UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

Kernel metódy a aplikácie

DIPLOMOVÁ PRÁCA

2018 Bc. Oliver Dendis

UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

Kernel metódy a aplikácie

DIPLOMOVÁ PRÁCA

Študijný program:	Ekonomicko-finančná matematika a modelovanie
Študijný odbor:	1114 Aplikovaná matematika
Školiace pracovisko:	Katedra aplikovanej matematiky a štatistiky
Vedúci práce:	doc. RNDr. Mária Trnovská, PhD.

Bratislava 2018 **Bc. Oliver Dendis**

71937020

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta:	Bc. Oliver Dendis
Študijný program:	ekonomicko-finančná matematika a modelovanie (Jednoodborové štúdium, magisterský II. st., denná forma)
Študijný odbor:	aplikovaná matematika
Typ záverečnej práce:	diplomová
Jazyk záverečnej práce:	slovenský
Sekundárny jazyk:	anglický

Názov: Kernel metódy a aplikácie.
Kernel methods and applications.

Cieľ: Cieľom práce je prehľadne spracovať uvedené metódy a algoritmy a ilustrovať ich fungovanie na vybraných aplikáciách.

Vedúci:	doc. RNDr. Mária Trnovská, PhD.
Katedra:	FMFI.KAMŠ - Katedra aplikovanej matematiky a štatistiky
Vedúci katedry:	prof. RNDr. Daniel Ševčovič, CSc.
Dátum zadania:	25.01.2017

Dátum schválenia: 27.01.2017 prof. RNDr. Daniel Ševčovič, CSc.
garant študijného programu

.....
študent

.....
vedúci práce

Poďakovanie Touto cestou sa chcem veľmi poďakovať svojej vedúcej diplomovej práce Doc. RNDr. Marii Trnovskej, PhD. za mimoriadnu ochotu, pomoc, odborné rady a podnetné pripomienky, vďaka ktorým vznikla táto práca. Ďakujem aj svojej rodine a priateľom za ich trpezlivosť a podporu počas celého vysokoškolského štúdia.

Abstrakt v štátnom jazyku

DENDIS, Oliver: Kernelove metódy a aplikácie [Diplomová práca], Univerzita Komenského v Bratislave, Fakulta matematiky, fyziky a informatiky, Katedra aplikovanej matematiky a štatistiky; školiteľ: doc. RNDr. Mária Trnovská, PhD., Bratislava, 2018, 48 s.

V našej práci sa zaoberáme využitím kernelových metód v technikách strojového učenia s dôrazom na ich geometrický význam. Popisujeme použitie kernelových funkcii v Support vector machine (SVM). Našim cieľom bolo prehľadne spracovať teóriu SVM a čiastočne SVR geometrickým prístupom, odvodiť a dokázať niektoré vzťahy a dostatočne zrozumiteľne graficky interpretovať niektoré teoretické výsledky. Rovnako sme sa pokúsili prehľadne spracovať matematický základ kernelových funkcii a prispieť vlastnými dôkazmi pri vlastnostiach konečných kernelových funkcii. Popri teoretickom spracovaní sme ukázali využitie kernelových metód v praxi na aplikáciu v strojovom učení.

Kľúčové slová: Support vector machines, Support vector regressioon, Kernelove metódy, Machine learning, Spracovanie prirodzeného jazyka

Abstract

DENDIS, Oliver: Kernel methods and applications [Master Thesis], Comenius University in Bratislava, Faculty of Mathematics, Physics and Informatics, Department of Applied Mathematics and Statistics; Supervisor: doc. RNDr. Mária Trnovská, PhD., Bratislava, 2011, 38p.

In our work we investigate the geometric approach to kernel methods with respect to machine learning. We describe applicability of kernel methods in SVM. Our goal was to process the theory of SVM and partially SVR with geometric approach, deduce and prove some relations with graphical interpretations. We also processed mathematical theory of kernel functions and proved some properties of finite kernel functions. We also chose and showed an application of kernel methods in field of machine learning.

Keywords: Support vector machines, Support vector regression, Kernel methods, Machine learning, Natural language processing

Obsah

Zoznam symbolov

$x^T y$ - Štandardný skalárny súčin, teda $\sum_{i=1}^{n} x_i y_i$.

$||x||_2$ - Euklidovská norma, $||x||_2 = \sqrt{\sum_{i=1}^{n} x_i^2}$.

$||x||_1$ - $||x||_2 = \sum_{i=1}^{n} |x_i|$.

Ω - Hilbertov priestor.

$\langle \cdot, \cdot \rangle_\Omega$ - Skalárny súčin na Ω.

$|| \cdot ||_\Omega$ - Norma na Ω.

L - Lagrangeova funkcia.

κ - Kernelová funkcia.

K - Kernelová matica.

ϕ - Charakteristické zobrazenie kernelovej funkcie.

$\mathbf{1}$ - Vektor samých jednotiek.

$\mathbf{1}\mathbf{1}^T$ - Matica samých jednotiek.

$A \odot B$ - Hadamardov súčin matíc A a B.

A_H^d - $\underbrace{A \odot A \odot ... \odot A}_{d}$.

e_H^A - Maticová exponenciála v zmysle Hadamardovho súčinu.

Úvod

Kernelové metódy je v strojovom učení trieda algoritmov používaných najmä na klasifikáciu a predikciu. Najznámejšia z týchto metód je metóda oporného bodu, v anglickej terminológii Support vector machine, ktorá sa primárne používa najmä na biklasifikáciu, resp. opakovanou aplikáciou algoritmu aj na multiklasifikáciu. Táto metóda vedie k optimalizačnému problému, resp. k duálnej úlohe, ktorá sa následne rieši. Túto metódu odvodili v roku 1995 a publikovali v článku autori Vapnik a Cortes [4], avšak nie vždy bolo jednoduché tieto úlohy riešiť. S nástupom modernej techniky a s novými výpočtovými technológiami nabrala SVM na popularite a táto metóda sa začala využívať na rôzne aplikácie strojového učenia.

Názov kernelové metódy je odvodený z podstaty algoritmu a to z kernelových funkcii. Tieto metódy nahradzujú vo formuláciách vyššie spomenutých optimalizačných úloh skalárny súčin kernelovou funkciou. Tento prístup vedie k tomu, že úlohu môžeme riešiť v inom priestore, častokrát s vyššou dimenziou. Zobrazením dostupných dát v inom priestore môžeme získať lepšie štruktúrované dáta, resp. v prípade, že dáta nie je možné lineárne separovať, v novom priestore budú lineárne separovateľné.

Našim cieľom bolo spracovať teóriu SVM a s využitím duality ponúknuť alternatívnu interpretáciu týchto úloh. Dualita medzi štandardnými SVM úlohami úzko súvisí s kernelovými metódami. Okrem teoretického prehľadu niektorých vlastností kernelových funkcii sme ukázali, že kernelové metódy možno chápať ako zovšeobecnenie štandardných optimalizačných techník, ktoré sa aplikujú po vnorení do iného (spravidla väčšieho) priestoru, v ktorom je skalárny súčin jednoznačne charakterizovaný kernelovou funkciou.

V prvej kapitole teda uvedieme formulácie úloh oporného bodu (SVM), prčom budeme vychádzať najmä z [2], [4] a [12] a taktiež formuláciu úlohy regresie pomocou metódy oporného bodu (SVR) [1]. V tejto kapitole ďalej uvedieme naše odvodenia duálnych úlohoch, pričom ich popíšeme s dôrazom na geometrickú interpretáciu.

V druhej kapitole sa venujeme teórii kernelových funkcii, jej všeobecnej definícií, ale aj definícií konečnej kernelovej funkcie. Definície, na ktorých sme stavali, vychádzali z [11],[13],[14],[15]. V tejtko kapitole navyše popíšeme základné vlastnosti týchto funkcii, pričom uvedieme naše dôkazy. Taktiež popíšeme základné a najčastejšie používané

kernelové funkcie spolu s popisom ich využívania.

V tretej kapitole prepojíme kernelové funkcie s teóriou SVM, naformulujeme optimalizačné úlohy tak, aby sme mohli použiť poznatky z druhej kapitoly, pričom formuláciou primárnej úlohy sa čiastočne inšpirujeme [17], pričom však túto úlohu modifikujeme pre zretelnejšie prepojenie kernelových funkcii a úlohy SVM. K týmto úlohám ďalej odvodíme duálne úlohy, pričom v ich formuláciách nahradíme skalárny súčin kernelovou funkciou. Tým získame problém, ktorý dokážeme zapísať v kompaktnom tvare a odvodíme podmienky, kedy je úloha riešiteľná.

V poslednej kapitole predstavíme aplikáciu kernelových metód. Pomocou programu, ktorý sme napísali v programovacom jazyku Python vytvoríme detektor autorstva textov, pričom využijeme práve SVM a rôzne kernelové funkcie s rozličnými parametrami. Na slovenských textoch z bakalárskych a diplomových prác sme sledovali 6 rôznych nami vybraných parametrov, podľa ktorých sme určovali či autor z trénovacej sady dát napísal aj niektoré texty z testovacej sady. Jednotlivé kernelové funkcie sme pri použití v aplikácii medzi sebou porovnali a výsledky graficky vyhodnotili. Inšpiráciu na výber sledovaných znakov sme čerpali z [13], [16],[18] a [19].

1 Geometrický náhľad na metódy oporného bodu

V tejto kapitole popíšeme a sformulujeme optimalizačné problémy lineárnej separácie a regresie s využitím metódy oporného bodu. Pri formulácii základných typov optimalizačných úloh sme vychádzali najmä z [1], [2], [4] a [12]. Následne k nim odvodíme duálne problémy, vďaka ktorým získame alternatívnu interpretáciu daných problémov, ktorú budeme interpretovať aj graficky.

1.1 Problém lineárnej separácie

Majme množinu X, ktorá obsahuje body $x_i \in R^n.i = 1, ...M$ a množinu Y, pričom jej prvky sú body $y_i \in R^n, i = 1, ...N$. Problém separácie je nájsť takú funkciu $f : R^n \to R$, ktorá pre body z množiny X nadobúda kladné hodnoty a pre body z množiny Y hodnoty záporné hodnoty, teda podľa [2] môžeme túto úlohu formálne zapísať:

$$\begin{aligned} f(x_i) > 0 \quad i = 1, ...N, \\ f(y_i) < 0 \quad i = 1, ...M. \end{aligned}$$

Pri probléme lineárnej separácie navyše požadujeme od funkcie f, aby bola afínna. Hľadáme teda takú funkciu $f(z) = a^T z - b$, pre ktorú platí:

$$\begin{aligned} f(x_i) = a^T x_i - b > 0 \quad i = 1, ...N, \\ f(y_i) = a^T y_i - b < 0 \quad i = 1, ...M. \end{aligned} \tag{1.1}$$

V takomto prípade nazývame nadrovinu $\{z | a^T z = b\}$ oddeľujúcou nadrovinou. Ak je takýchto nadrovín viac, najlepšou voľbou pre výber oddeľujúcej nadroviny bude vybrať takú, ktorá je od jednotlivých bodov oboch množín čo najďalej. Takto zabezpečíme robustný prístup k úlohe, ktorý je veľmi ľahko formulovateľný ako kvadratický optimalizačný problém, známy ako úloha prípustnosti. Pre účely zostavenia optimalizačnej úlohy však bude vhodnejšie nahradiť vo formulácii separácie ostré nerovnosti neostrými, pričom na pravej strane bude konštantné, kladné, zanedbateľné ϵ. Ak sú množiny separovateľné, tak je takáto formulácia separácie ekvivalentná s pôvodnou, preto:

$$\begin{aligned} a^T x_i - b \geq \epsilon, i = 1, ...N, \\ a^T y_i - b \leq -\epsilon, i = 1, ...M. \end{aligned} \tag{1.2}$$

Keď obe nerovnosti prenásobime $\frac{1}{\epsilon}$, o ktorom vieme, že je kladné a zvolíme nové označenie preškálovaných premenných $\widetilde{a} = \frac{a}{\epsilon}, \widetilde{b} = \frac{b}{b}$, môžeme úlohu prepísať do tvaru:

$$\begin{aligned} \widetilde{a}^T x_i - \widetilde{b} \geq 1, i = 1, ...N, \\ \widetilde{a}^T y_i - \widetilde{b} \leq -1, i = 1, ...M. \end{aligned} \tag{1.3}$$

Úloha (1.2) je úlohou lineárneho programovania, vieme ju teda vyriešiť.

1.2 Formulácia optimalizačného problému robustnej lineárnej separácie

Vďaka formulácii robustného prístupu k lineárnej separácii uvedenej v predošlej kapitole sa môžeme na túto úlohu pozerať ako na maximalizáciu vzdialeností pevne zvolených bodov od roviny, ktorej parametre nepoznáme, pričom ho môžeme podľa [2] formálne zapísať:

$$\begin{aligned} \max_{a,b,t} \quad & t \\ & a^T x_i - b \geq t \ , i = 1, ...N, \\ & a^T y_i - b \leq -t \ , i = 1, ...M. \end{aligned} \tag{1.4}$$

Keďže vhodným škálovaním afínnej funkcie na ľavej strane by sme mohli hodnotu t neustále zväčšovať bez ohraničenia, je potrebné normalizovať vektor a. Preto pridáme podľa [2] ohraničenie v tvare nerovnosti $\|a\| \leq 1$, úloha teda získa tvar:

$$\begin{aligned} \max_{a,b,t} \quad & t \\ & a^T x_i - b \geq t \ , i = 1, ...N, \\ & a^T y_i - b \leq -t \ , i = 1, ...M, \\ & \|a\|_2 \leq 1. \end{aligned} \tag{1.5}$$

Jedná sa o úlohu konvexnej optimalizácie, keďže maximalizujeme lineárnu funkciu na konvexnej množine. Ak sú množiny X, Y lineárne separovateľné, optimálna hodnota $\|t^\star\|_2$ je vždy väčšia ako 0. Je jednoduché ukázať, že v optime je vždy $\|a^\star\| = 1$. Keby totiž pre optimálne $t^\star$ platilo, že $\|a^\star\|_2 < 1$, môžeme nájsť konštantu $\alpha > 1$, takú že $\|\alpha a^\star\|_2 = 1$. Pre $\alpha a^\star$, $\alpha b^\star$ a $\alpha t^\star$ platí, že sú prípustné, keďže optimálne riešenie musí byť prípustné a prenásobenie jednotlivých nerovností kladnou konštantou ich

platnosť nijak neovplyvní. Keďže $\alpha > 1$, tak $\alpha t^\star > t^\star$. Získali sme teda prípustné riešenie, v ktorom nadobúda účelová funkcia väčšiu hodnotu ako v predošlom optime, čo je spor s definíciou optima. Preto v optimálnom riešení vyššie uvedenej úlohy je $\|a^\star\|_2 = 1$. To umožňuje lepší geometrický náhľad na úlohu. Aby to však oblo možné vidieť, je potrebné odvodiť vzdialenosť bodu x_i od oddeľujúcej nadroviny. Vzdialenosť tohto bodu od nadroviny je najkratšia vzdialenosť medzi x_i a akýmkoľvek bodom oddeľujúcej nadroviny. Takýto bod ležiaci na nadrovine je možné nájsť projekciou bodu x_i na oddeľujúcu nadrovinu. Ak bod projekcie označíme $\widehat{x_i}$, tak zrejme platí $a^T\widehat{x_i} = b$ a vzdialenosť bodu x_i od oddeľujúcej nadroviny je $\|x_i - \widehat{x_i}\|_2$. Keďže bod $\widehat{x_i}$ je kolmá projekcia bodu x_i, tak platí $x_i - \widehat{x_i} = \mathrm{k}a$. Prenásobením zľava vektorom a^T získame

$$a^T x_i - a^T\widehat{x_i} = \mathrm{k}a^T a.$$

Keďže $a^T\widehat{x_i} = b$ a $a^T a = \|a\|_2$, môžeme vyššie uvedenú rovnicu prepísať do tvaru

$$a^T x_i - b = \mathrm{k}\,\|a\|_2^2\,,$$

čím získame vyjadrenie pre k:

$$\mathrm{k} = \frac{a^T x_i - b}{\|a\|_2^2}.$$

Vďaka vyjadreniu pre k, môžeme rovnicu $x_i - \widehat{x_i} = \mathrm{k}a$ prepísať do tvaru:

$$x_i - \widehat{x_i} = \frac{a^T x_i - b}{\|a\|_2^2} a.$$

Pre vzdialenosť teda platí:

$$\|x_i - \widehat{x_i}\|_2 = \frac{|a^T x_i - b|}{\|a\|_2^2}\,\|a\|_2 = \frac{|a^T x_i - b|}{\|a\|_2},$$

čím sme získali vyjadrenie pre vzdialenosť bodu x_i od oddeľujúcej nadroviny. A keďže $\|a^\star\|_2 = 1$, tak v takom prípade je $a^T x_i - b$ presne euklidovská vzdialenosť bodu x_i od oddeľujúcej nadroviny. Rovnako je $-a^T y_i + b$ euklidovská vzdialenosť bodu y_i od tejto nadrodivny. Teda vyriešením optimalizačného problému (1.5) sme získali nadrovinu, ktorá oddeľuje množiny X a Y, pričom vzdialenosť od týchto množín je maximálna. Túto skutočnosť vyjadruje aj Obrázok 1.

Ak v účelovej funkcii nahradíme maximalizáciu minimalizáciou prevrátenej hodnoty

t a jednotlivé ohraničenia predelíme t, získame úlohu:

$$\begin{aligned} \min_{a,b,t} \quad & \frac{1}{t} \\ & \frac{a^T x_i}{t} - \frac{b}{t} \geq 1 \ , i = 1, ...N, \\ & \frac{a^T y_i}{t} - \frac{b}{t} \leq -1 \ , i = 1, ...M, \\ & \frac{\|a\|_2}{t} \leq \frac{1}{t}. \end{aligned} \tag{1.6}$$

Ak zvolíme nové preškálované premenné $\widetilde{a} = a/t$ a $\widetilde{b} = b/t$, potom aj $\|\widetilde{a}\| = \frac{\|a\|}{|t|} = \frac{\|\widetilde{a}\|}{t}$. Optimalizačná úloha s novými premennými vyzerá nasledovne:

$$\begin{aligned} \min_{\widetilde{a},\widetilde{b},t} \quad & \frac{1}{t} \\ & \widetilde{a}^T x_i - \widetilde{b} \geq 1 \ , i = 1, ...N, \\ & \widetilde{a}^T y_i - \widetilde{b} \leq -1 \ , i = 1, ...M, \\ & \|\widetilde{a}\|_2 \leq \frac{1}{t}. \end{aligned} \tag{1.7}$$

Keďže minimalizujeme premennú, ktorá je zároveň horným ohraničením pre $\|\widetilde{a}\|_2$ a v žiadnej inej z nerovností už nevystupuje, môžeme túto nerovnosť z ohraničení optimalizačnej úlohy odstrániť a priamo v účelovej funkcii požadovať minimalizáciu $\|\widetilde{a}\|_2$. Teda ekvivalentná úloha bude vyzerať:

$$\begin{aligned} \min_{\widetilde{a},\widetilde{b}} \quad & \|\widetilde{a}\|_2 \\ & \widetilde{a}^T x_i - \widetilde{b} \geq 1 \ , i = 1, ...N, \\ & \widetilde{a}^T y_i - \widetilde{b} \leq -1 \ , i = 1, ...M. \end{aligned} \tag{1.8}$$

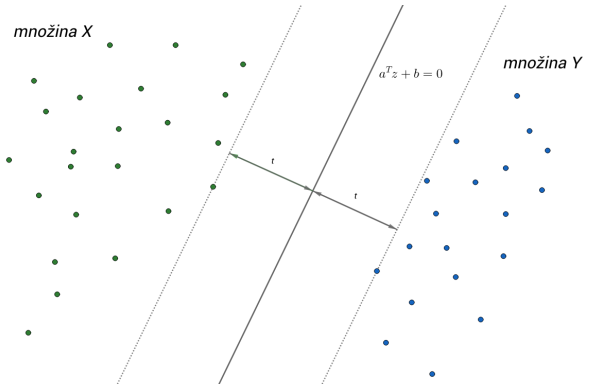
1.3 Duálna úloha k problému lineárnej separácie

V nasledujúcej podkapitole odvodíme duálnu úlohu k problému lineárnej separácie a popíšeme aj jej geometrickú interpretáciu. Pri odvodzovaní duálnych úloh v ďalších častiach tejto práce budeme často využívať lemu:

Lema 1: Pre infimum nasledujúcich funkcii platí:

(a)

$$\inf_x \quad c^T x = \begin{cases} 0 & c = 0, \\ -\infty & inak. \end{cases} \tag{1.9}$$



Obr. 1: Geometrický náhľad na optimalizačnú úlohu: našim cieľom je vybrať takú rovinu, aby vzdialenosť od najbližších bodov bola čo najväčšia

(b)

$$\inf_{x \geq 0} \quad c^T x = \begin{cases} 0 & c \geq 0, \\ -\infty & inak. \end{cases} \tag{1.10}$$

(c)

$$\inf_{x} \quad \alpha ||x||_2 + \beta^T x + \gamma = \begin{cases} \gamma & \alpha - ||\beta||_2 \geq 0, \\ -\infty & inak. \end{cases} \tag{1.11}$$

(d)

$$\alpha \geq 0 : \inf_{x} \quad \alpha x^T x + \beta^T x + \gamma = -\frac{\beta^T \beta}{2\alpha} + \gamma. \tag{1.12}$$

Dôkaz: Tvrdenia (a) a (b) sú triviálne.

(c) Pri tomto tvrdení použijeme Cauchy-Schwarzovu nerovnosť na člen $\beta^T x$, teda bude platiť

$$|\beta^T x| \leq \|\beta\|_2 \|x\|_2 ,$$

vďaka čomu môžeme výraz $\beta^T x$ ohraničiť:

$$- \|\beta\|_2 \|x\|_2 \leq \beta^T x \leq \|\beta\|_2 \|x\|_2 .$$

Zameriame sa na prvú nerovnosť. Po pričítaní členu $\alpha||x||_2 + \gamma$ k obom stranám nerovnosti a vyňatí $||x||_2$ v ľavej strane nerovnosti získame:

$$||x||_2(-||\beta||_2 + \alpha) + \gamma \leq \beta^T x + \alpha||x||_2 + \gamma.$$

Táto nerovnosť platí všeobecne, teda aj pre infimum oboch výrazov v nerovnosti:

$$\inf_{x} \quad ||x||_2(-||\beta||_2 + \alpha) + \gamma \leq \inf_{x} \quad \beta^T x + \alpha||x||_2 + \gamma. \tag{1.13}$$

Zrejme platí:

$$\inf_x \quad ||x||_2(-||\beta||_2+\alpha) = \begin{cases} 0 & \alpha - ||\beta||_2 \geq 0, \\ -\infty & inak. \end{cases} \tag{1.14}$$

Pripočítaním konštanty γ k hodnote infima funkcie $||x||_2(-||\beta||_2+\alpha)$ získame hodnotu infima $||x||_2(-||\beta||_2+\alpha)+\gamma$. Podľa (1.13) sme získali dolné ohraničenie infima funkcie $\beta^T x + \alpha||x||_2 + \gamma$ pre dva rôzne prípady a keďže v oboch prípadoch táto funkcia dané hodnoty dosahuje, našli sme najväčšie dolné ohraničenie tejto funkcie, čím sme dokázali tvdenie (c). (d) Pri dôkaze (1.12) využijeme konvexnosť funkcie $\alpha x^T x + \beta^T x + \gamma$. Zderivovaním podľa x nájdeme stacionárny bod, pre ktorý platí $\widehat{x} = \frac{-\beta}{2\alpha}$. Dosadením tejto hodnoty dostaneme práve hodnotu $-\frac{\beta^T\beta}{2\alpha} + \gamma$, čo bolo treba dokázať. □

Ak v optimalizačnej úlohe (1.5) maximalizáciu t nahradíme minimalizáciou $-t$, Lagrangeova funkcia získa tvar:

$$L(a,b,t;\alpha,\beta,\lambda) = -t + \sum_{i=1}^{N}\alpha_i(t+b-a^T x_i) + \sum_{i=1}^{M}\beta_i(t-b+a^T y_i) + \lambda(\|a\|-1), \tag{1.15}$$

pričom $\alpha \succeq 0, \beta \succeq 0, \lambda \geq 0$. Pre formuláciu duálnej funkcie je potrebné nájsť infimum lineárnej Lagrangeovej funkcie cez primárne premenné a, b, t. Vyňatím primárnych premenných z pôvodnej formulácie Lagranegovej úlohy získame tvar:

$$L(a,b,t;\alpha,\beta,\lambda) = t\Big(-1+\sum_{i=1}^{N}\alpha_i + \sum_{i=1}^{M}\beta_i\Big) + b\Big(\sum_{i=1}^{N}\alpha_i - \sum_{i=1}^{M}\beta_i\Big) + a^T\Big(\sum_{i=1}^{M}\beta_i y_i - \sum_{i=1}^{N}\alpha_i x_i\Big) + \lambda(\|a\|-1). \tag{1.16}$$

Keďže funkcia L je separovateľná v a, b, t, teda $L(a,b,t;\alpha,\beta,\lambda) = L_1(a;\alpha,\beta,\lambda) + L_2(b;\alpha,\beta,\lambda) + L_3(t;\alpha,\beta,\lambda)$, tak pre infimum platí

$$\inf_{a,b,t} \quad L(a,b,t;\alpha,\beta,\lambda) = \inf_a \quad L_1(a;\alpha,\beta,\lambda) + \inf_b \quad L_2(b;\alpha,\beta,\lambda) + \inf_t \quad L_3(t;\alpha,\beta,\lambda). \tag{1.17}$$

Hľadanie infima funkcie $L_1(t;\alpha,\beta,\lambda)$ je vlastne úloha (1.9) s jednorozmenrou premennou t. Preto pre infimum platí, že je rovné 0 ak:

$$-1+\sum_{i=1}^{N}\alpha_i + \sum_{i=1}^{M}\beta_i = 0, \tag{1.18}$$

inak $-\infty$. Podobne, pre funkciu $L_2(b;\alpha,\beta,\lambda)$ s jednorozmernou premennou b platí, že pokiaľ platí:

$$\sum_{i=1}^{N}\alpha_i - \sum_{i=1}^{M}\beta_i = 0, \tag{1.19}$$

tak je infimum rovné 0, inak $-\infty$.

Ak označíme :

$$z = \sum_{i=1}^{M}\beta_i y_i - \sum_{i=1}^{N}\alpha_i x_i,$$

tak o funkcii $L_3(t;\alpha,\beta,\lambda) = a^T z + \lambda||z||_2 - \lambda$ vďaka (1.11) vieme, že pre jej infimum platí:

$$\inf_{a} \quad L_3(a,\alpha,\beta,\lambda) = \begin{cases} -\lambda; & \lambda - \|z\| \geq 0, \\ -\infty; & inak. \end{cases}$$

Vďaka vlastnosti 1.17 teda môžeme napísať infimum funkcie ako:

$$\inf_{a,b,t} \quad L(a,b,t;\alpha,\beta,\lambda) = \begin{cases} -\lambda; & \sum_{i=1}^{N}\alpha_i - \sum_{i=1}^{M}\beta_i = 0, \\ & -1 + \sum_{i=1}^{N}\alpha_i + \sum_{i=1}^{M}\beta_i = 0, \\ & \lambda - \left|\left| \sum_{i=1}^{M}\beta_i y_i - \sum_{i=1}^{N}\alpha_i x_i \right|\right| \geq 0, \\ -\infty; & inak. \end{cases} \tag{1.20}$$

Duálna úloha teda bude vyzerať:

$$\begin{aligned} \max \quad & -\lambda \\ & \sum_{i=1}^{N}\alpha_i - \sum_{i=1}^{M}\beta_i = 0, \\ & \sum_{i=1}^{N}\alpha_i + \sum_{i=1}^{M}\beta_i = 1, \\ & \left\| \sum_{i=1}^{M}\beta_i y_i - \sum_{i=1}^{N}\alpha_i x_i \right\| \leq \lambda, \\ & \alpha,\beta \succeq 0. \end{aligned} \tag{1.21}$$

Formuláciu úlohy je možné taktiež zjednodušiť tým, že premennú λ nahradíme priamo jej dolným ohraničením. Ak navyše úlohu prevedieme na minimalizačnú, môžeme

získať túto úlohu v ekvivalentnom tvare:

$$\begin{aligned} \min \quad & \left\| \sum_{i=1}^{M} \beta_i y_i - \sum_{i=1}^{N} \alpha_i x_i \right\| \\ & \sum_{i=1}^{N} \alpha_i - \sum_{i=1}^{M} \beta_i = 0, \\ & \sum_{i=1}^{N} \alpha_i + \sum_{i=1}^{M} \beta_i = 1, \\ & \alpha, \beta \succeq 0. \end{aligned} \tag{1.22}$$

Vyriešením sústavy rovníc získaných z ohraničení úlohy získame vzťahy:

$$\begin{aligned} \sum_{i=1}^{N} \alpha_i &= \frac{1}{2}, \\ \sum_{i=1}^{M} \beta_i &= \frac{1}{2}. \end{aligned} \tag{1.23}$$

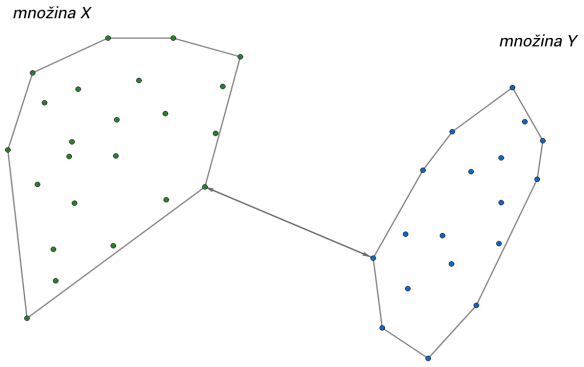
a pri preškálovaní premenných $\widetilde{\alpha}_i = 2\alpha_i, i = 1, ...N$ a $\widetilde{\beta} = 2\beta, i = 1, ...M$ vyzerá úloha geometricky intuitívnejšie:

$$\begin{aligned} \min \quad & \frac{1}{2} \left\| \sum_{i=1}^{M} \widetilde{\beta}_i y_i - \sum_{i=1}^{N} \widetilde{\alpha}_i x_i \right\| \\ & \sum_{i=1}^{N} \widetilde{\alpha}_i = 1, \\ & \sum_{i=1}^{M} \widetilde{\beta}_i = 1, \\ & \widetilde{\alpha}, \widetilde{\beta} \succeq 0. \end{aligned} \tag{1.24}$$

Na základe tejto formulácie je možné vidieť, že duálna úloha k úlohe lineárnej separácie je hľadanie vzdialenosti konvexných obalov pôvodných množín X a Y.

1.4 Prípad lineárne neseparovateľných množín

Vo všeobecnosti, v problémoch reálneho sveta sú často množiny lineárne neseparovateľné. Uvažujme formuláciu problému separácie ako v (1.4). V takomto prípade je možné pristupovať k úlohe separácie veľmi podobne ako je to pri lineárne separovateľných množinách avšak s pridaním nových nezáporných „slackových“ premenných u



Obr. 2: Geometrický náhľad na duálnu úlohu k problému lineárnej separácie: Našim cieľom je hľadať vzájomnú vzdialenosť konvexných obalov jednotlivých množín

a v do ohraničení. Takýmto spôsobom získali nové ohraničenia aj pôvodní autori teórie SVM v [4]. Nové ohraničenia tak budú vyzerať:

$$\begin{aligned} a^T x_i - b &\geq 1 - u_i \ , i = 1, ...N, \\ a^T y_i - b &\leq -1 + v_i \ , i = 1, ...M. \end{aligned} \tag{1.25}$$

Pridaním nových premenných teda relaxujeme jednotlivé ohraničenia, vďaka tomu v prípade neseparovateľných množín môže pre niektoré body platiť $a^T x_i - b < 1$, resp. $a^T y_i - b < -1$. Cieľom je čo najlepšie aproximovať úlohu (1.8) pre prípad neseparovateľných množín. Chceme teda aby „slackové“ premenné boli čo najmenšie a zároveň aby bola splnená podmienka v ohraničeniach. Z vyššie popísaného postupu je možné škonštruovať optimalizačný problém vychádzajúc z úlohy o lineárne separovateľných množinách. Táto metóda separácie sa nazýva metóda oporného bodu (SVM,[4]).

Keďže je našim cieľom minimalizovať pomocné premenné, do účelovej funkcie pridáme penalizačnú funkciu a to jednotkové normy vektorov „slackových“ premenných vynásobený konštantou C upravujúcou váhu týchto premenných. Ohraničenia budú vyzerať veľmi podobne ako v úlohe o separovateľných množinách s tým rozdielom, že pravá strana bude brať do úvahy aj „slackové“ premenné. Optimalizačný problém bude teda vyzerať:

$$\begin{aligned} \min_{a,b} \quad & ||a||_2 + C||u||_1 + C||v||_1 \\ & a^T x_i - b \geq 1 - u_i \ , i = 1, ...N, \\ & a^T y_i - b \leq -1 + v_i \ , i = 1, ...M. \end{aligned} \tag{1.26}$$

resp. v rozpísanom tvare, kde je potrebné pridať ohraničenie na nezápornosť dopln-

kových premenných.

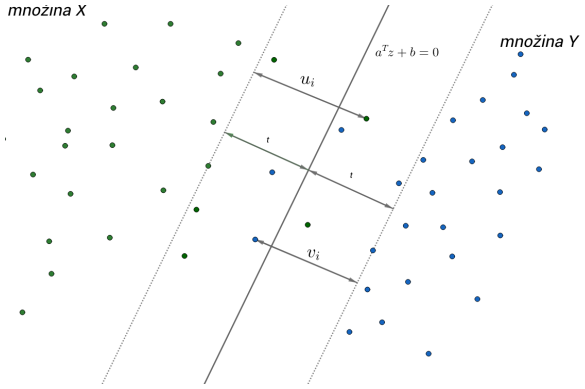
$$\begin{aligned} \min_{a,b} \quad & ||a||_2 + C\sum_{i=1}^{N} u_i + C\sum_{i=1}^{M} v_i \\ & a^T x_i - b \geq 1 - u_i \ , i = 1, ...N, \\ & a^T y_i - b \leq -1 + v_i \ , i = 1, ...M, \\ & u, v \succeq 0. \end{aligned} \tag{1.27}$$

čím získame totožné vzťahy ako sú uvedené v [2] alebo [12]. Táto formulácia s penalizačnou funkciou v tvare súčtu jednotkových noriem vektorov "slackových"premenných sa používa najčastejšie. Výhodou je dobrá geometrická interpretácia duálnej úlohy. Môžeme však zvoliť aj inú penalizačnú funkciu. Ohraničenie o nezápornosti "slackových"premenných je možné odstrániť ak v účelovej funkcii nahradíme súčet jednotkových noriem vektorov "slackových"premenných súčtom druhých mocnín ich l_2 noriem. Keby totiž akákoľvek slack premenná bola menšia ako nula, tak pravá strana príslušného ohraničenia by bola väčšia ako 1. V takom prípade by to však nemohlo byť optimum, pretože akékoľvek väčšia a zároveň záporná hodnota slack premennej by viedla k nižšej hodnote účelovej funkcie a stále by spĺňala ohraničenie. Takto získame inú formuláciu problému separácie so "slackovými"premennými a je možné ju zapísať takto:

$$\begin{aligned} \min_{a,b} \quad & ||a||_2 + C\sum_{i=1}^{N} u_i^2 + C\sum_{i=1}^{M} v_i^2 \\ & a^T x_i - b \geq 1 - u_i \ , i = 1, ...N, \\ & a^T y_i - b \leq -1 + v_i \ , i = 1, ...M. \end{aligned} \tag{1.28}$$

Na obrázku č.3 je možné vidieť ilustráciu separácie množín v R^2, ktoré su lineárne neseparovateľné.

Pozn.: Úloha (1.27) je úlohou lineárneho programovania, pričom využívame $\ell 1$ regularizáciu premenných u, v, čo má za následok, že vektory u a v sú v optime riedke, t.j. majú veľa núl. V úlohe (1.28) sa využíva tzv. Tichonovovská regularizácia premenných u, v a je to úloha kvadratického programovania.



Obr. 3: Geometrický náhľad na relaxovaný problém: Našim cieľom je vybrať také oporné nadroviny, aby boli od seba čo najďaleja zároveň aby súčet slackov bol čo najmenší

1.5 Duálna úloha k problému lineárne neseparovateľných množín

V nasledujúcej podkapitole ponúkneme naše vlastné odvodenie duálnej úlohy problému 1.7 a jeho geometrickú interpretáciu aj pomocou obrázkov. Ak vo formulácií problému 1.7 nahradíme účelovú funkciu prevrátenou hodnotou t, získa úloha tvar:

$$\begin{aligned} \min_{a,b} \quad & \frac{1}{t} \\ & a^T x_i - b \geq 1 - u_i \ , i = 1, ...N, \\ & a^T y_i - b \leq -1 + v_i \ , i = 1, ...M, \\ & ||a||_2 + C \sum_{i=1}^{N} u_i + C \sum_{i=1}^{M} v_i \leq \frac{1}{t}, \\ & u, v \succeq 0. \end{aligned} \tag{1.29}$$

úlohu môžeme prevrátením účelovej funkcie pretransformovať na maximalizačnú a jednotlivé ohraničenia prenásobiť t, o ktorom vieme, že je kladné, teda úlohu 1.29 môžeme prepísať do tvaru:

$$\begin{aligned} \max_{a,b,t} \quad & t \\ & ta^T x_i - bt \geq t - tu_i \ , i = 1, ...N, \\ & ta^T y_i - bt \leq -t + tv_i \ , i = 1, ...M, \\ & t||a||_2 + tC \sum_{i=1}^{N} u_i + tC \sum_{i=1}^{M} v_i \leq 1, \\ & tu, tv \succeq 0. \end{aligned} \tag{1.30}$$

Ak označíme vo formulácii predošlej úlohy $\widetilde{a} = ta, \widetilde{b} = tb, \widetilde{u_i} = tu_i$ a $\widetilde{v_i} = tv_i$ potom aj $\|\widetilde{a}\| = \|a\| \, |t| = \|\widetilde{a}\| \, t$, môžeme úlohu prepísať na:

$$\begin{aligned} \min_{a,b,t} \quad & -t \\ & \widetilde{a}^T x_i - \widetilde{b} \geq t - \widetilde{u_i} \, , i = 1, ...N, \\ & \widetilde{a}^T y_i - \widetilde{b} \leq -t + \widetilde{v_i} \, , i = 1, ...M, \\ & ||\widetilde{a}||_2 + C \sum_{i=1}^{N} \widetilde{u_i} + C \sum_{i=1}^{M} \widetilde{v_i} \leq 1, \\ & \widetilde{u}, \widetilde{v} \succeq 0. \end{aligned} \tag{1.31}$$

Z formulácie 1.31 môžeme zostaviť Lagranegovú funkciu:

$$L(\widetilde{a}, \widetilde{b}, t, \widetilde{u}, \widetilde{v}; \alpha, \beta, \lambda) = -t + \sum_{i=1}^{N} \alpha_i (t + \widetilde{b} - \widetilde{a}^T x_i - \widetilde{u_i}) + \sum_{i=1}^{M} \beta_i (t - \widetilde{b} + \widetilde{a}^T y_i - \widetilde{v_i}) + \lambda(\|a\|_2 - 1 - C \sum_{i=1}^{N} \widetilde{u_i} - C \sum_{i=1}^{M} \widetilde{v_i}), \tag{1.32}$$

pričom $\alpha \succeq 0, \beta \succeq 0, \lambda \geq 0$. Pre formuláciu duálnej funkcie je potrebné nájsť infimum lineárnej Lagrangeovej funkcie cez premenné a, b, t, u, v. Vyňatím primárnych premenných z pôvodnej formulácie Lagranegovej úlohy získame:

$$L(\widetilde{a}, \widetilde{b}, t, \widetilde{u}, \widetilde{v}; \alpha, \beta, \lambda) = t\Big(-1 + \sum_{i=1}^{N} \alpha_i + \sum_{i=1}^{M} \beta_i \Big) + \widetilde{b}\Big(\sum_{i=1}^{N} \alpha_i - \sum_{i=1}^{M} \beta_i \Big) + \left[\widetilde{a}^T (\sum_{i=1}^{M} \beta_i y_i - \sum_{i=1}^{N} \alpha_i x_i) + \lambda \left\| \widetilde{a} \right\|_2 \right] + \widetilde{u}^T (\lambda C \mathbf{1} - \alpha) + \widetilde{v}^T (\lambda C \mathbf{1} - \beta) - \lambda. \tag{1.33}$$

Podobne ako v prípade lineárne separovateľných množín, táto funkcia je separovateľná a to v premenných a, b, t, u, v a v premenných b, t, u, v je lineárna. Analogickým postupom ako v kapitole 1.3 dokážeme pomocou (1.9) nájsť infimum cez premenné b, t a pomocou (1.11) infimum cez a. Keďže pre premenné u, v platí, že sú nezáporné, tak dokážeme nájsť infimum veľmi jednoducho, pretože je to úloha (1.10). Vďaka separovateľnosti $L(\widetilde{a}, \widetilde{b}, t, \widetilde{u}, \widetilde{v}; \alpha, \beta, \lambda)$ môžeme infimum definovať ako:

$$\inf_{a,b,t,u\succeq 0,v\succeq 0} \quad L(\widetilde{a},\widetilde{b},t,\widetilde{u},\widetilde{v};\alpha,\beta,\lambda) = \begin{cases} -\lambda; & \sum_{i=1}^{N}\alpha_i - \sum_{i=1}^{M}\beta_i = 0, \\ & -1 + \sum_{i=1}^{N}\alpha_i + \sum_{i=1}^{M}\beta_i = 0, \\ & \lambda - \left|\left| \sum_{i=1}^{M}\beta_i y_i - \sum_{i=1}^{N}\alpha_i x_i \right|\right|_2 \geq 0, \\ & \alpha \preceq C\lambda\mathbf{1}, \\ & \beta \preceq C\lambda\mathbf{1}, \\ -\infty; & inak. \end{cases} \tag{1.34}$$

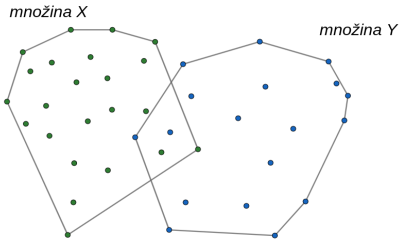
Analogickým postupom ako v prípade lineárne separovateľných množín môžeme teda formulovať duálnu úlohu:

$$\begin{aligned} \max \quad & -\lambda \\ & \sum_{i=1}^{N}\alpha_i = \sum_{i=1}^{M}\beta_i = \frac{1}{2}, \\ & \left\| \sum_{i=1}^{M}\beta_i y_i - \sum_{i=1}^{N}\alpha_i x_i \right\|_2 \leq \lambda, \\ & 0 \preceq \alpha \preceq C\lambda\mathbf{1}, \\ & 0 \preceq \beta \preceq C\lambda\mathbf{1}, \end{aligned} \tag{1.35}$$

pri preškálovaní premenných $\widetilde{\alpha_i} = 2\alpha_i, i = 1,...N, \widetilde{\beta_i} = 2\beta_i, i = 1,...M$ a $\widetilde{\lambda} = 2\lambda$ a transformácií na minimalizačnú úlohu vyzerá optimalizačný problém nasledovne:

$$\begin{aligned} \min \quad & \widetilde{\lambda} \\ & \sum_{i=1}^{N}\widetilde{\alpha}_i = \sum_{i=1}^{M}\widetilde{\beta}_i = 1, \\ & \left\| \sum_{i=1}^{M}\widetilde{\beta}_i y_i - \sum_{i=1}^{N}\widetilde{\alpha}_i x_i \right\|_2 \leq \widetilde{\lambda}, \\ & 0 \preceq \widetilde{\alpha} \preceq C\widetilde{\lambda}\mathbf{1}, \\ & 0 \preceq \widetilde{\beta} \preceq C\widetilde{\lambda}\mathbf{1}. \end{aligned} \tag{1.36}$$

Z ohraničení je zrejmé, že platí $C\widetilde{\lambda} \geq 0$. Pokiaľ je $C\widetilde{\lambda} \geq 1$, horné ohraničenie na $\widetilde{\alpha}, \widetilde{\beta}$ je redundantné a teda problém sa dá geometricky interpretovať ako hľadanie dvoch bodov, jedného z konvexného obalu množiny X, druhého z konvexného obalu množiny



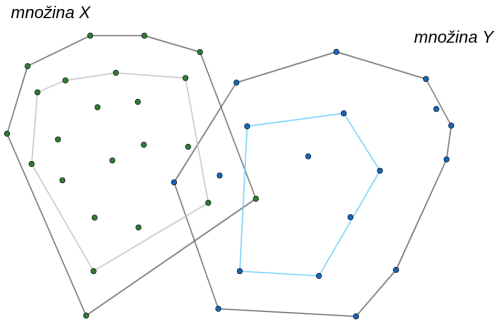
Obr. 4: Konvexné obaly v prípade lineárne neseparovateľných množín sa prekrývajú

Y, pričom sa tieto obaly prekrývajú, teda majú spoločné body, čo je možné vidieť na Obrázku 4. To znamená ze stačí zvoliť také duálne premenné, aby obe konvexné kombinácie množín vyjadrovali ten istý bod. Táto úloha je teda určite prípustná a v optime platí, že $\widetilde{\lambda}C = 1$.

Ak sa pozrieme na prípad hľadania minima problému (1.36) pre $\widetilde{\lambda}C < 1$, horné ohraničenia na $\widetilde{\alpha}$ a $\widetilde{\beta}$ sú v tejto úlohe podstatné. Keďže redukovaný konvexný obal mnoziny Z definujeme podľa [11] ako:

$$\Big\{ \sum_{i=1}^{K} \delta_i z_i | \sum_{i=1}^{K} \delta_i = 1; 0 \leq \delta_i \leq \mu; \mu < 1 \Big\}.$$

Na úlohu (1.36) je možné teda nazerať geometricky ako na hľadanie vzdialenosti redukovaných konvexných obalov množín X a Y, obrázková interpretácia je na Obrázku 5.



Obr. 5: Geometrický náhľad na separáciu lineárne separovateľných množín: optimalizačný problém je možné geometricky interpretovať ako hľadanie vzdialenosti redukovaných konvexných obalov

1.6 Metóda oporného bodu v regresii

Metóda oporného bodu môže byť použitá aj v regresií, v anglickej literatúre sa často pomenuváva ako Support vector regression, resp. SVR, inšpirovali sme sa [1]. Majme teda množinu W, ktorú tvorí N bodov $w_i = (x_i, y_i); i = 1, ..., N$, pričom $x_i \in \mathbb{R}^m$ a $y_i \in \mathbb{R}$. Narozdiel od klasickej regresie je našim cieľom nájsť pre body množiny W takú regresnú nadrovinu $f(z) = a^T z + b$, aby pre vopred dané ϵ platilo $|y_i - f(x_i)| \leq \epsilon, \forall x_i \in X$. Od regresnej roviny teda požadujeme, aby boli jednotlivé reziduá zhora ohraničené hodnotou ϵ.

Najnižšiu možnú hodnotu ϵ, pre ktorú je možné zostaviť prípustnú úlohu SVR môžeme nájsť vyriešením optimalizačného problému, ktorý možno naformulovať takto:

$$\begin{aligned} \min \quad & \epsilon \\ & y_i - a^T x_i - b \leq \epsilon; i = 1, .., N, \\ & -y_i + a^T x_i + b \leq \epsilon; i = 1, .., N. \end{aligned} \tag{1.37}$$

Optimálne riešenie tejto úlohy môžeme označiť $\hat{\epsilon}$. Ak teda zvolíme $\epsilon \geq \hat{\epsilon}$, regresná rovina bude určite existovať. Ak je zvolené ϵ dostatočne veľké, regresných rovín môže byť nekonečne veľa. Našim cieľom by mala byť taká nadrovina, pre ktorú je $||a||_2$ čo najmenšie. Keby totiž niektorá zložka a bola veľmi veľká, mohla by zásadne ovplyvňovať veľkosť $a^T x_i$. Pre fixne dané ϵ môžeme teda naformulovať takto:

$$\begin{aligned} \min_{a,b} \quad & ||a||_2 \\ & y_i - a^T x_i - b \leq \epsilon; i = 1, .., N, \\ & -y_i + a^T x_i + b \leq \epsilon; i = 1, .., N. \end{aligned} \tag{1.38}$$

ak presunieme ϵ v ohraničeniach úlohy (1.38) na ľavú stranu získame ekvivalentné ohraničenia:

$$\begin{aligned} & y_i - \epsilon - a^T x_i - b \leq 0; i = 1, .., N, \\ & -(y_i + \epsilon) + a^T x_i + b \leq 0; i = 1, .., N. \end{aligned} \tag{1.39}$$

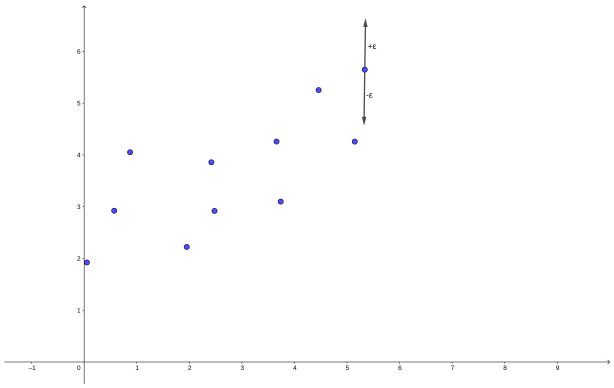
Ak zapíšeme tieto podmienky vektorovo, tak platí:

$$\begin{aligned} & (-a, 1)^T (x_i, y_i - \epsilon) - b \leq 0; i = 1, .., N, \\ & (a, -1)^T (x_i, y_i + \epsilon) + b \leq 0; i = 1, .., N. \end{aligned} \tag{1.40}$$

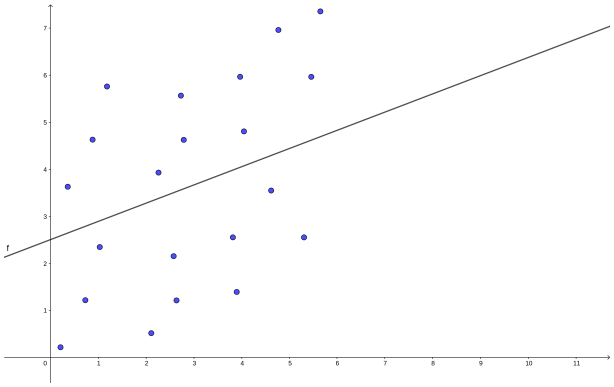
Substitúciou $\widetilde{a} = (a, -1)$ a prenásobením prvej nerovnosti -1 získame podmienky:

$$\begin{aligned} \widetilde{a}^T(x_i, y_i - \epsilon) - b \geq 0; i = 1, .., N, \\ \widetilde{a}^T(x_i, y_i + \epsilon) - b \leq 0; i = 1, .., N. \end{aligned} \tag{1.41}$$

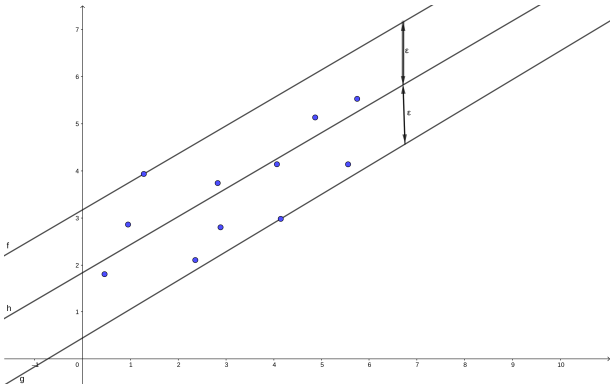
Ako je možné vidieť, vyššie uvedená formulácia priamo určuje separujúcu nadrovinu $g(z) = \widetilde{a}^T z - b$, ktorá oddeluje množinu $W_\epsilon = \left\{(x_i, y_i + \epsilon)|(x_i, y_i) \in W\right\}$ a množinu $W_{\epsilon-} = \left\{(x_i, y_i - \epsilon)|(x_i, y_i)\right\} \in W$. Problém SVR teda môžeme geometricky intrepretovať ako úlohu separácie 2 množín, pričom prvná vznikne z pôvodnej množiny W pridaním ku zložke y každého bodu množiny W fixne danú konštrantu ϵ a druhá vznikne taktiež z pôvodnej množiny W a to pridaním ku zložke y každého bodu fixne danú konštrantu $-\epsilon$. Z každého bodu získame teda jeden bod množiny W_ϵ a jeden bod množiny $W_{-\epsilon}$. Situáciu reflektujú Obrázok 6,Obrázok 7 a Obrázok 8.



Obr. 6: Z každého bodu množiny W získame posunutím jeden bod, ktorý bude patriť $W_{+\epsilon}$ a jeden bod, ktorý bude patriť do $W_{-\epsilon}$



Obr. 7: Úloha SVR je vlastne úloha na separáciu vzniknutých množín $W_{+\epsilon}$ a $W_{-\epsilon}$

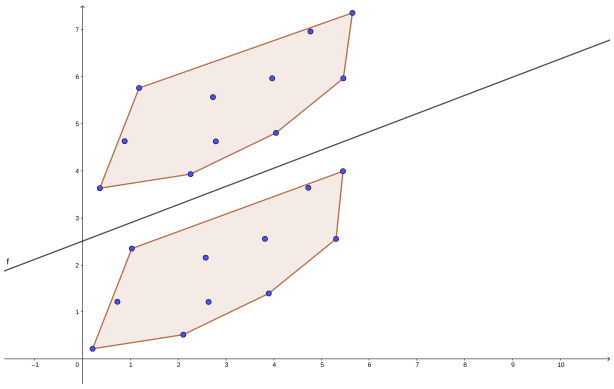


Obr. 8: Vzniknutá separujúca nadrovina je zároveň regresnou rovinou pre body množiny W

Duálnu úlohu môžeme teda interpretovať rovnako ako v kapitole 1.3, teda ako hľadanie konvexných obalov dvoch množín.Teda formálne zapísané:

$$\begin{aligned} \min \quad & \frac{1}{2} \left\| \beta^T (x_i, y_i + \epsilon) - \alpha^T (x_i, y_i - \epsilon) \right\| \\ & \sum_{i=1}^{M+1} \alpha_i = 1, \\ & \sum_{i=1}^{M+1} \beta_i = 1, \\ & \alpha, \beta \succeq 0. \end{aligned} \tag{1.42}$$

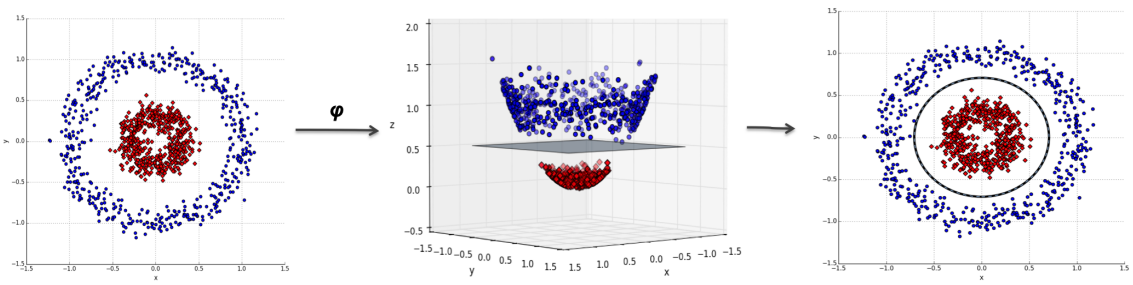
Geometrický pohľad je možné vidieť na Obrázku 9.



Obr. 9: Duálna úloha SVR je vlastne hľadanie vzdialenosti konvexných obalov množín $W_{+\epsilon}$ a $W_{-\epsilon}$

2 Teória kernelových funkcii

V tejo kapitole predstavíme teóriu kernelových funkcii, jej súvislosť so skalárnym súčinom a Hilbertovým priestorom. Taktiež predstavíme niektoré základné aplikácie kernelových funkcii, ktorým sa však nebudeme nejak zásadne venovať. Cieľom tejto kapitoly je vybudovať teóriu kernelových metód kvôli tzv. "kernelovému triku", ktorý je možné použiť v ďalších kapitolách. Hlavnou myšlienkou tohto triku je zobrazenie dostupných dát v inom priestore vďaka čomu môžu byť tieto dáta lepšie štruktúrované, resp. budú množiny týchto dát lineárne separovateľné. Použitím lineárnej separácie v inom priestore získame separačnú nadrovinu, ktorá oddeľuje v tomto priestore pôvodné množiny. Spätným premietnutím do pôvodného priestoru získame nelineárny klasifikátor. Tento postup je možné vidieť na Obrázku 10."Kernelový trik"sa používa častokrát v takých situáciách, kedy pracujeme najmä so skalárnym súčinom daných vektorov. Vďaka tomu nie je potrebné vedieť presné obrazy daných bodov v inom priestore, ale len ich skalárny súčin v tomto priestore, čo značne znižuje náročnoť úlohy. V tejto kapitole sme čerpali z [11],[13],[14],[15].



Obr. 10: Grafická interpretácia kernelového triku, lineárne neseparovateľné body su v priestore s vyššou dimenziou lineárne separovateľné. Obrázok je z [22]

2.1 Unitárny a Hilbertov priestor

Aby bolo možné vybudovať a pochopiť teóriu okolo kernelových funkcií je potrebné poznať niektoré teoretické poznatky z oblasti unitárneho a Hilbertovho priestoru. Nasledovné definície z [14] nám pomôžu v ďalších podkapitolách s budovaním matematickej teórie kernelových funkcii. V nasledujúcih definíciách budeme vždy predpokladať, že lineárny priestor je reálny.

Definícia 2.1: Unitárnym priestorom nazývame lineárny vektorový priestor Ω so skalárnym súčinom, t.j. s takým zobrazením $\langle \cdot, \cdot \rangle : \Omega \times \Omega \mapsto \mathbb{R}$, že platí:

1. $\langle x, x \rangle \geq 0$ pre všetky $x \in \Omega$; $\langle x, x \rangle = 0 \Leftrightarrow x = 0$,
2. $\langle x, y \rangle = \langle y, x \rangle$ pre všetky $x, y \in \Omega$,
3. $\langle x + y, z \rangle = \langle x, z \rangle + \langle y, z \rangle$ pre všetky $x, y \in \Omega$,
4. $\lambda \langle x, y \rangle = \langle \lambda x, y \rangle$, pre všetky $x, y \in \Omega; \lambda \in \mathbb{R}$.

Definícia 2.2: Lineárny vektorový priestor Ω sa nazýva normovaným, ak existuje také zobrazenie $|| \cdot || : \Omega \mapsto \mathbb{R}$, že platí:

1. $||x|| \geq 0$ pre všetky $x \in \Omega$; $||x|| = 0 \Leftrightarrow x = 0$,
2. $||x + y|| \leq ||x|| + ||y||$ pre všetky $x, y \in \Omega$,
3. $||\lambda x|| = \lambda ||x||$, pre všetky $x \in \Omega; \lambda \in \mathbb{R}$.

Ak je teda Ω unitárny priestor a pre ľubovoľné $x \in \Omega$ položíme $||x|| = \sqrt{\langle x, x \rangle}$, potom je Ω s normou $|| \cdot ||$ lineárnym normovaným priestorom.

Definícia 2.3: Unitárny priestor Ω nazývame Hilbertov priestor, ak platí, že je úplný, teda každá cauchyovská postupnosť $\{h_i \in \Theta\}_{i=1}^{\infty}$ taká že:

$$\lim_{n \to \infty} \sup_{m > n} ||h_n - h_m|| = 0,$$

konverguje k prvku $h \in \Omega$

Ak pre Hilbertov priestor platí, že existuje spočítateľná podmnožina $\widehat{\Omega} = \{h_i \in \Omega\}_{i=1}^{\infty}$ taká, že pre všetky $h \in \Omega$ a $\epsilon > 0$ existuje $h_i \in \widehat{\Omega}$ také že $||h_i - h|| < \epsilon$, tak je navyše separovateľný.

2.2 Kernelové funkcie

Autor [20] sformuloval nasledovnú všeobecnú definíciu kernelovej funkcie:

Definícia 2.4: Nech χ je priestor s mierou a nech $L^2(\chi \times \chi)$ je množina všetkých merateľých funkcii $\chi \times \chi$ s integrovateľným kvadrátom. Potom κ je kernelová funkcia ak $\kappa \in L^2(\chi \times \chi)$ a existuje separovateľný Hilbertov priestor Ω so skalárnym súčinom

$\langle \cdot, \cdot \rangle$a zobrazenie $\phi : \chi \mapsto \Omega$ také, že platí: $\forall x, x' \in \chi : \kappa(x, x') = \langle \phi(x), \phi(x') \rangle_\Omega$
Pozn.: Zobrazenie ϕ budeme nazývať charakteristické zobrazenie (z angl. feature map).

Pri aplikovaní kernelových funkcii v strojovom učení sa však pohybujeme na konečnorozmernom priestore, pretože dáta, na ktoré aplikujeme kernelove funckie sú konečné. Budeme teda v nasledujúcom zjednodušene predpokladať, že . Vďaka tejto vlastnosti uvedieme definíciu pre kernelové metódy na konečnom priestore:

Definícia 2.5: Konečná kernelová funkcia κ je kernelová funkcia na $\chi = \{x_1, x_2, ...x_n\}$. Takže existuje separovateľný Hilbertov priestor Ω a zobrazenie $\phi : \chi \mapsto \Omega$, také, že $\kappa(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle_\Omega, \forall i, j \in \{1, ..., n\}$. Každú konečnú kernelovú funkciu možno teda reprezentovať $n \times n$ maticou K takou, že $K_{ij} = K(x_i, x_j)$.

Kernelová matica je vlastne zovšeobecnením Gramovej matice, pretože ak $H = \mathbb{R}^m$,$\langle \cdot, \cdot \rangle$ je štandardný skalárny súčin, teda $\langle x, y \rangle = x^T y$ a zobrazenei ϕ je identita, tak K je Gramova matica. Keďže prvky matice K sú hodnoty kernelovej funkcie všetkých dvojíc vektorov z χ, tak pri aplikovaní kernelového triku potrebujeme práve tieto hodnoty, preto je poznanie kernelovej matice dostatočné na aplikovanie kernelového triku v metódach strojového učenia. Vďaka týmto vlastnostiam teda nepotrebujeme poznať zobrazenie ϕ, ani presný tvar $\kappa(x, y)$.

Je známe, že symetrická matica je Gramova matica práve vtedy, keď je kladne semidefinitná. Toto tvrdenie možno analogicky rozšíriť na kernelové matice, ako to uvedieme v nasledujúcej vete.
Pozn.: Táto veta je špeciálnym prípadom Mercerovej vety [20].

Nasledovná veta nám značne uľahčí spôsob, ako určiť či funkcia je konečnou kernelovou funkciou.

Veta 2.6: Matica je kernelovou maticou práve vtedy keď je symetrická a kladne semidefinitná .

Dôkaz: V implikáciu ($\Rightarrow$) stačí iba dokázať kladnú semidefinitnosť matice K, kedže vieme, že kernelová funkcia je definovaná ako skalárny súčin, ktorý je symetrický, tak musí platiť aj $K_{ij} = K_{ji}$, teda K je symetrická. Nech $z \in \mathbb{R}^m$ je ľubovoľný vektor,

potom platí:

$$\begin{aligned} z^T K z = \sum_{i,j} z_i K_{i,j} z_j &= \sum_{i,j} z_i \langle \phi(x_i), \phi(x_j) \rangle_\Omega z_j \\ &= \left\langle \sum_i z_i \phi(x_i), \sum_i z_j \phi(x_j) \right\rangle_\Omega \\ &= \left\| \sum_i z_i \phi(x_i) \right\|_\Omega^2 \geq 0, \end{aligned}$$

teda matica K je aj kladne semidefinitná.

Implikáciu ($\Leftarrow$) môžeme dokázať pomocou vlastností kladne semidefinitných matíc. Keďže platí, že K je symetrická a kladne semidefitniná, tak pre jej spektrálny rozklad $K = U\Lambda U^T$ platí, že aj matica D je kladne semidefinitná. Potom môžeme definovať zobrazenie $\chi : \chi \mapsto \mathbb{R}^n = \Omega$:

$$\phi(x_i) : x_i \mapsto (\sqrt{\lambda_i} u_{1i}, \sqrt{\lambda_i} u_{2i}, ..., \sqrt{\lambda_i} u_{ni}),$$

pričom Ω je v tomto prípade $\mathbb{R}^n$ so štandardným skalárnym súčinom, teda $\phi : \chi \mapsto \mathbb{R}^n$ a $\lambda_i = \Lambda_{ii}$ a u_i je $i-$ty stĺpec matice U. Potom platí:

$$\langle \phi(x_i), \phi(x_j) \rangle = \sum_{l=1}^{n} \lambda_l u_{li} u_{lj} = (V \Lambda V^T)_{ij} = K_{i,j} = K(x_i, x_j).$$

2.3 Vlastnosti konečných kernelových funkcii

Niektoré vlastnosti konečných kernelových funkcii sme získali z [11] a [13], pričom sme niektoré vlastnosti doplnili a ku všetkým týmto tvrdeniam uvádzame naše vlastné dôkazy.

Veta 2.7: Nech $\chi = \{x_1, x_2, ..., x_n\}$ a predpokladajme, že $\kappa_1, \kappa_2, ...\kappa_N \chi \times \chi \to \mathbb{R}$ sú konečné kernelové funkcie, $a, c \geq 0, f : \chi \to \mathbb{R}$ a funkcia $P : \mathbb{R} \to \mathbb{R}$ je polynóm s nezápornými koeficientami, teda $P(z) = \sum_{i=1}^{k} a_i z^i$, kde $a_i \geq 0, i = 1, ...m$. Potom všetky nasledovné funkcie sú konečné kernelové funkcie:

1. $\kappa(x, y) = \kappa_1(x, y) + \kappa_2(x, y)$,
2. $\kappa(x, y) = a\kappa_1(x, y)$,
3. $\kappa(x, y) = \kappa_1(x, y) \cdot \kappa_2(x, y)$,

4. $\kappa(x, y) = \kappa_1(x, y)^d$,

5. $\kappa(x, y) = \kappa_1(x, y) + c$,

6. $\kappa(x, y) = f(x)f(y)$,

7. $\kappa(x, y) = P(\kappa_1(x, y))$,

8. $\kappa(x, y) = f(x)\kappa_1(x, y)f(y)$,

9. $\kappa(x, y) = \lim_{N\to\infty} \sum_{i=0}^{N} \kappa_i(x, y)$,

10. $\kappa(x, y) = e^{\kappa_1(x,y)}$.

Dôkaz: Keďže χ je konečné, tak každá kernelová funkcia κ_1, κ_2 je jednoznačne určená kernelovou maticou. Preto môžeme v jednotlivých dôkazoch využiť Vetu 2.6. Označme $K_1, K_2, ...K_N$ kernelové matice určené kernelovými funkciami $\kappa_1, \kappa_2, ...\kappa_N$.

1. Matica, ktorá vznikne aplikovaním súčtu kernelových funkcii na sadu vektorov X je vlastne súčtom kernelových matíc pôvodných kernelových funkcii. Teda platí $K = K_1 + K_2$. Kedže K_1 a K_2 sú kladne semidefinitné, tak platí: $\forall z : z^T K_1 z \geq 0$ a $z^T K_2 z \geq 0$ a teda aj $z^T K_1 z + z^T K_2 z \geq 0$, resp. $z^T(K_1 + K_2)z$, teda aj matica $K = K_1 + K_2$ je kladne semidefinitná.

2. Podobne ako v prvej časti, kernelová matica funkcie $a\kappa_1$ pričom $a > 0$ môžeme označiť $K = aK_1$ a kedže $z^T K_1 z \geq 0$, tak aj $z^T aK_1 z = az^T K_1 z \geq 0$, teda aj matica K je kladne semidefinitná.

3. Keďže kernelové matice K_1 a K_2 sú kladne semidefinitné a symetrické, tak aj ich Hadamardov súčinom je kladne semidefinitná matica. Dôkaz tohto tvrdenia je v Prílohe 1. Súčin kernelových matíc je teda podľa Vety 2.6 kernelovou maticou.

4. Keďže zobrazenie $\kappa_1(x, y)^d$ je reprezentované maticou, ktorá vznikne d-násobným Hadamardovým súčinom kladne semidefinitnej matice, tak aj K_H^d je kladne semidefinitná. Dôkaz tohto tvrdenia je možné nájsť v Prílohe 1.

5. Zobrazenie $\kappa_1(x, y) + c$ je reprezentované maticou $K_1 + c\mathbf{1}\mathbf{1}^T$. Obe matice sú kladne semidefinitné a tak aj ich súčet je kladne semidefinitná matica.

6. Ak označíme $f(x_i) = f_i; i = 1, ..., n$, tak platí $\kappa(x_i, x_j) = f_i f_j$. Jednotlivé prvky K teda vyzerajú $K^{ij} = f_i f_j$, preto ak označíme $v = (f_1, f_2, ..., f_n)$, tak $K = vv^T$, čo znamená že je to kladne semidefinitná matica.

7. Keďže polynóm s nezápornými koeficientami s kernelovou funkciou v argumente je súčet mocnín kernelovej funkcie vynásobený nezápornými koeficientami, tak dôkaz vyplýva priamo z vlastností 1-5.

8. Podobne ako v 6, označíme $f(x_i) = f_i; i = 1, ..., n,$. Potom jednotlivé prvky K vyzerajú $K^{ij} = f_i K_1^{ij} f_j$, ak označíme $v = (f_1, f_2, ..., f_n)$, môžeme túto maticu zapísať ako $K = K_1 \odot vv^T$. O matici K vieme, že je kladne semidefinitná, pretože je to kernelová matica, taktiež matica vv^T je kladne semidefinitná matica hodnosti 1. Preto aj ich Hadamardov súčin je kladne semidefinitná matica, pričom dôkaz tohto tvrdenia je možné nájsť v Prílohe 1.

9. Kernelová matica kernelu $\kappa(x, y)$ bude v tomto prípade vyzerať ako nekonečná suma matíc, teda: $K = \lim_{N\to\infty} \sum_{i=0}^{N} K_i$. Keďže K_i sú kernelové matice, tak sú kladne semidefinitné. Kladne semidefinitné matice tvoria podľa [2] uzavretý kužeľ, tým pádom limita postupnosti kladne semidefinitných matíc je opäť kladne semidefinitná matica, z čoho vyplýva, že aj K je kladne semidefinitná.

10. Prvky matice K budú v tomto prípade vyzerať: $K^{ij} = e^{K_1^{ij}}$. V nasledujúcej formulácií budeme využívať označenie maticových "mocnín"v tvare Hadamardovho súčinu:

$$A_H^d = \underbrace{A \odot A \odot ... \odot A}_{d}.$$

Pomocou nekonečného rozvoja funkcie $f(z) = e^z$ môžeme teda zapísať tvar matice K: $K = \lim_{N\to\infty} \sum_{i=0}^{N} \frac{(K_1)_H^i}{i!}$. Podľa Dodatku 2 v Prílohe 1 môžeme označiť $K = e_H^{K_1}$, teda K je maticová exponenciála v zmysle Hadamardovho súčinu. Matica K_1 je kladne semidefinitná, rovnako jej mocniny v zmysle Hadamardovho súčinu, pričom dôkaz je možné nájsť v Prílohe 1. Teda jedná sa o nekonečnú sumu kladne semidefinitných matíc. Opäť, kladne semidefinitné matice tvoria podľa [2] uzavretý kužeľ, takže limita postupnosti kladne semidefinitných matíc je opäť kladne semidefinitná matica takže aj matica K je kladne semidefinitná.

2.4 Príklady kernelových funkcii

V tejto podkapitole predstavíme niektoré používané kernelové funkcie a u niektorých odvodíme aj presný tvar zobrazenia ϕ. Pri niektorých funkciách sme sa inšpirovali [11]. Pomocou vlastností uvedených vo Vete 2.7 môžeme jednotlivé nižšie uvdené kernelové funkcie transformovať na vytvárať nové kernelové funkcie. Týmito príkladmi kernelových funkcii sme sa inšpirovali z [21]. Poznamenávame, že v praxi sa využívajú aj všeobecné kernelové funkcie, ktoré vo všeobecnosti nie sú kladne semidefinitné.

2.4.1 Lineárna kernelová funkcia

Najjednoduchšou kernelovou funkciou je klasický skalárny súčin, v tomto prípade platí $\phi : \chi \mapsto \mathbb{R}^m$, teda Hilbertov priestor je v tomto prípade $\mathbb{R}^m$ s klasickým skalárnym súčinom. Niekedy sa zvykne používať súčet skalárneho súčinu a konštanty, teda $\kappa(x, y) = x^T y + c$. Z vlastnosti 4 Vety 2.7 vieme, že taktáto funkcia je kernelová, nazýva sa lineárna kernelová funkcia.

2.4.2 Polynomiálna kernelová funkcia

Umocnením lineárneho kernelu na $d > 0$ získame kernelovú funkciu $\kappa(x, y) = (x^T y + c)^d$, ktorá sa zvykne nazývať polynomiálna kernelová funkcia [13]. Opäť, vďaka vlastnosti 3 z Vety 2.7 vieme, že súčin kernelov je opäť kernelová funkcia, tym pádom aj umocňovanie. Pre prípad $d = 2, c = 0$ dokážeme relatívne ľahko nájsť zobrazenie ϕ, nech $x, y \in \mathbb{R}^m$, potom :

$$\begin{aligned}\kappa(x, y) = (x^T y)^2 &= \Big(\sum_{i=1}^{n} x_i y_i\Big)\Big(\sum_{j=1}^{n} x_j y_j\Big) \\ &= \sum_{i,j=1}^{n} x_i y_i x_j y_j \\ &= \sum_{i,j=1}^{n} (x_i x_j)(y_i y_j),\end{aligned}$$

z toho je možné vidieť, že $\phi : \chi \mapsto \mathbb{R}^{n^2}$. Napríklad zobrazenie pre n=2 vyzerá:

$$(x_1, x_2) \mapsto (x_1 x_1, x_1 x_2, x_2 x_1, x_2 x_2).$$

Pre $d = 3$ použijeme podobný postup:

$$\begin{aligned}\kappa(x,y) = (x^T y)^3 =& \Big(\sum_{i=1}^{n} x_i y_i\Big)\Big(\sum_{j=1}^{n} x_j y_j\Big)\Big(\sum_{k=1}^{n} x_k y_k\Big) \\ &= \sum_{i,j,k=1}^{n} x_i y_i x_j y_j x_k y_k \\ &= \sum_{i,j,k=1}^{n} (x_i x_j x_k)(y_i y_j y_k),\end{aligned}$$

opäť na príklade pre $m = 2$ bude zobrazenie ϕ vyzerať:

$$(x_1, x_2) \mapsto (x_1x_1x_1, x_1x_1x_2, x_1x_2x_1, x_1x_2x_2, x_2x_1x_1, x_2x_1x_2, x_2x_2x_1, x_2x_2x_2).$$

Pre polynomiálnu kernelovú funkciu je zobrazenie $\phi : \chi \mapsto \mathbb{R}^{n^d}$, pričom Hilbertov priestor je $\mathbb{R}^{n^d}$ s klasickým skalárnym súčinom. V príklade vyššie je teda $\phi : \chi \mapsto \mathbb{R}^8$. Aj tu je možné vidieť výhody používania kernelovej funkcie, miesto výpočtov v priestore $\mathbb{R}^8$ pracujeme stále iba so skalárnym súčinom.

v prípade nenulovej konštanty pre $d = 2$ platí:

$$\begin{aligned}\kappa(x,y) =& (x^T y + c)^2 \\ &= \sum_{i,j=1}^{n} (x_i x_j)(y_i y_j) + \sum_{i=1}^{n} (\sqrt{c}x_i)(\sqrt{c}y_i) + c^2,\end{aligned}$$

takže v prípade $n = 2$ dokážeme ľahko nájsť aj zobrazeni:

$$\phi : (x_1, x_2) \mapsto (x_1x_1, x_1x_2, x_2x_1, x_2x_2, \sqrt{c}x_1, \sqrt{c}x_2, c).$$

V takomto prípade zobrazuje ϕ z χ do priestoru $\mathbb{R}^{\binom{n+d}{d}}$.

2.4.3 RBF kernelová funkcia

Jedna z najpoužívaješích kernelových funkcii sa nazýva RBF (Radial basis function) a vyzerá nasledovne:

$$k(x,y) = e^{-\frac{||x-y||_2^2}{2\sigma^2}},$$

ako je možné vidieť, funkcia nepracuje so skalárnym súčinom, ale s euklidovskou vzdialenosťou daných vektorov.Tým istým spôsobom zachytáva mieru podobnosti dvoch vektorov. Jediným voľným parametrom v tejto funkcii je σ, niekedy je možné nájsť

v literatúre jej ekvivalentný tvar, kde je $\frac{1}{2\sigma^2}$ nahradené γ. Táto kernelová funkcia ma niekoľko modifikácii, ak sa používa v exponente iba prvá mocnina euklidovskej vzdialenosti, jedná sa o exponenciálnu kernelovú funkciu. Ak navyše použijeme v menovateli miesto σ^2 iba σ, kernelová funkcia sa nazýva Laplaceova. Laplaceova kernelová funkcia je menej závislá od parametra σ.

Pre dôkaz kladnej semidefinitnosti RBF kernelu je nutné rozpísať výraz v exponente:

$$e^{-\frac{||x-y||_2^2}{2\sigma^2}} = e^{-\frac{(x-y)^T(x-y)}{2\sigma^2}} = e^{-\frac{(x^Tx)-2(x^Ty)+(y^Ty)}{2\sigma^2}}.$$

Posledný výraz je možné prepísať do tvaru:

$$e^{-\frac{(x^Tx)-2(x^Ty)+(y^Ty)}{2\sigma^2}} = e^{-\frac{(||x||_2)}{2\sigma^2}} e^{\frac{x^Ty}{\sigma^2}} e^{-\frac{(||y||_2)}{2\sigma^2}}.$$

Vďaka vlastnosti 10 z Vety 2.7 vieme, že $\kappa = 1e^{\frac{x^Ty}{\sigma^2}}$ je kernelová funkcia, pretože aj $\kappa_{lin} = \frac{x^Ty}{\sigma^2}$ je kernelová funkcia, konkrétne lineárna kernelová funckia s kernelovou maticou K_{lin}. Tým pádom, ak označíme $f(z) = e^{-\frac{(||z||_2)}{2\sigma^2}}$, môžeme využiť vlastnosť 8 Vety 2.7, z čoho priamo vyplýva, že $e^{-\frac{||x-y||_2^2}{2\sigma^2}}$ je kernelová funkcia.

Pre RBF funkciu vieme ľahko nájsť aj tvar kernelovej matice, keďže kernelová matica funkcie $\kappa_1 = e^{\frac{x^Ty}{\sigma^2}}$ je podľa vlastnosti 10 Vety 2.7 maticová exponenciála v zmysle Hadamardovho súčinu teda $e_H^{K_{lin}}$. Tvar výslednej kernelovej matice priamo vyplýva z vlastnosti 8 Vety 2.7.

2.4.4 Sigmoid kernelová funkcia

Ďalšou často používanou kernelovou funkciou je sigmodiová funkcia, ktorá má tvar:

$$k(x, y) = \tanh(\alpha x^T y + c).$$

Ako je možné vidieť, má dva voľné parametre α a c, čím ju môžeme prispôsobovať na konkrétne aplikácie. Táto funkcia sa pôvodne používala iba v neurónových sieťach, no postupne sa začala používať aj v niektorých kernelových metódach. Zaujímavosťou je, že ak sa podľa [21] použije v metóde oporného bodu, ktorou sa budeme v ďalších kapitolách zaoberať, stane sa táto úloha ekvivalentná dvojvrstvovej neurónovej sieti. Sigmoid kernelová funkcia je príkladom kernelovej funkcie, ktorej kernelová matica nemusí byť vždy kladne semidefinitná. Avšak vhodným zvolením parametrov α a c dokážeme docieliť, aby za nejakých okolností túto vlastnosť mala. Keďže je známe, že

diganonálne dominantná symetrická matica je kladne semidefinitná, stačí zvoliť také parametre α a c, aby výsledná kernelová matica bola diagonálne dominantná.

2.4.5 Zlomková kvadratická kernelová funkcia

Podľa [21] by pri použití RBF kernelovej funkcie mohlo niekedy dôjsť k problémom s výpočtovou náročnosťou, v takom prípade sa používa zlomková kvadratická kernelová funkcia, ktorá má tvar:

$$k(x, y) = 1 - \frac{||x - y||^2}{||x - y||^2 + c}$$

a má práve jeden voľný parameter a to c. Kernelová matica tejto kernelovej funkcie taktiež nemusí byť nutne kladne semidefinitná. Podobne ako pri sigmoid kernelovej funkcii, aj tu dokážeme vhodným zvolením parametra c docieliť, aby kernelová matica priamo určená zlomkovou kvadratickou kernelovou funkciou bola kladne semidefinitná a to zvolením takej hodnoty c, aby kernelová matica bola diagonálne dominantná.

Kernelových funkcii je omnoho viac, výber kernelovej funkcie sa viaže častokrát ku konkrétnemu problému a dostupným dátam. Pomocou voľných parametrov a rôznych operácii podľa Vety 2.7, môžeme jednotlivé kernelové funkcie čiastočne upravovať pre naše potreby.

2.5 Použitie kernelových funkcii

Pomocou kernelových funkcii dokážeme jednoducho vypočítať nie len skalárny súčin obrazov daných vektorov v inom priestore, ale aj ich vzdialenosť v tomto priestore, resp. veľkosť, pretože tieto vzťahy vychádzajú zo skalárneho súčinu. Vďaka týmto vlastnostiam dokážeme napríklad kernelové funkcie využiť pri normalizácii dát, resp. pri centrovaní dát. Viac informácii o týchto aplikáciách je možné nájsť v [8].

Uplatnenie kernelových funkcii je v takých metódach, kde potrebujeme poznať iba skalárny súčin vektorov. Najčastejšou a najvýznamnejšou kernelovou aplikáciou je metóda oporného bodu. Táto metóda našla, ako bolo možno vidieť v predošlej kapitole, svoje uplatnenie v klasifikácii alebo regresii. V ďalších kapitolách sa budeme venovať aplikovaniu kernelového triku v probléme SVM.

3 Kernelový trik v metóde oporného bodu

V prvej kapitole sme uviedli a odvodili primárne a duálne úlohy metódy SVM. V predošlej kapitole sme uviedli, že zobrazením bodov v inom priestote môžeme získať lepšie štrukturované dáta, resp. môžeme pracovať s metódami, s ktorými to v pôvodnom priestore nebolo možné. Taktiež sme ukázali, že nie vždy je nutné vždy poznať presné obrazy bodov v novom priestore, ale len ich skalárne súčiny, vďaka čomu sú úlohy výpočtovo jednoduchšie. V tejto kapitole spojíme poznatky z oboch kapitol a tak budeme používať metódu SVM nie na pôvodné dáta, ale len na ich obrazy v inom priestore, pričom jediné čo budeme potrebovať sú hodnoty ich skalárnych súčinov, teda hodnoty kernelovej funkcie.

3.1 Alternatívne formulácie problému SVM

Ak zobrazíme pôvodné $x_i, x_2, ...x_N$ z množiny X a $y_i, y_2, ...y_M$ z množiny Y a označíme $Z = X \cup Y$, tak pomocou charakteristického zobrazenia $\phi : Z \mapsto \Omega$ získame obrazy týchto bodov v priestore Ω, teda $\phi(x_1), \phi(x_2), ...\phi(x_N)$, resp. $\phi(y_1), \phi(y_2), ...\phi(y_M)$.

Ak teda v klasickej SVM metóde bez „slackov“ (1.8) budeme pracovať s obrazmi bodov v inom priestore, úloha bude vyzerať nasledovne:

$$\begin{aligned} \min_{a,b} \quad & ||a||_2 \\ & a^T\phi(x_i) - b \geq 1 \ , i = 1, ...N, \\ & a^T\phi(y_i) - b \leq -1 \ , i = 1, ...M. \end{aligned} \tag{3.1}$$

V tomto prípade budeme zjednodušene označovať $a^T\phi(x_i)$, resp. $a^T\phi(y_i)$ skalárny súčin na priestore Ω. Geometrická interpretácia vyššie uvedeného problému je rovnaká ako v prípade pôvodnej úlohy (1.8). Teda opäť hľadáme nadrovinu, ktorá čo najlepšie oddeľuje obrazy bodov z pôvodných množín. K tejto úlohe vieme podobne ako v 1. kapitole odvodiť duálnu úlohu, ktorú možno v takom prípade inteprtovať ako hľadanie vzdialenosti konvexných obalov množín obrazov bodov v novom priestore.

Analogicky možno zovšeobecniť úlohu (1.27) so slackovými premennými. V tomto prípade uvažujeme ako možné riešenie aj takú nadrovinu, ktorá povoľuje mierne odchýlky, teda nie všetky obrazy bodov pôvodných množín musia byť v dvoch rozdielnych polp-

riestoroch, ktoré určuje hľadaná nadrovina. Úloha teda získa tvar:

$$\begin{aligned} \min_{a,b} \quad & ||a||_2 + C\sum_{i=1}^{N} u_i + C\sum_{i=1}^{M} v_i \\ & a^T\phi(x_i) - b \geq 1 - u_i \ , i = 1, ...N, \\ & a^T\phi(y_i) - b \leq -1 + v_i \ , i = 1, ...M. \\ & u, v \succeq 0. \end{aligned} \tag{3.2}$$

Rovnako aj v tomto prípade vieme odvodiť duálnu úlohu rovnakým postupom ako v 1. kapitole, geometrická interpretácia vyššie spomenutého problému je teda hľadanie vzdialenosti redukovaných konvexných obalov jednotlivých množín obrazov bodov.

Obe vyššie spomenuté úlohy ponúkajú zrozumiteľné geometrické interpretácie, avšak aby sme mohli skutočne využiť kernelový trik, teda nahradenie skalárneho súčinu kernelovou funkciou a dané úlohy riešiť, je potrebné účelovú funkciu oboch problémov mierne upraviť. Postačuje, aby sme v úlohe (3.1) a (3.2) nahradili v účelovej funkcii člen $||a||_2$ členom $\frac{||a||_2^2}{2} = \frac{1}{2}a^Ta$, čo ukážeme odvodením duálnej úlohy v ďalšej podkapitole. Úpravou účelovej funkcie sme sa čiastočne inšpirovali [17]. Úloha (3.1) teda bude vyzerať:

$$\begin{aligned} \min_{a,b} \quad & \frac{1}{2}a^Ta \\ & a^T\phi(x_i) - b \geq 1 \ , i = 1, ...N, \\ & a^T\phi(y_i) - b \leq -1 \ , i = 1, ...M. \end{aligned} \tag{3.3}$$

A úloha (3.2) získa tvar:

$$\begin{aligned} \min_{a,b} \quad & \frac{1}{2}a^Ta + C\sum_{i=1}^{N} u_i + C\sum_{i=1}^{M} v_i \\ & a^T\phi(x_i) - b \geq 1 - u_i \ , i = 1, ...N, \\ & a^T\phi(y_i) - b \leq -1 + v_i \ , i = 1, ...M. \\ & u, v \succeq 0. \end{aligned} \tag{3.4}$$

3.2 Duálna úloha alternatívnych formulácii SVM

V nasledujúcej časti odvodíme duálne úlohy pre (3.3) a (3.4), čím získame formulácie duálnych úloh, kde bude možné využiť "kernelový trik", teda nahradíme skalárne súčiny

kernelovou funkciou. Lagrangeova funkcia úlohy (3.3) vyzerá:

$$L(a,b;\alpha,\beta) = \frac{a^T a}{2} - a^T\Big(\sum_{i=1}^{M}\beta_i\phi(y_i) - \sum_{i=1}^{N}\alpha_i\phi(x_i)\Big) + \sum_{i=1}^{N}\alpha_i + \sum_{i=1}^{M}\beta_i \\ - b\sum_{i=1}^{N}\alpha_i + b\sum_{i=1}^{M}\beta_i. \quad (3.5)$$

Pre formuláciu duálnej funkcie potrebujeme nájsť infimum $L(a,b;\alpha,\beta)$ cez premenné a,b. Keďže je funkcia $L(a,b;\alpha,\beta)$ v primárnych premenných separovateľná, tak platí:

$$L(a,b;\alpha,\beta) = L_1(a,\alpha,\beta) + L_2(b,\alpha,\beta), \quad (3.6)$$

kde

$$L_1(a,\alpha,\beta) = \frac{a^T a}{2} - a^T\Big(\sum_{i=1}^{M}\beta_i\phi(y_i) - \sum_{i=1}^{N}\alpha_i\phi(x_i)\Big) + \sum_{i=1}^{N}\alpha_i + \sum_{i=1}^{M}\beta_i, \quad (3.7)$$

$$L_2(b,\alpha,\beta) = b\Big(\sum_{i=1}^{M}\beta_i - \sum_{i=1}^{N}\alpha_i\Big). \quad (3.8)$$

Vďaka tomu je infimum $L(a,b;\alpha,\beta)$ súčet infím jednotlivých funkcii L_1, L_2. Funkcia $L_2(b,\alpha,\beta)$ je lineárna v b, preto môžeme využiť (1.9). Pre $L_1(a,\alpha,\beta)$ vieme nájsť jednoducho infimum vďaka (1.12). V takom prípade, ak označíme

$$\psi = \sum_{i=1}^{M}\beta_i\phi(y_i) - \sum_{i=1}^{N}\alpha_i\phi(x_i),$$

tak vo vektorovom zápise platí:

$$\inf_a \quad L_1(a,\alpha,\beta) = -\frac{1}{2}\psi^T\psi + (\alpha+\beta)^T\mathbf{1} \quad (3.9)$$

Našli sme teda infimum funkcii L_1, L_2, a preto:

$$\inf_{a,b} \quad L(a,b;\alpha,\beta) = \begin{cases} -\frac{1}{2}\psi^T\psi + (\alpha+\beta)^T\mathbf{1}; & \sum_{i=1}^{M}\beta_i = \sum_{i=1}^{N}\alpha_i, \\ \\ -\infty; & inak. \end{cases} \quad (3.10)$$

Spätnou substitúciou získame duálnu úlohu (3.11). Ak pôvodné vektorové výrazy rozpíšeme, duálnu úlohu môžeme zapísať ako:

$$\max_{\alpha,\beta} \quad -\frac{1}{2}\psi^T\psi + \sum_{i=1}^{N}\alpha_i + \sum_{i=1}^{M}\beta_i \\ \sum_{i=1}^{M}\beta_i = \sum_{i=1}^{N}\alpha_i, \quad (3.11) \\ \alpha,\beta \succeq 0.$$

V prípade (3.4) so zmenenou účelovou funkciou je v Lagrangeovej funkcii (3.5) ešte jeden člen a to $u^T(C\mathbf{1}-\alpha)+v^T(C\mathbf{1}-\beta)$. Lagrangeova funkcia $L(a,b,u,v;\alpha,\beta)$ je však aj v tomto prípade separovateľná v každej jednej primárnej premennej, teda platí:

$$L(a,b,u,v;\alpha,\beta) \;=\; L_1(a,\alpha,\beta) \;+\; L_2(b,\alpha,\beta) \;+\; L_3(u,\alpha,\beta) \;+\; L_4(v,\alpha,\beta). \quad (3.12)$$

Funkcie $L_1(a,\alpha,\beta)$ a $L_2(b,\alpha,\beta)$ vyzerajú rovnako ako v (3.7) a (3.8), pre funkcie $L_3(u,\alpha,\beta)$ a $L_4(v,\alpha,\beta)$ platí:

$$L_3(u,\alpha,\beta) = u^T(C\mathbf{1}-\alpha),$$

$$L_4(v,\alpha,\beta) = v^T(C\mathbf{1}-\beta).$$

Keďže z (3.4) vieme, že $u,v \succeq 0$, tak pri hľadaní infima funkcii $L_3(u,\alpha,\beta)$ a $L_4(v,\alpha,\beta)$ použijeme (1.10). Tým sa však nezmení tvar účelovej funkcie duálnej úlohy, ale získame nové ohraničenia na α,β, pretože vektory $C\mathbf{1}-\alpha$ a $C\mathbf{1}-\beta$ musia mať všetky zložky nezáporné podľa (1.10). Duálna úloha teda v takomto prípade vyzerá:

$$\begin{aligned} \max_{\alpha,\beta} \quad & -\frac{1}{2}\psi^T\psi + \sum_{i=1}^{N}\alpha_i + \sum_{i=1}^{M}\beta_i \\ & \sum_{i=1}^{M}\beta_i = \sum_{i=1}^{N}\alpha_i, \\ & 0 \preceq \alpha,\beta \preceq C\mathbf{1}. \end{aligned} \quad (3.13)$$

3.3 Kompaktný zápis úloh SVM

Úlohy (3.11) a (3.13) vieme zapísať kompaktnejšie pri zvolení nových premenných. Ak označíme vektor duálnych premenných $\gamma = (\alpha, -\beta)$ a vektor z_i ako:

$$z_i = \begin{cases} x_i; & i = 1,...,N, \\ \\ y_i; & i = N+1,...,N+M. \end{cases}$$

dokážeme naformulovať nový tvar duálnej úlohy oboch pôvodných primárnych problémov. V takom prípade platí:

$$\psi = \sum_{i=1}^{M}\beta_i\phi(y_i) - \sum_{i=1}^{N}\alpha_i\phi(x_i) = \sum_{i=1}^{N+M}\gamma_i\phi(z_i).$$

Pomocou matice:

$$\begin{bmatrix} I & 0 \\ 0 & -I \end{bmatrix},$$

možno problém (3.11) naformulovať v tvare:

$$\begin{aligned} \max_{\alpha,\beta} \quad & -\frac{1}{2}\Big(\sum_{i=1}^{N+M} \gamma_i \phi(z_i)\Big)^T \Big(\sum_{i=1}^{N+M} \gamma_i \phi(z_i)\Big) + \mathbf{1}^T J\gamma \\ & \mathbf{1}^T\gamma = 0, \\ & J\gamma \succeq 0. \end{aligned} \tag{3.14}$$

Skalárny súčin v účelovej funkcii môžeme po roznásobení zapísať v tvare súčtu a tak kompaktný tvar (3.11):

$$\begin{aligned} \max_{\alpha,\beta} \quad & -\frac{1}{2}\Big(\sum_{i=1,j}^{N+M} \gamma_i \phi(z_i)^T \phi(z_j) \gamma_j\Big) + \mathbf{1}^T J\gamma \\ & \mathbf{1}^T\gamma = 0, \\ & J\gamma \succeq 0. \end{aligned} \tag{3.15}$$

V probléme (3.13) je navyše len horné ohraničenie na α, β, preto analogicky vyzerá kompaktný tvar:

$$\begin{aligned} \max_{\alpha,\beta} \quad & -\frac{1}{2}\Big(\sum_{i=1,j}^{N+M} \gamma_i \phi(z_i)^T \phi(z_j) \gamma_j\Big) + \mathbf{1}^T J\gamma \\ & \mathbf{1}^T\gamma = 0, \\ & 0 \preceq J\gamma \preceq C\mathbf{1}. \end{aligned} \tag{3.16}$$

3.4 Kernelový trik v úlohách SVM

V úlohách (3.15) aj (3.16) dokážeme využiť kernelový trik, pretože vo formulácií úlohy nevystupujú samostatne žiadne obrazy bodov, iba ich skalárne súčiny, teda môžeme nahradiť tieto skalárnych súčiny kernelovou funkciou $\kappa(x, x')$, čo vedie k formulácii:

$$\begin{aligned} \max_{\gamma} \quad & -\frac{1}{2}\Big(\sum_{i=1,j}^{N+M} \gamma_i \kappa(z_i, z_j) \gamma_j\Big) + \mathbf{1}^T J\gamma \\ & \mathbf{1}^T\gamma = 0, \\ & J\gamma \succeq 0, \end{aligned} \tag{3.17}$$

resp. v prípade (3.16) analogicky k:

$$\begin{aligned} \max_{\gamma} \quad & -\frac{1}{2}\Big(\sum_{i=1,j}^{N+M} \gamma_i \kappa(z_i, z_j)\gamma_j\Big) + \mathbf{1}^T J\gamma \\ & \mathbf{1}^T\gamma = 0, \\ & 0 \preceq J\gamma \preceq C\mathbf{1}. \end{aligned} \tag{3.18}$$

Navyše, účelovú funkciu takto dokážeme celú zapísať v maticovom zápise s využitím kernelovej matice, pričom ju môžeme naformulovať ako minimalizačnú úlohu:

$$\begin{aligned} \min_{\gamma} \quad & \frac{1}{2}\gamma^T K\gamma - \mathbf{1}^T J\gamma \\ & \mathbf{1}^T\gamma = 0, \\ & J\gamma \succeq 0. \end{aligned} \tag{3.19}$$

Analogicky, v prípade (3.16):

$$\begin{aligned} \min_{\gamma} \quad & \frac{1}{2}\gamma^T K\gamma - \mathbf{1}^T J\gamma \\ & \mathbf{1}^T\gamma = 0, \\ & 0 \preceq J\gamma \preceq C\mathbf{1}. \end{aligned} \tag{3.20}$$

Matica K je kernelová matica priamo určená kernelovou funkciou $\kappa(x, x')$. Vo formuláciách problémov SVM teda nepotrebujeme poznať presný tvar kernelovej funkcie, postačujúca je znalosť kernelovej matice. V kapitole 2 sme uviedli, že kernelová matica je kladne semidefinitná, preto problémy popísané vyššie sú riešiteľné konvexné úlohy. V týchto úlohách však úplne stačí, ak je matica K kladne semidefinitná na priestore určenom ohraničenami, v našom prípade na $[\mathbf{1}]^{\perp}$.

4 Aplikácia SVM

V tejto kapitole predstavíme dostupné algoritmy, ktoré sa používajú na riešenie úloh SVM a SVR. Pre metódu SVM uvedieme aj výsledky aplikovanie metódy SVM v oblasti spracovania prirodzeného jazyka, porovnáme niekoľko kernelých funkcii vzľadom na rozličné parametre. V tejto časti sme sa prevažne inšpirovali [18],[13], [16], [19].

4.1 Používané algoritmy

Algoritmy od vzniku SVM sa postupne menili a vyvýjali. Jedným z prvých používaných algoritmov bola Modifikovaná gradientová projekcia [13]. Týmto algoritmom bol vyriešený prvý experimentálny SVM problém. Hlavnou myšlienkou je iteratívne hľadanie vhodných smerov zo štartovacieho bodu. Táto metóda však bola veľmi náročná na výpočtový čas a pamäť, postupne bola vylepšovaná. Vtedajšie dostupné solvre všaj vyžadovali znalosť celej kernelovej matice, ktorá však nebola nutná. Preto prišla onedlho metóda dekompozície hlavného SVM problému na niekoľko menších podproblémov kvadratického programovania. Táto metóda obchádzala nutnosť znalosti celej kernelovej matice a viedla tak k efektívnejším výpočtom. Po čase bola metóda dekompozície upravená natoľko, že sa jednotlivé riešenia optimalizačných podproblémov hľadali v smere, ktorý obsahoval iba dve nenulové zložky. Tento algoritmus sa nazýva SMO [13]. Ohraničenia SVM navyše spôsobia, že sa jednotlivé podporblémy stanú jednorozmernými. Tým sa náročnosť hľadania newtonovského kroku rapídne zmenšila a výpočtový čas sa skrátil. Väčšina súčansých moderných solverov pracuje práve s SMO. Dostupné informácie o všetkých vyššie popísaných algoritmoch je možné nájsť v [13].

V súčasnosti, okrem stáleho zefektívňovania existujúcich algoritmov sa stále pracuje na nových. Niektoré využívajú práve geometrický prístup k problému SVM, ktorú sme popísali v druhej kapitole. Sú aj také, ktoré využívajú napr. tú vlastnosť, že geometrická interpretácia duálneho problému je hľadanie redukovaných konvexných obalov. Príkladom tejto metódy je algoritmus popísaný v [12], ktorý využíva vlastnosti redukovaných konvexných obalov pre nájdenie optimálneho riešenia SVM.

4.2 Spracovanie prirodzeného jazyka a strojové učenie

Spracovanie prirodzeného jazyka je medziodborová vedecká disciplína, ktorá stojí na rozhraní matematiky, informatiky, jazykových vied ale aj psychológie. Je to jedna z najväčších oblastí aplikácie umelej inteligencie a metód strojového učenia. Hlavným cieľom je skúmanie komunikácie pomocou výpočtovej techniky, či už komunikácie ľudí medzi sebou alebo komunikácie počítača s človekom pomocou bežného ľudského jazyka. Spracovanie prirodzeného jazyka v zmysle strojového učenia sa najviac využíva na analýzu bežných ľudsky čitateľných textov a hovorených slov. Medzi najčastejšie aplikácie spracovania prirodzeného jazyka, ktorý sa niekedy označuje skratkou NLP (Natural language processing) patrí rospoznávanie hlasu, detekcia autorstva, rospoznávanie písma a klasifikácia textov.

V nasledujúcej časti sa budeme venovať práve detekcii autorstva. Detekcia autorstva ako taká existovala omnoho skôr ako existoval odbor zaoberajúci sa spracovaním prirodzeného jazyka. Problém so správnym určením autorov textu sa prejavoval už v stredoveku a o nejaký čas s týmto problémom bojovali aj v súdnictve. V polovici 20. storočia vznikol samostatný jazykovedný odbor a to forenzná lingvistika. Jej hlavným cieľom bolo na základe analýzy obsahu a skladby viet určit či autor iného textu je autorom skúmaného textu. Z forenznej lingvistiky sa mnohé princípy prebrali a tak detekcia autorstva bola platnou aplikáciou spracovania prirodzeného jazyka, kde úlohu forenzného lingvistika plní počítač.

V spracovaní prirodzeného jazyka sa pre porozumenie vety počítačom zvyčajne používajú dva prístupy, tým prvým je analýza kľúčových slov, druhý je syntaktickosémantická analýza textu. V detekcií autorstva sa častokárt využíva ten druhý z menovaných. Tento prístup sa osvedčil ako správny najmä preto, že analyzuje text hĺbkovo, po stránke obsahu ale aj skladby, rovnako ako to robí lingvista pri analýze textu, ktorého autorstvo je spochybnené.

4.3 Príprava dát pre detekciu autorstva

V tejto časti využijeme poznatky z SVM a spracovania prirodzeného jazyka v aplikácii detekcie autorstva, ktorú sme čiastočne popísali v predošlej podkapitole. SVM bola donedávna najpoužívanejšs metóds v strojovom učení pre detekovanie autorstvva textu,

dnes sa do popredia často dostávajú neurónové siete s čiastočne lepšou úspešnosťou.

V súčasných riešeniach dostupný softvér pre detekciu autorstva využíva veľa rozličných znakov, ktoré sú však prispôsobené na anglický jazyk. V slovanských jazykoch sú však mnohé prvky, ktoré je možné sledovať narozdiel od anglického jazyka a časť znakov, ktoré sa pre slovenčinu využiť nedajú. Ako trénovacie dáta sme preto vybrali slovenské texty a to konkrétne z úvodu a záveru bakalárskych prác študentov odboru Ekonomická a finančná matematika a diplomových prác študentov odboru Ekonomicko-finančná matematika a modelovanie. Rozhodli sme sa vyberať práve trénovacie texty z úvodu a záveru, pretože obsahujú minimum citovaných častí a zachytávajú najviac autenticity autora. Kedže znaky, ktoré sme sa rozhodli sledovať sú závislé od správneho použitia interpunkcie, podmienkou na trénovacie texty bola používať texty, kde autor bude interpunkciu dodržiavať, preto sme zvolili záverečnú prácu miesto bežných autorových textov dostupných na sociálnych sieťach. Priemerná dĺžka jedného trénovacieho textu bola približne 200 slov, pretože sme každý text v úvode resp. závere rozdelili na niekoľko osobitných častí.

V tejto práci sme sa rozhodli sledovať 6 znakov. Pri ich výbere sme sa inšpirovali napríklad [7] a [18]. Z každého trénovacieho textu sme pomocou nami napísaného programu postupne získavali vektor charakteristík. Tento program je uvedený v Prílohe 2. Jednotlivé zložky vektora charakteristík boli tvorené číselnými údajmi reprezentujúcimi:

1. **Priemerná dĺžka viet**

   Týmto základným znakom sme sledovali, koľko slov využíva priemerne autor textu vo vete. Tým sme chceli rozlíšiť autorov, ktorý používajú skôr krátke vety od tých, ktorí vety častejšie rozvíjajú do súvetí. Túto charakterisitku sme určovali podľa počtu bodiek v texte, každá bodka reprezentovala začiatok novej vety. V tomto prípade sme texty očisťovali od bodiek, ktoré slúžili na iný účel.

2. **Štandardná odchýlka dĺžky viet**

   V predošlom znaku sme získali informáciu o priemernej dĺžke vety, avšak nevedeli sme zhodnotiť či jednoduché vety s dlhými vetami strieda alebo píše vety približne rovnako dlhé. Týmto znakom dokážeme zistiť aký veľký rozptyl bol v dĺžke viet.

3. **Priemerný počet čiarok**

   Pokiaľ autor sa uchyľuje k písaniu dlhých viet (na základe znaku č.1), tak tento znak napovie viac či autor využíva spájanie viet pomocou spojok alebo uprednostňuje využívanie čiarok. Taktiež nám viac napovedá o synaktickej rovine viet a teda či autor využíva skôr priraďovacie súvetia alebo podraďovacie.

4. **Priemerný počet slov Hapax legomenon**

   Hapax legomenon je slovo, ktoré je využité v texte iba jedenkrát. Sú to istým spôsobom unikántne slová. V tomto znaku počítame priemerný počet týchto slov, ktoré sa v textoch nachádzajú. Táto charakteristika autora taktiež hovorí aj o jeho vzťahu k využívaniu synoným, pretože pokiaľ sa autor venuje nejakej téme, využívaniu toho istého slova sa vyhne iba využitím vhodných synoným.

5. **Priemerný počet stop slov**

   Stop slová su slová s čisto syntaktickým významom v texte a väčšinou nemajú žiaden význam, keď sú izolovane použité mimo kontext. Touto charakteristikou sledujeme ako často autor využíva tieto slová. V tomto znaku uchovávame informáciu o podiele počtu stop slov voči všetkým slovám. Náš program držal pole najčastejších slovenských stop slov, ktorých výskyt v texte kontroloval.

6. **Bohatosť slovnej zásoby**

   V tomto znaku držíme bohatosť autorovej slovnej zásoby označenej K, ktorú sme vypočítatali podľa vzorca, ktorý navrhol v roku 1944 britský štatisitk Udny Yule [18]

   $$K = \frac{10^4(\sum i^2 V_i - N)}{N^2},$$

   pričom N označuje počet slov vo vete a V_i označuje počet slov s frekvenciou i.

Vyššie popísané znaky je možné rozdeliť do dvoch skupín. Tou prvou skupinou znakov sme sledovali najmä syntax, konkrétne priemerný počet čiarok, priemernú dĺžku viet a jej štandarndú odchýlku, druhou skupinou znakov sme pozorovali najmä sémantiku, konkrétne počet slov, ktoré izolovane nemajú žiaden význam, bohatosť slovnej zásoby a počet slov využitých práve raz. Väčšinu týchto znakov by sa dali aplikovať na detekovanie autorstva aj v inom jazyku. V súčasnosti prichádza do popredia softvér, ktorý väčšiu pozornosť sústreďuje na znaky typické pre daný jazyk. Pre slovenčinu by

to mohli byť napríklad ďalšie syntaktické znaky, ako napríklad slovosled, používanie priamych a nepriamych predmetov. Takéto programy majú častokrát väčšiu úspešnosť. V tejto práci sa však venujeme väčšine znakov, ktoré sú aplikovateľné aj na niektoré iné jazykov, s výnimkou "stop slov", ktoré sa vzťahujú na slovenské slová. Vďaka pravidlám slovenčiny však dokážeme lepšie interpretovať niektoré znaky ako naprílad priemerný počet čiarok, pretože častokrát hovoria, aké súvetia autor použil, čo sa nedá použiť ako kritérium v niektorých jazykoch, resp. toto kritérium nemá až taký dôležitý význam.

4.4 Implementácia detekcie autorstva v Pythone

Znaky, ktoré sme popísali v predošlej kapitole sme vyhľadávali v texte pomocou nami napísaného programu v programovacom jazyku Python 3, na riešenie úlohy SVM sme použili knižnicu *sklearn* a solver LIBSVM. Tento solver ma naimplementovaný vyššie spomínaný veľmi efektívny SMO algoritmus. Každý text trénovacích dát sme označili značkou 1, pokiaľ chceme zistiť či tento autor napísal aj texty z testovacích dát. V opačnom prípade sme text označili 0. Vo väčšine prípadov sme poznávali jednotlivé znaky, ako napr. počet slov pomocou medzier, resp. počet viet na základe bodiek. Pred analýzou každého znaku sme sa snažili text ošetriť pred anomáliami, ktoré tam mohli byť a meniť čiastočne hodnoty niektorých charakteristík, ako napríklad trojbodka v prípade počtu viet.

Ako trénovacie dáta sme použili 26 textov, ako testovacie dáta 10 textov. Program každý raz vybral pseudonáhodne 10 textov, ktoré považoval za testovaciu vzorku, výsledky sa preto každým spustením programu menili, čo znamenalo, že dát je málo. S väčším množstvom dát sme narazli s niektorými kernelmi na problém s výpočtovým časom. Taktiež bolo našim cieľom skôr interpretovať použitie aplikácie SVM ako reálne vytvoriť program na spoľahlivú detekciu autorstva.

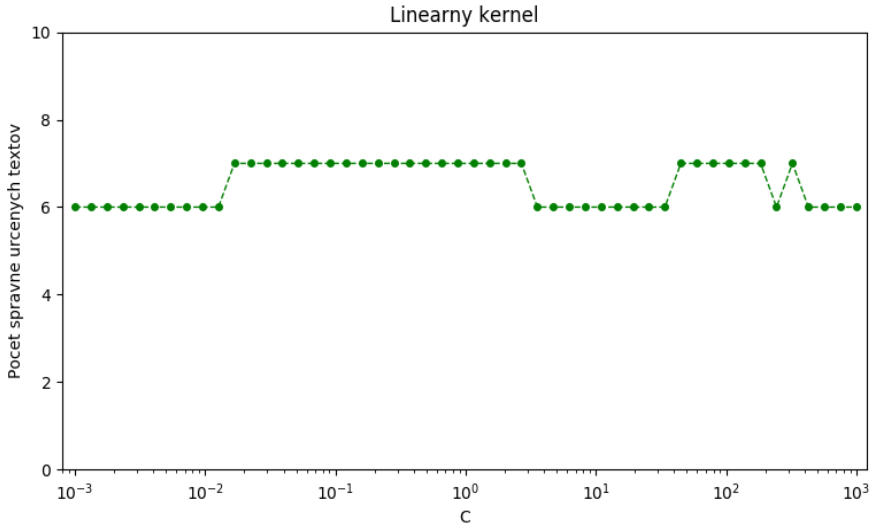
Pri implementácií detekcie autorstva sme použili 3 rôzne kernelové funkcie:

1. **Lineárna kernelová funkcia**

   Lineárna kernelová funkcia je vlastne klasický skalárny súčin teda platí

$$K(x, x') = x^T x'$$

Použtím lineárnej kernelovej funkcie teda riešime problém (2.12), teda úlohu lineárnej separácie so slackovými premennými. jediný parameter, ktorý môžeme pri použití lineárnej kernelovej funkcie použiť je parameter C, ktorý je regularizačný parameter vyjadrujúci trade off medzi dosiahnutím správnej klasifikácie v trénovacích dátach a dosiahnutím väčšej vzdialenosti medzi jednotlivými množinami separovaných dát a oddeľujúcej nadroviny. Pri zvolení malého C získame nadrovinu, ktorej vzdialenosť od separovaných množín z trénovacích dát bude veľká, avšak niektoré body budú klasifikované nesprávne. Naopak, zvolením veľkého C získame nadrovinu, ktorá väčšinu bodov z trénovacej sady dát klasifikuje správne, avšak vzdialenosť nadroviny od separovaných množín bude menšia. Pre lineárnu kernelovú funkciu sme testovali 50 rôznych hodnôt parametra C na logaritmickej škále v rozmedzí od 10^{-3} do 10^{3}. Najväčšiu úspešnosť sme dostali správnym klasifikovaním 7 textov z 10 pre rozličné hodnoty parametra C. Výsledky je možné vidieť na Obrázku 11.



Obr. 11: Výsledky pre 26 trénovacích dát a 10 náhodne vybraných testovacích dát pri použití lineárnej kernelovej funkcie

2. **Polynomiálna kernelová funkcia**

   Polynomiálna kernelová funkcia vyzerá nasledovne:

$$K(x, x') = (x^T x' + r)^d,$$

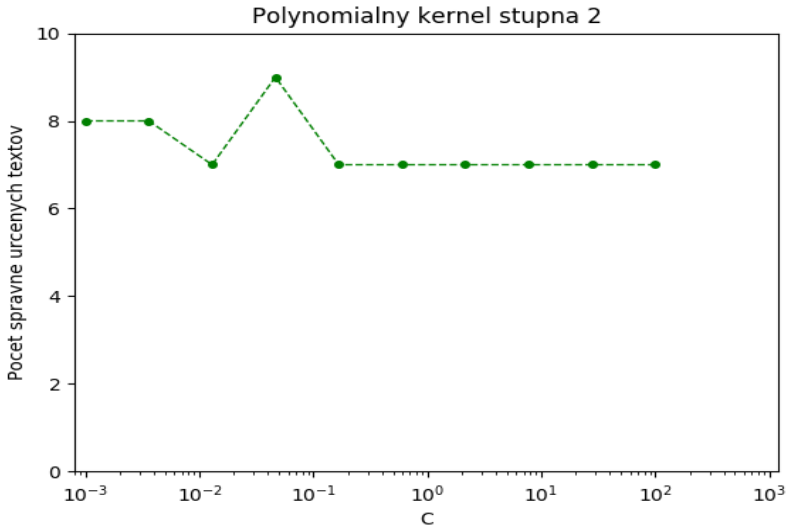
klasický skalárny súčin je len špeciálnym prípadom polynomiálnej kernelovej funkcie. Napriek tomu, že polynomiálna kernelová funkcia nie je až taká často využívaná v bežných aplikáciách SVM, v problémoch spracovania prirodzeného jazyka sa používať zvykne. Označenie d nazývame stupeň polynomiálnej kernelovej funkcie a najčastejšie sa používa $d = 2$, pri väčšom stupni často dochádza k problému "prefitovania", resp. po anglicky overfitting. Problém "prefitovania"je problém, ktorý sa prejavuje v rozličných častiach strojového učenia, dochádza k nemu vtedy, keď sú parametre nastavené tak, aby fitovacia krivka priveľmi kopírovala trénovacie dáta, ktoré častokrát obsahujú aj nejakú chybu. V prípade testovacích dát tak môže dôjsť k veľmi nespoľahlivým predikciám. V polynomiálne kernelovej funkcií môžeme taktiež meniť parameter r. Parameter r zabezpečuje posunutie a v Pythone je predvolene nastavený na hodnotu 0, ktorú v našej aplikácii nebudeme meniť. Pre polynomiálnu kernelovú stupňa 2 sme testovali na 10 rôznych hodnotách na logaritmickej škále v rozmedzí od 10^{-3} do 10^2. Výsledky je možné vidieť na obrázku, pre relatívne veľké hodnoty parametra C sa výsledky nijak nemenia. To však neznamená, že oddeľujúca nadrovina je rovnaká, ale, že všetkých 10 bodov je stále v rovnakých polpriestoroch. Najväčšiu úspešnosť sme získali správnym klasifikovaných 9 textov z 10, pričom 5 textov bolo od autora, ktorého sme skúmali a 5 textov od iných autorov. Výsledky je možné vidieť na Obrázku 12. A

3. **RBF kernelová funkcia**

   RBF kernelová funkcia, resp. Radial Basis Function je najštandardnejšia a najčastejšie používaná nelineárna kernelová funkcia v strojovom učení a má tvar:

$$K(x, x') = e^{-\gamma||x-x'||^2}.$$

   Ako je možné vidieť, narozdiel od ostatných, táto funkcia vôbec nepracuje so skalárnym súčinom dvoch vektorov. Rovnako je, narozdiel od predošlých kernelových funkcii, nekonečne diferencovateľná a vďaka využitiu euklidovskej vzdialenosti miesto skalárneho súčinu, je aj invariantná voči posunutiu. Jediný parameter v RBF kerneli je parameter γ. Tento parameter v podstate predstavuje inverznú hodnotu štandardnej odchýlky normálneho rozdelenia. Pokiaľ sa hodnota tohto parametra zväčšuje, hodnota kernelovej funkcie sa zmenšuje, takže pri

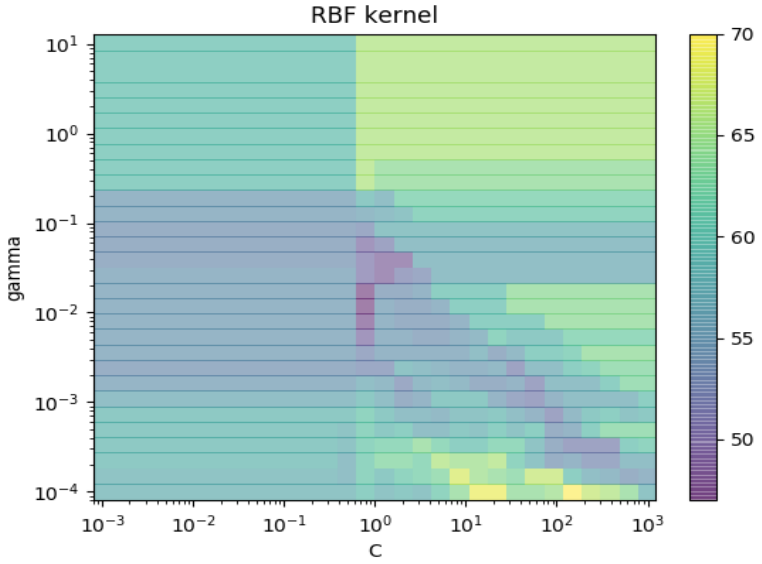


Obr. 12: Výsledky pre 26 trénovacích dát a 10 náhodne vybraných testovacích dát pri použití polynomiálnej kernelovej funkcie

dostatočne veľkej hodnote parametra γ dosahuje kernelová funkcia veľmi nízke hodnoty. Keďže RBF funkcia berie do úvahy vzdialenosť vektorov, hovorí teda v istom zmysle o podobnosti dvoch dát. Ak zvolíme veľkú hodnotu γ, tak hodnota kernelovej funkcie dvoch vektorov bude nízka a teda dve vzorky dát budú považované za podobné, aj napriek tomu, že by v skutočnosti boli tieto vektory od seba veľmi vzdialené. V opačnom prípade, pri zvolení nízkej hodnoty γ, je hodnota kernelovej funkcie veľká a tak aj body z dát, ktoré sú relatívne blízko majú veľmi veľkú hodnotu kernelovej funkcie a tak sú považované za málo podobné.

Štandardne sa hodnota tohto parametra v Pythone určuje ako $\gamma = \frac{1}{počet\_charakteristík}$, táto hodnota je iba inicializačná, keď o povahe dát nevieme veľa a postupne ju upravujeme pre dosiahnutie lepších výsledkov. V našej práci testujeme pre 30 rôznych hodnôt parametra C 30 rozličných hodnôt parametra γ, hodnoty oboch parametrov boli zvolené na logaritmickej škále. Hodnoty C sú v rozmedzí od 10^{-3} do 10^{3}. Hodnoty γ od 10^{-4} do 10^{3}. Pri použití RBF kernelovej funkcie, sme pre rozličné trénovacie a testovacie sady získali odlišné výsledky, to reflektuje stav, že dát nie je až tak veľa a pre spoľahlivý detektor autorstva je potrebné mať dostatočne veľkú sadu dát. Keďže RBF kernelová funkcia bola mimoriadne rýchla, spustili sme 10 testov, kedy každý raz program pseudonáhodne vybral 26

trénovacích dát a 10 testovacích. Spolu teda dokopy určoval 100 textov, pričom každý raz bol natrénovaný na inej sade. Výsledky je možné vidieť na Obrázku 13, kde je na škále zobrazená percentuálna úspešnosť. Zdá sa, že najlepšie výsledky sa dosiahli keď sa γ pohybuje rádovo okolo 10^{-4} a hodnoty C sú väčšie ako 10.



Obr. 13: Výsledky pre 10 testov s každý raz náhodne vybranými trénovacími a testovacími dátami pri použití RBF kernelovej funkcie. Škála zobrazuje percentuálnu úspešnosť

Ako bolo možné vidieť na výsledkoch použitia rozličných kernelových funkcii, výber kernelu výrazne ovplyvňuje dosiahnuté výsledky. Neexistuje všeobecný návod, ako vybrať správny kernel, závisí to od typu riešného problému a od povahy dostupných dát. Častokrát sa ako prvá nelineárna kernelová funkcia testuje RBF. Výber parametrov, ako bolo možné vidieť, rovnako výrazne ovplyvňuje jednotlivé výsledky. Po výbere kernelu tak je nutné zvoliť správne parametre. V súčasnosti existuje viacero metód ako vybrať vhodné parametre, častokrát na základe krížovej validácie alebo pomocou Nelder-Mead simplexovej metódy [13].

Naša implemetácie detekcie autorstva je len zjednodušeným modelom. Pre vybudovanie skutočne spoľahlivého detektoru autorstva by bolo najvhodnejšie použiť nejaký jazykový korpus na získanie dostatočného počtu dát. Trénovacia vzorka by mala byť rádovo niekoľkonásobne väčšia od testovacej. Pre získanie relevantých výsledkov percentuálnej úspešnosti je samozrejme nutné pracovať s omnoho väčšou sadou testovacích dát. To sa potvrdilo pri spúšťaní nášho programu, kedy sme pre rozličné trénovacie dáta

získavali častokrát výrazne odlišné výsledky, pretože dát nebolo veľa a tak každý bod z trénovacej vzorky mal veľký vplyv na výslednú nadrovinu.

Aby sme však mohli pracovať s výrazne väčšou sadou dát, je potrebná omnoho väčšia výpočtová sila. Totiž, už pri malom množstve dát sme mali v prípade polynomiálneho kernelu problémy s veľmi dlhým výpočtovým časom.

Záver

V našej práci sme sa snažili spracovať teóriu kernelových metód iným spôsobom ako sme našli v dostupnej litearúte. Ukázali sme súvis kernelových metód s primárno-duálnymi formuláciami pre SVM klasifikátor a ponúkli sme rôzne geometrické interpretácie daných problémov. Zároveň sme stručne spracovali vlastnosti konečných kernelových funkcii a dali ich do kontextu so štandardne používanými kernelmi. V tejto práci sa venujeme najmä kernelovej SVM metóde, pričom sme spomenululi aj geometrickú interpretáciu SVR. Možným rozšírením tejto práce by tak mohlo byť odvodenie tvaru úlohy SVR po použití kernelového triku.

Okrem spracovanej teórie sme sa pokúsili aj o prehľad používaných algoritmov, ktoré sa na úlohy zvyknú používať, pričom sme uviedli aj niektoré neštandardné geometrické algoritmy. Naprogramovali sme v prostredí jazyka Python aplikáciu kernelových metód z oblasti strojového učenia, konkrétne spracovania prirodzeného jazyka. Podarilo sa nám tak vytvoriť detektor autorstva, ktorý je prispôsobený na odborné texty zo slovenskho jazyka, avšak po miernych úpravách je použiteľný na viacere jazyky. Ďalším rozšírením tejto práce by mohlo byť použitie väčšieho počtu charakteristík autora tak, aby detektor autorstva bol viac zameraný na slovenské texty, resp. na detekovanie textov nie len z odbornej oblasti, ale aj z bežného hovorového jazyka.

Pri písaní práce ma najviac zaujal geometrický prístup k celej problematike, či už k primárnym alebo duálnym úlohám metódy SVM, ale aj geometrický význam využitia kernelového triku. Taktiež ma zaujali podmienky riešiteľnosti úloh po využití kernelového triku, pričom táto problematika je ďalším možným rozšírením tejto práce.

Zoznam použitej literatúry

[1] Bi J., Bennet K.: *A Geometric Approach to Support Vector Regression*, Rensselaer Polytechnic Institute,dostupné na internete (10.2.2018): https://pdfs.semanticscholar.org/0e31fd511f31c97530af87133205ad38f65e3fff.pdf

[2] Boyd S., Vandenberghe L.: *Convex optimization*, Cambridge University Press, Cambridge, 2004

[3] Bottou L., Lin Ch.: *Support Vector Machine Solvers* , National Taiwan University , dostupné na internete (3.4.2018): https://www.csie.ntu.edu.tw/~cjlin/papers/bottou_lin.pdf

[4] Cortes, C., Vapnik, V.: *Support-vector network*, Machine Learning,1-25.s, 1995

[5] Hamala M., Trnovská M.: *Nelineárne programovanie*, Epos, Bratislava, 2012

[6] Jankova J.: *Asymptotic properties of support vector machines*, Univerzita Komenského v Bratislave, Bratislava, 2016

[7] Fisette, M.: *Author identification in short texts*,2010, dostupné na internete (1.3.2018): http://www.socsci.ru.nl/idak/teaching/batheses/MarciaFissette_scriptie.pdf

[8] Hoffman M: *Kernels and the Kernel Trick* , University of Bamberg, dostupné na internete (15.2.2018): http://www.cogsys.wiai.uni-bamberg.de/teaching/ss06/hs_svm/slides/SVM_and_Kernels.pdf

[9] Hofmann, T., Scholkopf, B., Smola, A.: *Kernel methods in machine learning* ,The Annals of Statistics, 2008, 1171–1220 s., dostupné na internete (1.3.2018): http://www.kernel-machines.org/publications/pdfs/0701907.pdf

[10] Horn R.A. ,Johnson Ch.R.: *Matrix Analysis*, Cambridge University Press, 1985

[11] Laskov, B. , Nelson, B.: *Advanced Topics in Machine Learning*, University of Tubingen, dostupné na internete (4.3.2018): http://www.ra.cs.uni-tuebingen.de/lehre/ss12/advanced_ml/lecture3.pdf

[12] Mavroforakis, M. , Theodoridis, S.: *A Geometric Approach to Support Vector Machine (SVM) Classification*, University of Chicago

[13] McCalister D: *Kernel Methods* , University of Chicago, dostupné na internete (15.3.2018): http://ttic.uchicago.edu/~dmcallester/ttic101-07/lectures/kernels/kernels.pdf

[14] Neubrunn T., Riečan B. : *Miera a integrál*, VEDA- Vydavateľstvo SAV, Bratislava 1981.

[15] Ng A: *CS229 Lecture notes* , Stanford University, dostupné na internete (23.3.2018): http://cs229.stanford.edu/notes/cs229-notes3.pdf

[16] Nováková V: *Identifikace autora ve forenzní lingvistice* , Diplomová práca, Univerzita Karlova v Praze, 2014

[17] Pardalos P., Coleman T.: *Lectures on Global optimization*, Fields, Toronto, 2009

[18] Rygl, J.: *Determining Authorship of Anonymous Texts*, návrh PhD práce, Masarykova Univerzita, Brno, 2013

[19] Stano S., Lu D., Hsu, I.: *Whose Book is it Anyway? Using Machine Learning to Identify the Author of Unknown Texts*, Standford, 2013, dostupné na internete (28.11.2017): http://cs229.stanford.edu/proj2013/StankoLuHsu-AuthorIdentification.pdf

[20] Thickstun, J.: *Mercer's Theorem*, University of Washington, dostupné na internete (5.2.2018): https://homes.cs.washington.edu/~thickstn/docs/mercer.pdf

[21] Kernel Functions for Machine Learning Applications, dostupné na internete (8.12.2017): http://crsouza.com/2010/03/17/kernel-functions-for-machine-learning-applications

[22] Support Vector Machines , dostupné na internete (5.2.2018): http://beta.cambridgespark.com/courses/jpm/05-module.html

Príloha 1

Hadamardov súčin

Pre potreby využívania Hadamardovho súčinu v kapitole 2 uvádzame niektoré definície a dôkazy z [10].

Definícia: Pre matice $A, B \in \mathbb{R}^{m\times n}$ definujeme Hadamardov súčin ako: $[A \odot B]_{ij} = A_{ij}B_{ij}\ \forall i, j : 1 \leq i \leq m, 1 \leq j \leq n$.

Veta: Ak $A, B \in \mathbb{R}^{n\times n}$ sú kladne semidefinitné, tak aj $A \odot B$ je kladne semidefinitná.
Dôkaz: Využitím spektrálneho rozkladu môžeme maticu A hodnosti k napísať ako:$A = v_1v_1^T + ... + v_kv_k^T$ a maticu B hodnosti m napísať ako:$A = w_1w_1^T + ... + w_mw_m^T$. Ak označíme $u_{ij} = v_i \odot u_j$, tak $\sum_{i,j=1}^{k,m} u_{ij}u_{ij}^T = A \odot B$. Teda matica $A \odot B$ je súčet kladne semidefinitných matíc hodnosti 1 a teda je kladne semidefinitná.

Dôsledok 1: Matematickou indukciou je možné dokázať, že aj matica A_H^d definovaná ako $\underbrace{A \odot A \odot ... \odot A}_{d}$ je kladne semidefinitná.

Dôsledok 2: Keďže A_H^d je kladne semidefinitná matica, tak aj súčet rôznych mocníc v zmysle Hadamardovho súčinu je kladne semidefinitná matica. Keďže podľa [2] kladne semidefinitné matice tvoria uzavretý kužeľ, limita postupnosti kladne semidefinitných matíc je opäť kladne semidefinitná matica, tak aj limita nekonečnej sumy kladne semidefintých matíc je kladne semidefinitná matica. Vďaka tomu môžeme označiť $e_H^A = \lim_{N\to\infty} \sum_{i=0}^{N} \frac{A_H^i}{i!}$, o ktorej z vyššie uvedeného vieme, že je kladne semidefinitná.

Príloha 2

Zdrojový kód

```
\begin{verbatim}
from __future__ import division
import numpy as np
import scipy.misc
from sklearn import svm
import math
from pylab import scatter
import pylab
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches
import random

def load_data(traindata):
    Textset=[]
    labels=[]
    for i in traindata:
        file = open('dataset2/data'+str(i)+'.txt', 'r')
        Data = file.read()
        Data = Data.replace('\n', '')
        labels.append(int(Data[0]))
        Textset.append(Data[1:])
        file.close()
    return labels,Textset

def word_count(data):
    data = data.replace('. ','.')
    data = data.replace(',',', ')
    data = data.replace('  ',' ')
    sentences = data.split('.')[:-1]
    sentenceCount = len(sentences)
    return (sum([len(s.replace(',','').split(' ')) for s in sentences])/sentenceCount)

def sentence_length_variation(data):
    data = data.replace('. ','.')
    sentences = data.split('.')[:-1]
    npWords = np.array([len(s.replace(',','').split(' ')) for s in sentences])
    return npWords.std()

def comma_count(data):
    data = data.replace('. ','.')
    commas = data.split(',')
    sentences = data.split('.')
    return (len(commas)-1)/len(sentences)

def stop_list(data):
    stopArr = ['a','aby','aj','ak','ako','ale','alebo','and','ani','áno','asi','až',
```

```
'bez','bude','budem','budeš','budeme','budete','budú','by','bol',
'bola','boli','bolo','byť','cez','čo','či','ďalší','ďalšia','ďalšie',
'dnes','do','ho','ešte','for','i','ja','je','jeho','jej','ich','iba',
'iné','iný','som','si','sme','sú','k','kam','každý','každá','každé',
'každí','kde','keď','kto','ktorá','ktoré','ktorou','ktorý','ktorí',
'ku','lebo','len','ma','mať','má','máte','medzi','mi','mna','mne','mnou','musieť',
    'môcť','môj','môže','my','na','nad','nám','náš','naši','nie',
    'nech','než','nič','niektorý','nové','nový','nová','nové','noví',
    'o','od','odo','of','on','ona','ono','oni','ony','po','pod',
    'podľa','pokiaľ','potom','práve','pre','prečo','preto','pretože',
    'prvý','prvá','prvé','prví','pred','predo','pri','pýta','s','sa',
    'so','si','svoje','svoj','svojich','svojím','svojími','ta','tak',
    'takže','táto','teda','te','tě','ten','tento','the','tieto','tým',
    'týmto','tiež','to','toto','toho','tohoto','tom','tomto','tomuto',
    'toto','tu','tú','túto','tvoj','ty','tvojími','už','v','vám','váš',
    'vaše','vo','viac','však','všetok','vy','z','za','zo','že']
    stopCount = 0
    words = data.replace('.','')
    words = words.replace(',','')
    words = words.split(' ')
    for word in words:
        if word in stopArr:
            stopCount += 1
    all_words = len(words)
    return stopCount/all_words

def vocabulray_richness(data):
    words_using = {}
    words = data.replace('.','')
    words = words.replace(',','')
    words = words.split(' ')
    for word in words:
        if word in words_using:
            words_using[word] += 1
        else:
            words_using[word] = 1
    all_words = len(words)
    sumN = 0
    for word in words:
        sumN += words_using[word]**2
    return 10000*(sumN-all_words)/all_words**2

def hapax_legomena(data):
    words_using = {}
    words = data.replace('.','')
    words = words.replace(',','')
    words = words.split(' ')
    for word in words:
        if word in words_using:
            words_using[word] += 1
```

```
        else:
            words_using[word] = 1
    hapax = 0
    for word in words:
        if(words_using[word] == 1):
            hapax += 1
    return hapax/len(words)

def prepareData(n,size,maxIter,method):
    features = []
    alldata = [i for i in range (n)]
    randomarray = []
    while len(randomarray) != size:
        radnum = random.randint(0,n-1)
        if (radnum not in randomarray):
            randomarray.append(radnum)
    testdata = [item for item in alldata if (item in randomarray)]
    traindata = [item for item in alldata if (item not in testdata)]
    dataset = load_data(traindata)
    print(traindata,testdata)
    labels = dataset[0]
    textset = dataset[1]
    for text in textset:
        feature=[]
        feature.append(word_count(text))
        feature.append(comma_count(text))
        feature.append(vocabulray_richness(text))
        feature.append(sentence_length_variation(text))
        feature.append(stop_list(text))
        feature.append(hapax_legomena(text))
        features.append(feature)
    print(labels)
    predicts = []
    true_array = []
    for i in testdata:
        file = open('dataset2/data'+str(i)+'.txt', 'r')
        predict = []
        text = file.read()
        true_array.append(int(text[0]))
        text = text.replace('\n', '')
        predict.append(word_count(text))
        predict.append(comma_count(text))
        predict.append(vocabulray_richness(text))
        predict.append(sentence_length_variation(text))
        predict.append(stop_list(text))
        predict.append(hapax_legomena(text))
        predicts.append(predict)
        file.close()
    print(true_array)
    if method=="rbf":
```

```
    clogas = np.logspace(math.log(0.001,10),math.log(1000,10),30)
    gammalogas = np.logspace(math.log(0.0001,10),math.log(10,10),30)
    results = []
    size = []
    x = []
    y = []
    col = []
    values = ['0','1','2','3','4','5','6','7','8','9','10']
    colours = ['white','black','dimgray','teal','steelblue',
    'powderblue','cyan','maroon','coral','olive','gold']
    for j in gammalogas:
        for i in clogas:
            clf = svm.SVC(kernel='rbf',gamma=j,C=i)
            clf.fit(features, labels)
            predicted = clf.predict(predicts)
            validCount = np.sum(true_array == predicted)
            results.append(validCount)
            size.append((2.3)**4)
            col.append(colours[validCount])
            x.append(i)
            y.append(j)
    print(max(results))
    sameValues = np.array(size)
    x = np.array(x)
    y = np.array(y)
    ax = plt.gca()
    ax.scatter(x=x, y=y, s=sameValues, label=results, c=col, alpha=0.5)
    ax.set_yscale('log')
    ax.set_xscale('log')
    ax.set_xlim([0.0008,1200])
    ax.set_ylim([0.00008,13])
    recs = []
    for i in range(0,len(colours)):
        recs.append(mpatches.Rectangle((0,0),1,1,fc=colours[i]))
    plt.legend(recs,values,loc=2,bbox_to_anchor=(1.01, 1),borderaxespad=0.)
    plt.title("RBF kernel")
    plt.xlabel("C")
    plt.ylabel('gamma')
    pylab.show()
if method=="linear":
    clogas = np.logspace(math.log(0.001,10),math.log(1000,10),10)
    results = []
    x = []
    for i in clogas:
        clf = svm.SVC(kernel='poly',degree=1,C=i)
        clf.fit(features, labels)
        predicted = clf.predict(predicts)
        validCount = np.sum(true_array == predicted)
        results.append(validCount)
        x.append(i)
```

```
        x = np.array(x)
        y = np.array(results)
        ax = plt.gca()
        ax.plot(x, results, 'go--', linewidth=1, markersize=4)
        ax.set_xscale('log')
        ax.set_xlim([0.0008,1200])
        ax.set_ylim([0,10])
        plt.title("Linearny kernel")
        plt.ylabel("Pocet spravne urcenych textov")
        plt.xlabel("C")
        pylab.show()
    if method=="poly":
        clogas = np.logspace(math.log(0.001,10),math.log(100,10),10)
        results = []
        x = []
        for i in clogas:
            clf = svm.SVC(kernel='poly',degree=2,C=i)
            clf.fit(features, labels)
            predicted = clf.predict(predicts)
            validCount = np.sum(true_array == predicted)
            results.append(validCount)
            x.append(i)
        x = np.array(x)
        y = np.array(results)
        ax = plt.gca()
        ax.plot(x, results, 'go--', linewidth=1, markersize=4)
        ax.set_xscale('log')
        ax.set_xlim([0.0008,1200])
        ax.set_ylim([0,10])
        plt.title("Polynomialny kernel stupna 2")
        plt.ylabel("Pocet spravne urcenych textov")
        plt.xlabel("C")
        pylab.show()
prepareData(36,10,5,"poly")
```