimality (39) je nutná i postačující podmínka řešení.

Metody řešení

Trénování SVM je konvexní optimizační problém, proto je možné pro řešení využít optimizační programy jako MINOS nebo LOQO. Úlohy, pro které jsou tyto programy navrženy, se však od SVM liší především v těchto bodech:

- Optimizační programy jsou často navrženy tak, aby využily řídkost dat v kvadratické části ztrátové funkce, ale ta v případě SVM nebývá řídká. Oproti tomu SVM mívají řídké řešení.
- Matice s hodnotami kernelové funkce K_{ij} se často nevejde do paměti počítače. Prvky matice je třeba vyčíslit vždy, když jsou potřeba, nebo vhodně využít vyrovnávací pamět.

Direction search

Optimizace duálního problému je dosažena prohledáváním ve vhodně zvolených směrech.
Předpokládejme, že máme počáteční bod α, který splňuje omezení

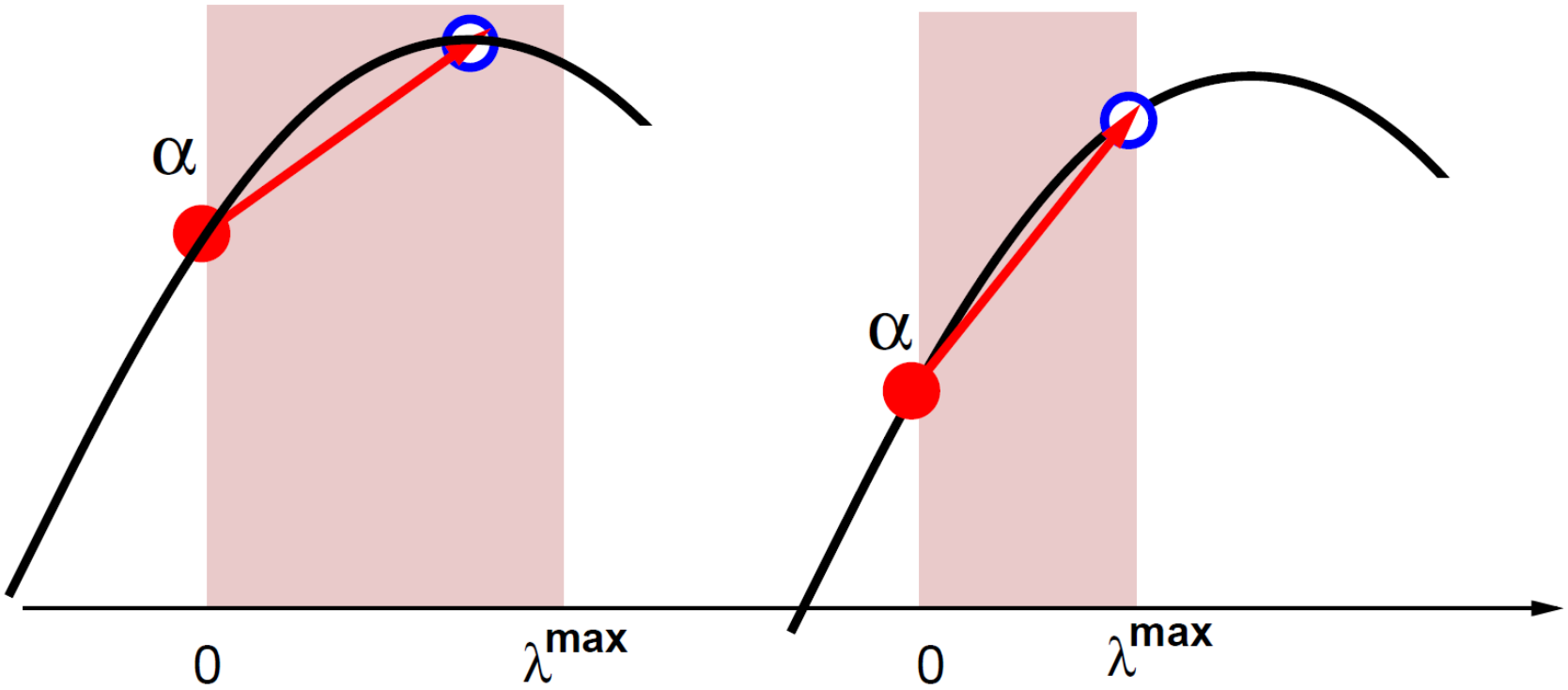
$$0 \leq \alpha_i \leq C \ \forall i \tag{40}$$

$$\sum_i \alpha_i y_i = 0 \tag{41}$$

Směr u je pak vhodný směr, pokud můžeme v tomto směru bod α posunout do bodu $\alpha + \lambda u$, aniž by došlo k porušení omezení. Množina všech $\lambda \geq 0$ splňujících omezení se značí Λ a protože se jedná o konvexní problém a hodnoty α jsou omezeny, je Λ uzavřený interval $< 0, \lambda_{\max} >$.
Optimizace duálního problému L_D je pak nalezení

$$\lambda^* = \arg\max_{\lambda \in \Lambda} L_D(\alpha + \lambda u) \tag{42}$$

Direction search



Direction search

Protože L_D je kvadratická funkce, maximální λ^+ se určí vztahem

$$\lambda^+ = \frac{\frac{\partial L_D(\alpha+\lambda u)}{\partial \lambda}|_{\lambda=0}}{\frac{\partial^2 L_D(\alpha+\lambda u)}{\partial \lambda^2}|_{\lambda=0}} = \frac{g^T u}{u^T H u} \tag{43}$$

kde g je gradient a H je Hessova matice duální funkce L_D.

$$g_i = 1 - y_i \sum_j y_j \alpha_j K_{ij}, \quad H_{ij} = y_i y_j K_{ij}$$

Řešení problému λ^* je pak λ^+ omezené na intervalu $\Lambda =< 0, \lambda_{\max} >$

$$\lambda^* = \max(0, \min(\lambda_{\max}, \frac{g^T u}{u^T H u})) \tag{44}$$

V každé iteraci se zvolí vhodný směr u a použije se vztah (44) dokud není nalezeno řešení.

Direction search

Vhodný směr u má pro SVM několik omezení:

- u je omezeno na lineární podprostor $\sum_i \alpha_i y_i = 0$.
- Omezení $0 \leq \alpha_i \leq C$ určuje znaménka některých koeficientů vektoru u. u_i je nezáporné pokud $\alpha_i = 0$ a nekladné pokud $\alpha_i = C$.
- Směr prohledávání musí být směru stoupání, tzn. směr u musí ležet v poloprostoru $g^T u > 0$

Modified gradient projection je algoritmus řešení SVM, který při určení směru prohledávání ignoruje omezení $0 \leq \alpha_i \leq C$. Poté z pracovní množiny vyřadí ty prvky, pro které by nalezený směr mohl porušit toto omezení.

Dekompoziční metody

Obecné solvery jako MINOS nebo LOQO předpokládají, že je k dispozici celá kernelová matice, to ale v případě SVM problému nemusí být možné ani vhodné.

1. Výpočet kernelové matice je drahý – experimenty potvrzují, že u moderních metod trénování SVM tvoří většinu času výpočet hodnot kernelové matice
2. Výpočet všech hodnot kernelové matice je zbytečný – vztah pro výpočet gradientu závisí pouze na hodnotách K_{ij} odpovídajících alespoň jednomu podpůrnému vektoru, ostatní se násobí nulou. Celkový čas trénování efektivního solveru je menší než čas potřebný pro spočtení celé kernelové matice.
3. Celá kernelová matice se nevejde do paměti – např. kernelová matice pro trénovací množinu o velikosti milion prvků zabere 4 TB paměti.

Dekompoziční metody byly navrženy proto, aby obešly tyto problémy s kernelovou maticí.

Iterative chunking

Jedná se o metodu řešení, která umožňuje obejít nedostatek paměti pro uložení celého problému. Metoda využívá řídkost řešení SVM.
Na začátku trénování se zvolí náhodná podmnožina trénovacích prvků a vloží se do pracovní množiny. Poté se nalezne řešení tohoto podproblému a provede se klasifikace trénovací množiny. Špatně klasifikované prvky jsou kandidáti na podpůrné vektory. Některé z nich přidáme do pracovní množiny a nalezneme řešení pro nový podproblém a pokračujeme, dokud není trénovací množina správně klasifikována. Tento postup funguje, protože se jedná o strojové učení a předpokládá se, že výsledný model dobře zobecňuje. Výsledkem je redukce velkého problému na posloupnost menších problémů.

Obecná dekompozice

Místo všech hodnot α se každou iteraci optimalizuje podmnožina $\alpha_i, i \in B$ a ostatní hodnoty $\alpha_j, j \notin B$ jsou nezměněny.
Z koeficientů α' se spočtou nové koeficienty α' použitím dalšího omezení duálního problému

Duální Lagrangeovská formulace problému

$$\text{Maximalizace } L_D = \sum_i \alpha'_i - \frac{1}{2} \sum_{i,j} \alpha'_i \alpha'_j y_i y_j K(\boldsymbol{x}_i, \boldsymbol{x}_j) \tag{45}$$

vzhledem k α'_i při omezení

$$\forall i \notin B \quad \alpha'_i = \alpha_i$$

$$\forall i \in B \quad 0 \leq \alpha'_i \leq C \ \forall i$$

$$\sum_i \alpha'_i y_i = 0$$

Každou iteraci se zvolí pracovní množina B a odpovídající podproblém použitím vhodného algoritmu optimalizace. Poté se použije rozdíl $\alpha' - \alpha$ pro aktualizaci gradientu. Tyto operace je možné provést s využitím pouze těch řádků kernelové matice, které odpovídají prvkům pracoví množiny.

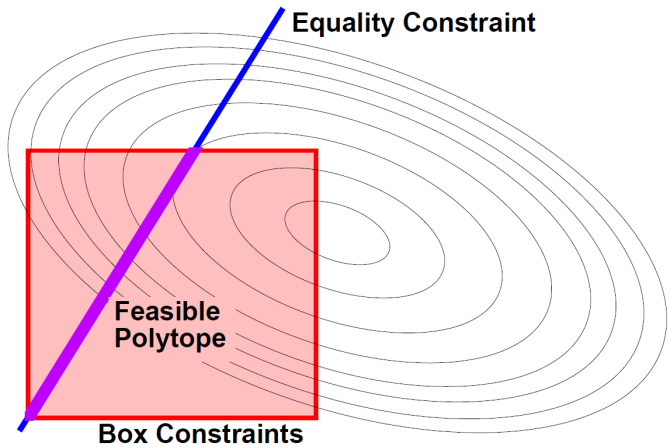
Sequential Minimal Optimization (SMO)

SMO je dekompoziční metoda, která používá nejmenší možný podproblém a řeší ho analyticky. Většina paralelních implementací trénování SVM využívá Plattovo algoritmus SMO vylepšený Keerthim a Fanem.
Z omezení $\sum_i \alpha_i y_i = 0$ plyne, že nejmenší možný podproblém zahrnuje dva Lagrangeovy multiplikátory. Omezení se pak pro jakékoliv dva multiplikátory α_i, α_j změní na

$$0 \leq \alpha_i, \alpha_j \leq C \tag{46}$$

$$y_i \alpha_i + y_j \alpha_j = s \tag{47}$$

kde s je záporný součet všech ostatních $y_k \alpha_k$, který je v každé iteraci neměnný. Tento podproblém lze vyřešit analyticky. Je třeba najít minimum jednorozměrné kvadratické funkce.



Výběr pracovní množiny

Iterace začíná s vektorem koeficientů α. Platí, že pouze dva koeficienty řešení podproblému α' se liší od α. Proto hledaný směr u má pouze dva nenulové koeficienty. Z omezení $\sum_i \alpha_i y_i = 0$ dále plyne, že hledaný směr $u^{ij} = (u_1^{ij}, \ldots, u_n^{ij})$ je

$$u_k^{ij} = \begin{cases} y_i & \text{když } k = i, \\ -y_j & \text{když } k = j, \\ 0 & \text{jinak} \end{cases} \tag{48}$$

Pro získání vhodného směru můžeme dále požadovat $i \in I_{\text{up}}$ a $j \in I_{\text{down}}$.
Nejefektivnější směr je takový, který maximalizuje růst duální funkce

$$u^* = \arg\max_{u^{ij} \in U} \max_{0 \leq \lambda} L_D(\alpha + \lambda u^{ij}) - L_D(\alpha) \tag{49}$$

při omezení $y_i \alpha_i + \lambda \leq B_i$ a $y_j \alpha_j - \lambda \geq A_j$.
Tento vztah přistupuje ke všem hodnotám kernelové matice každou iteraci. I když je výsledný počet iterací malý, každá iterace je velmi pomalá. Pro praktické použití je třeba vztah zjednodušit.

Výběr pracovní množiny

Provedeme aproximaci prvního řádu

$$L_D(\alpha + \lambda u^{ij}) - L_D(\alpha) \approx \lambda g^T u^{ij}$$

Požadovaný směr se najde vztahem

$$u^* = \arg\max_{u^{ij} \in U} \max_{0 \leq \lambda \leq \epsilon} \lambda g^T u^{ij} \tag{50}$$

Pokud není nalezeno řešení, existuje směr u takový, že $g^T u > 0$. Vztah upravíme na

$$u^* = \arg\max_{u^{ij} \in U} g^T u^{ij} \tag{51}$$

S využitím požadovaného tvaru hledaného směru (48) můžeme upravit

$$\max_{u^{ij} \in U} g^T u^{ij} = \max_{i \in I_{\mathbf{up}}, j \in I_{\mathbf{down}}} (y_i g_i - y_j g_j) = \max_{i \in I_{\mathbf{up}}} y_i g_i - \min_{j \in I_{\mathbf{down}}} y_j g_j$$

Tento vztah je kritérium optimality (39), nalezení směru proto najde tzv. maximal violating pair

$$i = \arg\max_{k \in I_{\mathbf{up}}} y_k g_k \tag{52}$$

$$j = \arg\min_{k \in I_{\mathbf{down}}} y_k g_k \tag{53}$$

Výběr pracovní množiny

Výběr maximal violating pair nevyžaduje žádné hodnoty kernelové matice. Aktualizace gradientu na konci iterace některé hodnoty kernelové matice vyžaduje, proto je zde můžeme využít pro vhodnější výběr pracovní množiny. Místo aproximace prvního řádu ponecháme kvadratický člen

$$L_D(\alpha + \lambda u^{ij}) - L_D(\alpha) \approx \lambda(y_i g_i - y_j g_j) - \frac{\lambda^2}{2(K_{ii} + K_{jj} - 2K_{ij})}$$

Optimální λ je pak (vztah má maximum v)

$$\frac{y_i g_i - y_j g_j}{K_{ii} + K_{jj} - 2K_{ij}}$$

Hledaný směr se pak určí vztahem

$$u^* = \arg\max_{u^{ij}} \frac{(y_i g_i - y_j g_j)^2}{2(K_{ii} + K_{jj} - 2K_{ij})}, \quad y_i g_i > y_j g_j \tag{54}$$

Výběr pracovní množiny

To stále vyžaduje prohledání téměř celé kernelové matice, proto se v praxi prvek i vybere pomocí prvního řádu a prvek j pomocí druhého

$$i = \arg\max_{k \in I_{\mathbf{up}}} y_k g_k \tag{55}$$

$$j = \arg\max_{k \in I_{\mathbf{down}}} \frac{(y_i g_i - y_k g_k)^2}{2(K_{ii} + K_{kk} - 2K_{ik})}, \quad y_i g_i > y_k g_k \tag{56}$$

Nalezení j nyní vyžaduje pouze diagonálu kernelové matice a i-tý řádek, který je potřeba i pro aktualizaci gradientu na konci iterace.

Sequential Minimal Optimization

Algoritmus trénování SMO

1. Inicializace $\alpha_k = 0,\ g_k = 1,\ \forall k$
2. Výběr pracovní množiny (*working set*)

$$i = \arg\max_{k \in I_{\mathbf{up}}} y_k g_k$$

$$j = \arg\max_{k \in I_{\mathbf{down}}} \frac{(y_i g_i - y_k g_k)^2}{2(K_{ii} + K_{kk} - 2K_{ik})}, \quad y_i g_i > y_k g_k$$

3. Kontrola optimality
   Ukonči algoritmus když $\max_{i \in I_{\mathbf{up}}} y_i g_i - \min_{j \in I_{\mathbf{down}}} y_j g_j < \epsilon$
4. Direction search $\lambda \leftarrow \min \left\{ B_i - y_i \alpha_i, y_j \alpha_j - A_j, \frac{y_i g_i - y_j g_j}{K_{ii} + K_{jj} - 2K_{ij}} \right\}$
5. Optimizace $\rightarrow \Delta\alpha_i$ a $\Delta\alpha_j$
6. Aktualizace gradientu $g_k \leftarrow g_k - \lambda y_k K_{ik} + \lambda y_k K_{jk}, \forall k$
7. Aktualizace koeficientů $\alpha_i \leftarrow \alpha_i + y_i \lambda,\ \alpha_j \leftarrow \alpha_j - y_j \lambda$
8. Zpět na krok 2

Optimized Hierarchical Decomposition SVM

Cíl

Upravit algoritmus trénování pro dnešní technologie, zejména GPU. Dosavadní algoritmy neumožňují naplno využít možnosti dnešního hardware.

Předpoklady pro vhodný algoritmus

- Zpracovat najednou co nejvíce dat
- Převést co nejvíce kroků algoritmu na maticové nebo vektorové operace
- Vhodně pracovat s kernelovou maticí, protože se celá nevejde do paměti

Náš algoritmus

Pro splnění předpokladů jsme iterativní algoritmus SMO rozdělili do dvou úrovní, nazývaných lokální a globální.

1. Lokální úroveň je bežný SMO algoritmus, pro výběr dvojice prvků k optimalizaci se používá heuristika druhého řádu.
2. Globální úroveň vybírá z trénovací množiny pracovní množinu o předem dané velikosti *WS*. Tu pak optimalizuje SMO v lokální úrovni.

Hodnota *WS* je mocnina 2 taková, aby grafická karta spustila blok vláken s odpovídající velikostí a pomocné proměnné se vešly do sdílené paměti, tj. 1024 pro dnešní nejvýkonnější GPU.
Pro dostatečnou rychlost trénování je třeba v globální iteraci vhodně vybrat pracovní množinu. Používáme metodu výběru, kterou navrhl Serafini.

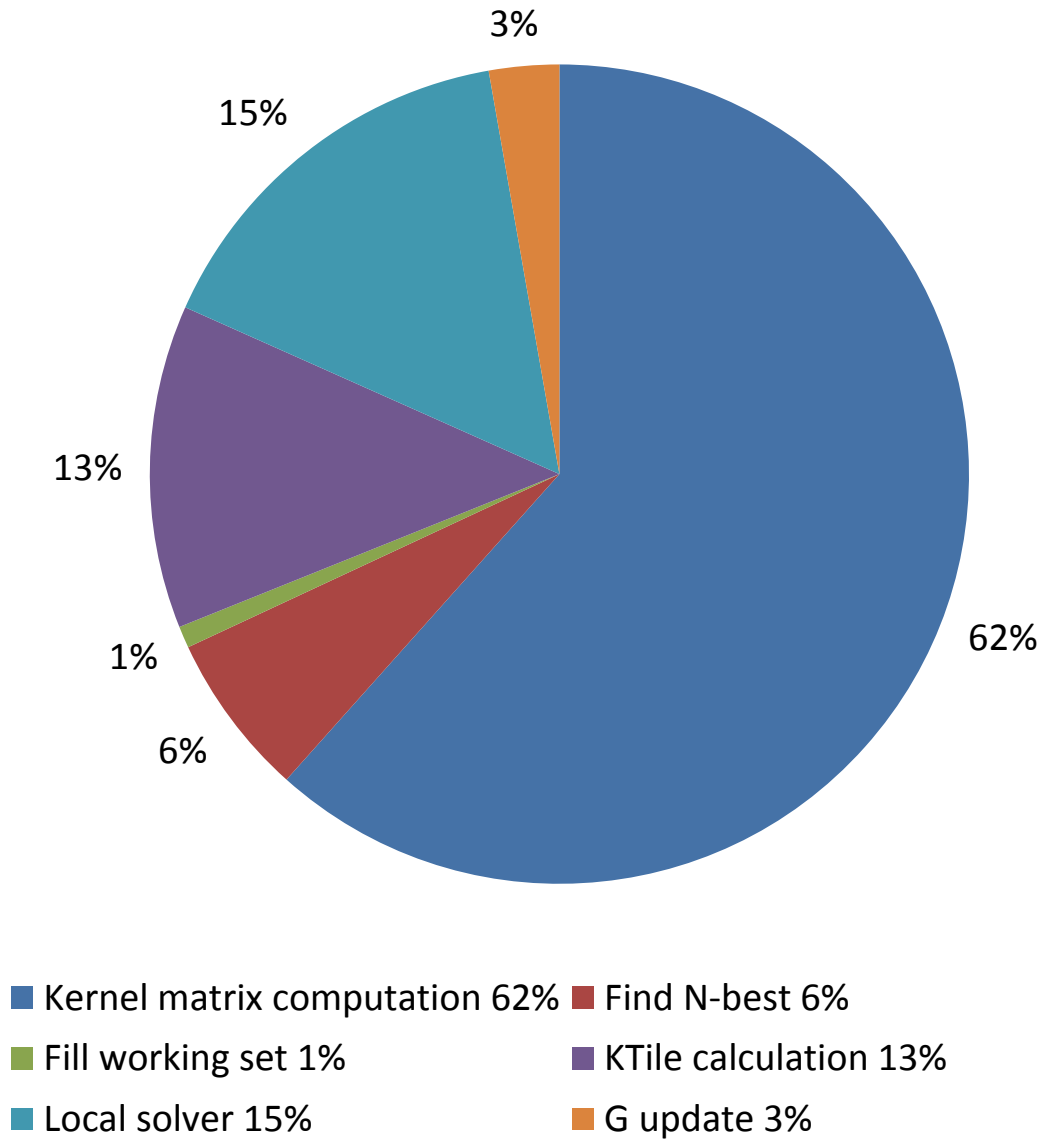
Metoda výběru pracovní množiny

Metoda výběru pracovní množiny

1. Definujme NC, obvykle rovnou WS/4
2. Výběr NC prvků z trénovací množiny pro i i j pomocí heuristiky prvního řádu
3. Zvolené prvky se vloží do nové pracovní množiny
4. Prvky ve staré pracovní množině se seřadí vzestupně podle počtu globálních iterací, ve kterých byly zvolené v pracovní množině
5. Nová pracovní množina se postupně zaplní prvky ze staré pracovní množiny. Nejdříve se volí volné vektory, poté lower bound a pak upper bound.

Tento algoritmus zvolí až 2NC nových bodů v každé globální iteraci a znovu použije alespoň polovinu pracovní množiny. Předpokládá se, že optimalizovaná pracovní množina po změně některých prvků už optimalizovaní není, ale je blízko řešení. Dále vyřazování prvků, které jsou v pracovní množině dlouho, eliminuje tzv. "zigzagging"mezi iteracemi.

Relativní čas jednotlivých částí trénovacího algoritmu



Tabulka: Čas trénování nejrychlejších implementací [s]

Implementace	Epsilon	Alpha	Adult	Web	COV1	MNIST	TIMIT
cuSVM	39.25	101.38	17.45	22.02	265.27	11.92	17.04
gpuSVM	28.04	26.26	3.36	7.58	301.04	5.89	2.86
multiSVM	51.34	90.04	18.69	22.52	WM[1] (191.05)	12.22	17.73
wuSVM	122.08	61.96	131.56	13.61	596.86	79.21	203.86
gtSVM LC	47.80	31.45	3.19	4.02	WM[1] (61.25)	3.72	2.43
gtSVM SC	77.22	36.38	3.37	4.43	WM[1] (112.12)	4.55	3.17
OHD-SVM	2.31	5.77	1.69	2.43	29.53	1.35	1.80

Tabulka: Přesnost natrénovaného modelu [%]

Implementace	Epsilon	Alpha	Adult	Web	COV1	MNIST	TIMIT
cuSVM	88.57	78.61	82.71	99.44	84.87	98.93	87.73
gpuSVM	88.57	78.59	82.71	99.44	84.85	98.91	87.70
multiSVM	88.59	78.62	82.71	99.44	WM[1] (62.75)	98.93	87.73
wuSVM	88.70	78.51	83.45	97.85	82.97	98.43	87.16
gtSVM LC	88.46	76.27	82.71	99.44	WM[1] (50.80)	98.93	87.72
gtSVM SC	88.53	77.14	82.71	99.44	WM[1] (62.97)	98.93	87.73
OHD-SVM	88.52	78.61	82.71	99.44	84.87	98.93	87.73

[1] WM znamená špatný model