UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

BAYESOVSKÁ OPTIMALIZÁCIA HYPERPARAMETROV METÓD STROJOVÉHO UČENIA

DIPLOMOVÁ PRÁCA

2022 Bc. Dominik NAJŠEL

UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

BAYESOVSKÁ OPTIMALIZÁCIA HYPERPARAMETROV METÓD STROJOVÉHO UČENIA

DIPLOMOVÁ PRÁCA

Študijný program:	Ekonomická a finančná matematika
Študijný odbor:	1113 Matematika
Školiace pracovisko:	Katedra aplikovanej matematiky a štatistiky
Vedúci práce:	doc. Mgr. Radoslav Harman, PhD.

Bratislava 2022 **Bc. Dominik NAJŠEL**

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

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

Názov: Bayesovská optimalizácia hyperparametrov metód strojového učenia
Bayesian optimization of the hyperparameters of machine learning methods

Anotácia: Základnou myšlienkou Bayesovskej optimalizácie je využiť pravdepodobnostné modelovanie na vyjadrenie neurčitosti rozsahu hodnôt účelovej funkcie v dosiaľ nepreskúmaných bodoch priestoru prípustných riešení. Bayesovská optimalizácia sa používa predovšetkým pre optimalizačné problémy, v ktorých majú vektory prípustných riešení malý rozmer, avšak evaluácia hodnôt účelovej funkcie je veľmi nákladná alebo zdĺhavá. Typickou aplikáciou je optimalizácia hyperparametrov pre metódy strojového učenia.

Vedúci: doc. Mgr. Radoslav Harman, PhD.
Katedra: FMFI.KAMŠ - Katedra aplikovanej matematiky a štatistiky
Vedúci katedry: prof. RNDr. Marek Fila, DrSc.

Dátum zadania: 13.01.2021

Dátum schválenia: 17.01.2021 prof. RNDr. Daniel Ševčovič, DrSc.
garant študijného programu

.....
študent

.....
vedúci práce

Poďakovanie Touto cestou by som sa rád úprimne poďakoval svojmu vedúcemu diplomovej práce doc. Mgr. Radoslavovi Harmanovi, PhD., za jeho čas, ochotu, podnetné pripomienky a odborné rady, ktorými mi pomohol pri písaní tejto diplomovej práce. Rovnako veľké ďakujem patrí aj mojim rodičom, bratovi a kamarátom, ktorý mi boli oporou a dopriali mi pokoj a priestor počas písania tejto práce.

Abstrakt

NAJŠEL, Dominik: Bayesovská optimalizácia hyperparametrov metód strojového učenia [Diplomová práca], Univerzita Komenského v Bratislave, Fakulta matematiky, fyziky a informatiky, Katedra aplikovanej matematiky a štatistiky; školiteľ: doc. Mgr. Radoslav Harman, PhD., Bratislava, 2022, 58 s.

Táto diplomová práca sa zaoberá Bayesovskou optimalizáciou parameterov metód strojového učenia. V prvej časti práce prinášame prehľad teórie Bayesovskej optimalizácie a metód oporných bodov. Špeciálne sa venujeme problému ako pomocou metódy, ktorá je primárne určená na binárnu klasifikáciu, skonštruovať multikategorický klasifikátor. V druhej časti práce sa zaoberáme hľadaním hodnôt najlepších hyperparametrov pomocou Bayesovskej optimalizácie, pre prípad metódy oporných bodov aplikovanej na problém klasifikácie rukou písaných cifier do kategórií 0-9. Výsledný klasifikátor má úspešnosť správneho určenia cifry takmer 95 percent.

Kľúčové slová: Bayesovská optimalizácia, strojové učenie, metóda oporných bodov, hyperparameter, binárna klasifikácia, multikategoriálna klasifikácia

Abstract

NAJŠEL, Dominik: Bayesian optimization of the hyperparameters of machine learning methods [Master Thesis], Comenius University in Bratislava, Faculty of Mathematics, Physics and Informatics, Department of Applied Mathematics and Statistics; Supervisor: doc. Mgr. Radoslav Harman, PhD., Bratislava, 2022, 58 p.

The topic of this thesis is Bayesian optimization of hyperparameters of machine learning methods. In the first part of the thesis we present an overview of the Bayesian optimization theory and methods of support vector machines. We especially focus on solving the problem of constructing a multicategorical classifier using methods that are primarily intended for binary classification. In the second part of the work we try to find the best values of hyperparameters using Bayesian optimization, for the case of the methods of support vector machines applied to the problem of classifying the handwritten digits into categories 0-9. The resulting classifier has a success rate of almost 95 percent.

Keywords: Bayesian optimization, machine learning, support vector machines, hyperparameter, binary classification, multiclass classification

Obsah

Úvod

Optimalizácia je v súčasnosti veľmi dôležitou oblasťou v matematike a informatike. Hlavnou úlohou optimalizácie je nájsť optimálnu hodnotu (minimum alebo maximum) účelovej funkcie na zadanej množine prípustných riešení. V dnešnej dobe už existuje veľké množstvo optimalizačných metód pre rôzne účelové funkcie a množiny prípustných riešení.

Takzvaná Bayesovská optimalizácia bola navrhnutá už v 70. rokoch 20. storočia, avšak jej užitočnosť sa naplno prejavila až s nástupom modernej výpočtovej techniky a snahou o riešenie veľmi zložitých optimalizačných úloh.

V optimalizácii existujú rôzne prístupy na výber vhodnej metódy v závislosti od špecifických vlastností riešeného optimalizačného problému.

Podľa typu priestoru prípustných riešení môžeme rozdeliť optimalizačné problémy na úlohy diskrétnej, resp. spojitej optimalizácie. Úlohy spojitej optimalizácie je možné ďalej deliť podľa analytických vlastností účelovej funkcie, napríklad na úlohy konvexnej, resp. nekonvexnej optimalizácie, alebo na problémy s diferencovateľnou, resp. nediferencovateľnou účelovou funkciou. Tieto aspekty optimalizačného problému musíme brať do úvahy, keď vyberáme metódu na riešenie.

Dôležitým faktorom pri výbere optimalizačnej metódy je aj to, ako náročné je vypočítať funkčnú hodnotu účelovej funkcie; treba si uvedomiť, že rozdiel medzi časom potrebným na evaluáciu jednoduchej účelovej funkcie a evaluáciu zložitej účelovej funkcie (pri ktorej je napríklad potrebné vykonať rozsiahlu simuláciu, alebo počítať systém diferenciálnych rovníc) môže byť mnoho rádov.

Bayesovská optimalizácia sa najčastejšie používa na riešenie úloh spojitej optimalizácie, pre spojitú, avšak nekonvexnú, prípadne aj nediferencovateľnú účelovú funkciu, a to najmä v prípade, že evaluácia účelovej funkcie je veľmi náročná. Bayesovská optimalizácia totiž využíva len funkčné hodnoty účelovej funkcie (nie napríklad jej gradient), no súčasne "šetrí" počtom evaluácií účelovej funkcie.

Cieľom tejto práce bude podrobnejšie vysvetlenie ako a prečo sa používa Bayesovská optimalizácia a následne aplikovanie Bayesovskej optimalizácie pre metódu výberu hyperparametrov pre problém multikategorickej optimalizácie metódou oporných bodov (angl. *support vector machines*, skrátene SVM) na dátový súbor z oblasti strojového učenia.

V prvej kapitole si uvedieme formuláciu všeobecného optimalizačného problému a k

čomu je dobrá Bayesovská optimalizácia. Vysvetlíme si základné pojmy, na ktorých je Bayesovská optimalizácia postavená, ako napríklad gaussovské procesy, akvizičná funkcia a tiež predstavíme základný algoritmus Bayesovskej optimalizácie.

V druhej kapitole vysvetlíme, čo je to metóda oporných bodov, vysvetlíme čo sú to hyperparametre a prečo sa používajú pri tejto metóde. Taktiež priblížime ako sa dá binárna SVM použiť na multikategoriálnu klasifikáciu. Bližšie predstavíme dátový subor *Pen-Based Recognition of Handwritten Digits*, s ktorým budeme ďalej pracovať. Dátový subor si vizualizujeme pomocou metódy hlavných komponentov.

V poslednej kapitole budeme za pomoci Bayesovskej optimalizácie optimalizovať hyperparametre pri SVM na náš dátový súbor. Týmto postupom získame algoritmus na rozpoznávanie rukou písaných cifier so spoľahlivosťou skoro 95 percent.

1 Bayesovská optimalizácia

V prvej kapitole sa oboznámime s konceptom Bayesovskej optimalizácie (angl. *Bayesian optimization*), jej využitím a výhodami oproti iným optimalizačným metódam. Uvedieme základné definície na pochopenie problematiky a vysvetlíme si základný algoritmus Bayesovskej optimalizácie.

Ukážeme si jednoduchý príklad, kde Bayesovská optimalizácia viedla k podobnému výsledku ako iné metódy, avšak na základe podstatne menšieho počtu evaluácii optimalizovanej funkcie (ktorú pre potreby tejto práce budeme nazývať „účelová" funkcia). Počas tejto kapitoly budeme vychádzať prevažne z článkov [3], [12] a kníh [7], [10] a [14].

1.1 Optimalizácia výpočtovo náročných funkcií

Základnou ideou globálnej optimalizácie je nájdenie globálneho maxima, resp. minima účelovej funkcie. Bayesovská optimalizácia sa používa najmä na prístup k optimalizácii účelových funkcií, ktorých vyhodnotenia trvajú dlhý čas (minúty až hodiny). Toto je typické pre funkcie, ktoré nevieme analyticky vyjadriť, ale máme k dispozícií len zložitý algoritmus, ktorý hodnoty tejto funkcie počíta numericky (ide o takzvanú „black-box" optimalizáciu)[1].

Optimalizačné metódy napríklad gradientné metódy ako Newtonova metóda, kváziNewtonova metóda alebo klesajúca gradientná metóda nemusia byť v takejto situácii vhodné, pretože nepoznáme explicitný predpis pre gradient účelovej funkcie a numerický výpočet tohto gradientu by bol časovo veľmi náročný.

Bayesovská optimalizácia patrí do triedy optimalizačných metód založených na pravdepodobnostnom modelovaní neurčitosti, ktoré riešia problém

$$\text{argmax}_{x \in X} f(x),$$

kde výpočtovo náročnú funkciu $f : X \to R$, $X \subset R^d$, $d \in N$, nazývame účelová funkcia a X je množina prípustných riešení [2]. Veľkosť dimenzie d býva spravidla menšia ako 20.

[1] Iná situácia, na ktorú je vhodná Bayesovská optimalizácia, je keď je na nájdenie hodnoty účelovej funkcie pre daný argument potrebné uskutočniť fyzický experiment, ktorý je často zdĺhavý a nákladný. Takýmto typom aplikácie sa však nebudeme v tejto práci zaoberať.

[2] Typicky ide o „jednoduchú" množinu, napríklad obdĺžnik v R^2.

Predpokladáme, že účelová funkcia je spojitá. Pri optimalizácii sa riešia dva základné aspekty, a to koľko potrebujeme evaluácii funkčných hodnôt, aby sme našli optimum a ako je algoritmicky náročné určiť, v ktorých bodoch má byť účelová funkcia evaluovaná. Bayesovská optimalizácia je primárne určená na úlohy, v ktorých je potrebné šetriť počtom evaluácií, ale nie je až také dôležité, aby bolo jednoduché určiť v ktorých bodoch evaluovať účelovú funkciu.

1.2 Gaussovské procesy

Kedže našu účelovú funkciu nepoznáme, budeme ju považovať za neznámu realizáciu gaussovského náhodného procesu[3]. Pri Bayesovskej optimalizácii sa snažíme zostrojiť pravdepodobnostný model, ktorý využíva apriórne pravdepodobnostné rozdelenie na parametre gaussovského procesu, pričom postupné zisťovanie hodnôt účelovej funkcie v strategicky zvolených bodoch mení toto apriórne rozdelenie na aposteriórne rozdelenia so zväčšujúcou sa informáciou ohľadom konkrétnej realizácie tohto procesu, čiže optimalizovanej funkcie. Na definovanie gaussovských procesov a samotného algoritmu Bayesovskej optimalizácie si najprv zavedieme pojem mnohorozmerného normálneho rozdelenia, pojem funkcie strednej hodnoty a pojem kovariančnej matice.

Definícia 1.1. *(Regulárne mnohorozmerné normálne rozdelenie). [14] Nech $\mu \in R^m$ a nech Σ je pozitívne definitná (t.j. regulárna pozitívne semidefinitná) matica typu $m \times m$. Nech $X = (X_1, ..., X_m)^T$ je spojitý náhodný vektor s hustotou*

$$f(x) = \frac{1}{\sqrt{(2\pi)^m \det(\Sigma)}} \exp\left(-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\right).$$

Potom hovoríme, že X má regulárne m-rozmerné normálne rozdelenie s parametrami μ a Σ. Túto skutočnosť značíme $X \sim \mathcal{N}_m(\mu, \Sigma)$.

Veta 1.2. *[7] Nech $X \sim \mathcal{N}_m(\mu, \Sigma)$. Potom stredná hodnota je $E(X) = \mu$ a kovariančná matica $D(X) = \Sigma$.*

Definícia 1.3. *(Gaussovský proces). [10] Hovoríme, že náhodná funkcia $f : X \to R$ je gaussovský proces charakterizovaný strednou hodnotou $\mu_0 : X \to R$ a pozitívne definitnou kovariančnou funkciou $\kappa : X \times X \to R$, ktorá je spojitá na diagonále, ak pre ľubovoľné*

[3]Pri dimenzii $d \geq 2$ sa používa aj pojem gaussovské „pole".

$N \in \mathcal{N}$ a pre ľubovoľné $x_1, ..., x_N \subset X$ má vektor $(f(x_1), ..., f(x_N))^T$ združené normálne rozdelenie so strednou hodnotou $(\mu_0(x_1), \ldots, \mu_0(x_N))^T$ a kovariančnou maticou Σ, kde $\Sigma_{ij} = \kappa(x_i, x_j)$. Označujeme $f \sim \mathcal{GP}(\mu_0, \kappa)$.

1.3 Stredná funkcia a jadro

Vlastnosti gaussovského rozdelenia na funkciách sú určené funkciou strednej hodnoty $\mu_0 : X \to R$ a kladne definitnou kovariančnou funkciou $\kappa : X \times X \to R$.

V prípade aplikácií v oblasti Bayesovskej optimalizácie, za najčastejšiu zvolenú funkciu strednej hodnoty je zvolená konštantá hodnota $\mu_0(x) = \mu$.

Kovariančná funkcia (nazývaná tiež jadro) opisuje kovarianciu náhodných premenných gaussovského procesu. Spolu so strednou funkciou jadro úplne definuje gaussovský proces. Jadro je funkciou dvoch argumentov a často uvažujeme o $\kappa(x, x')$ ako o číselnej charakteristike toho, nakoľko sú podobné resp. ako spolu súvisia pozorovania/merania v bodoch x a x'. Funkcia jadra je typicky volená tak, že korelácie medzi náhodnými premennými zodpovedajúcimi bodom x a x' rastú k hodnote jedna s klesajúcou vzdialenosťou bodov. Toto reprezentuje skutočnosť, že očakávame, že hodnoty účelovej funkcie sa vo všeobecnosti líšia menej pre blízke body a viac pre vzdialené body, resp. skutočnosť, že účelová funkcia je spojitá.

Medzi základne typy kovariančných funkcií patrí napríklad takzvané „štvorcové exponenciálne jadro" (angl. *squared exponential kernel*), inak nazývané aj gaussovské jadro, ktoré má nasledovný tvar:

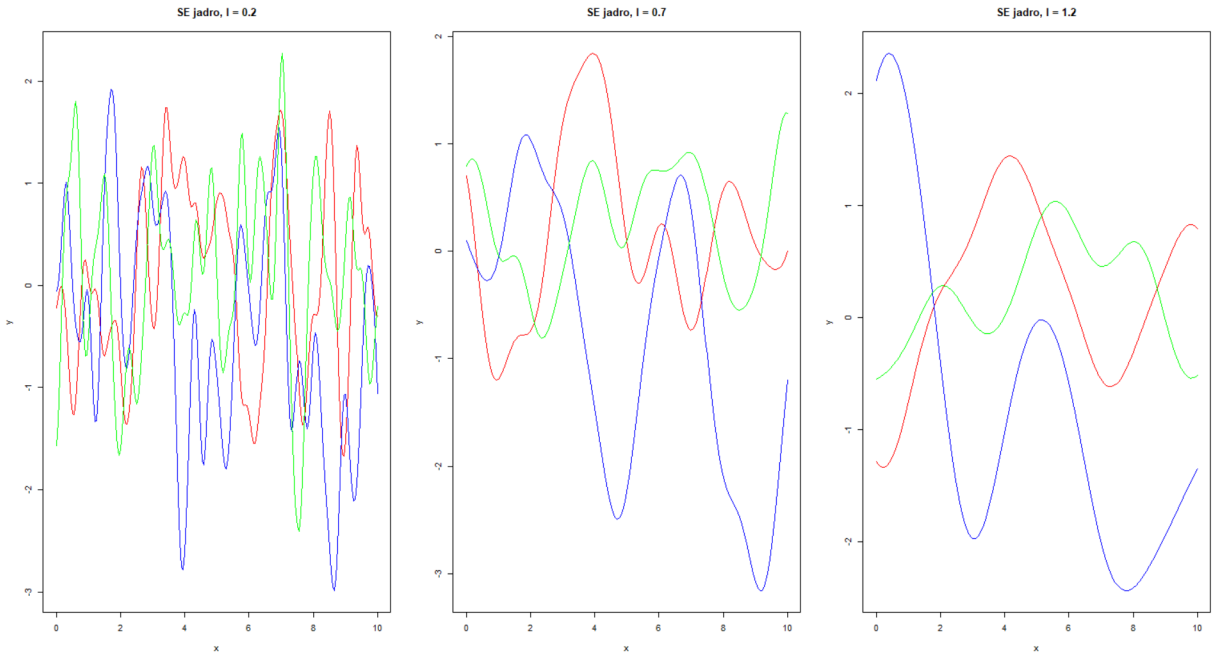
$$\Sigma_{SE}(x, x') = \sigma^2 \exp\left(-\frac{1}{2}\left(\frac{||x - x'||}{l}\right)^2\right), \tag{1}$$

kde $\sigma^2 > 0$ je parameter, ktorý určuje priemernú vzdialenosť funkcie od jej strednej hodnoty. Každé jadro má tento parameter a považuje sa len za mierkový parameter, $l > 0$ (nazývaný aj „lenghtscale") určuje dĺžku „pohybov" vo funkcii. Spolu tvoria parametre tohto jadra, ktoré teda závisí len od vzdialenosti medzi dvoma argumentmi.

Táto kovariančná funkcia je nekonečne diferencovateľná, čo znamená, že gaussovský proces s touto kovariančnou funkciou má derivácie všetkých rádov, a teda je hladká.

Na Obr. 1 môžeme vidieť znázornené náhodne funkcie s 1-rozmerným vstupom odvodeným z apriórneho gaussovského procesu so štvorcovým exponenciálnym jadrom zo

vzťahu (1) pre rôzne hodnoty parametra l a konštantným nastavením $\sigma^2 = 1$.



Obr. 1: Náhodné funkcie f získané z apriórneho gaussovského procesu so štvorcovým exponenciálnym jadrom. Každý graf zodpovedá zľava doprava hodnotám l=0.2, l=0.7 a l= 1.2 a zafixovanému parametru $\sigma^2 = 1$. Pre odlišné hodnoty parametra vidíme ako rýchlo sa $f(x)$ mení s x, teda čím sú hodnoty l menšie, tým funkcie častejšie „skáču".

Predpoklady hladkosti sú nereálne pre modelovanie mnohých fyzikálnych procesov a odporúča sa použiť Maternovo jadro (angl. *Matern kernel*). Toto jadro sa častejšie využíva v praxi a okrem parametrov σ^2 a l obsahuje parameter ν, ktorý riadi „mieru hladkosti" výslednej funkcie.

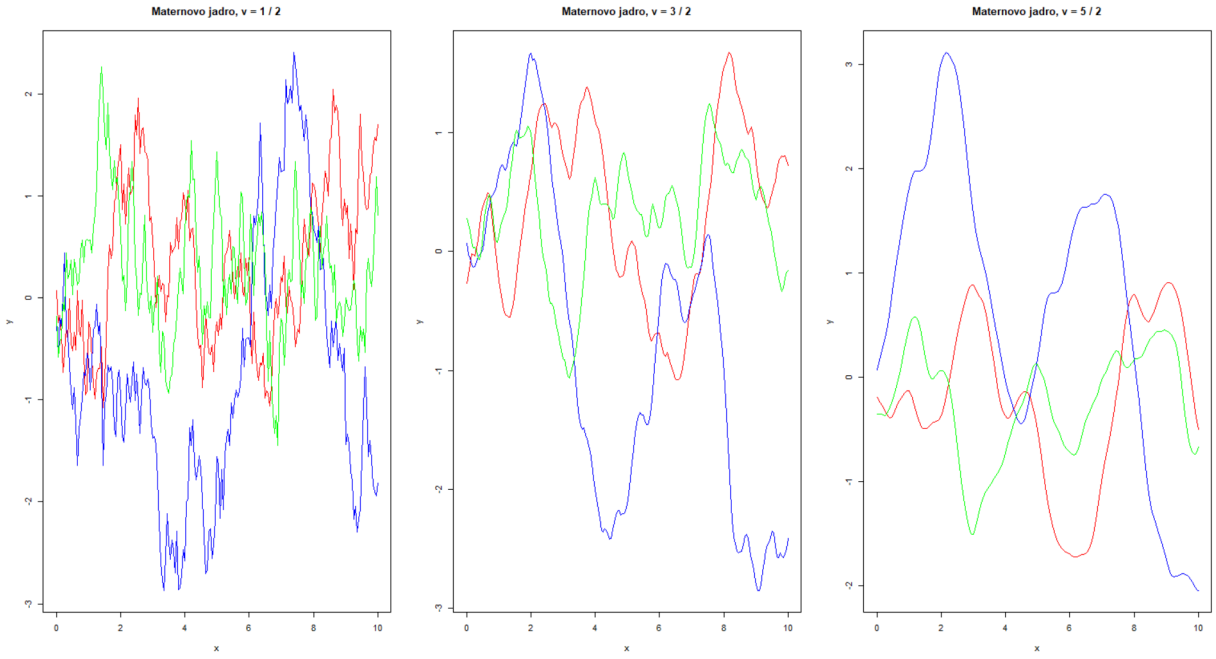
Predpis Maternovho jadra vyzerá nasledovne podľa [10]:

$$\Sigma_{M52}(x, x') = \sigma^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu}\|x - x'\|}{l} \right)^{\nu} K_\nu \left(\frac{\sqrt{2\nu}\|x - x'\|}{l} \right), \tag{2}$$

kde $\sigma^2 > 0$, $l > 0$, $\nu > 0$, Γ predstavuje Gamma funkciu a K_ν je modifikovaná Besselova funkcia.

V praxi sa parameter ν najčastejšie nastavuje na hodnoty $\nu = \frac{3}{2}$ alebo $\nu = \frac{5}{2}$.

Na Obr. 2 si ukážeme rovnaké náhodné funkcie f ako pri štvorcovom exponeniálnom jadre získané z apriórneho gaussovského procesu, avšak s Maternovým jadrom podľa vzťahu (2) pre rôzne hodnoty ν a konštantnými hodnotami $\sigma^2 = 1$ a $l = 1$.



Obr. 2: Náhodné funkcie f získané z apriórneho gaussovského procesu s Maternovým jadrom pre rôzne hodnoty ν. Jednotlivé grafy zodpovedajú zľava doprava hodnotám $\nu = \frac{1}{2}$, $\nu = \frac{3}{2}$ a $\nu = \frac{5}{2}$.

Rovnako ako pri štvorcovom exponenciálnom jadre pre odlišné hodnoty parametra vidíme ako rýchlo sa $f(x)$ mení s x, teda čím sú hodnoty menšie, tým funkcie častejšie „skáču".

Táto kovariančná funkcia Maternovho jadra vedie k účelovým funkciám, ktoré sú dvakrát diferencovateľné, čo je predpoklad, ktorý zodpovedá funkciám, ktoré je možné optimalizovať napríklad kvázi-Newtonovými metódami, ale bez potreby hladkosti druhej mocniny exponenciály.

1.4 Algoritmus Bayesovskej optimalizácie

Samotný algoritmus Bayesovskej optimalizácie pozostáva z viacerých krokov a výpočtov. Základom Bayesovskej optimalizácie je apriórny pravdepodobnostný model pre účelovú funkciu f vyjadrený ako gaussovský proces. Tento proces zachytáva jednak našu apriórnu neurčitosť funkčných hodnôt $f(x)$ pre kandidátov $x \in X$, ale aj korelácie medzi týmito funkčnými hodnotami.

Na aktualizovanie apriórneho rozdelenia pri novom bode a vypočítanie novej funkčnej

hodnoty potrebujeme vypočítať podmienené združené normálne rozdelenie.

Predpokladajme, že pozorujeme $f(x_1), ..., f(x_n)$ pre nejaké n a nejaké známe body $x_1, ..., x_n$. Za podmienky znalosti týchto funkčných hodnôt chceme odvodiť pravdepodobnostné rozdelenie pre hodnotu $f(x)$ pre nejaký nový bod x (uvedomíme si, že f je náhodný proces, takže $f(x)$ je náhodná premenná). Aby sme to urobili, nastavíme $N = n + 1$ a $x_N = x$ také, že apriórne rozdelenie je dané vzťahom z Definície 1.3, teda

$$f(x|x_1, ..., x_n) \sim \mathcal{N}(\mu_n(x), \kappa_n(x)),$$

potom strednú hodnotu a rozptyl daného podmieneného rozdelenia dostaneme na základe vzťahov

$$\mu_n(x) = \Sigma_0(x; x_1, ..., x_n)\Sigma_0(x_1, ..., x_n; x_1, ..., x_n)^{-1}(f(x_1, ..., x_n) - \mu_n(x_1, ..., x_n)) + \mu_0(x),$$
$$\kappa_n(x) = \Sigma_0(x; x) - \Sigma_0(x; x_1, ..., x_n)\Sigma_0(x_1, ..., x_n; x_1, ..., x_n)^{-1}\Sigma_0(x_1, ..., x_n; x),$$

kde $f(x_1, ..., x_n) = (f(x_1), ..., f(x_n))^T$ predstavuje vektor funkčných hodnôt, $\mu_n(x_1, ..., x_n) = (\mu(x_1), ..., \mu(x_n))^T$ je vektor stredných hodnôt a $\Sigma_0(x_1, ..., x_k; y_1, ..., y_r)$ je matica $k \times r$, ktorej ij-ty prvok predstavuje kovarianciu pozorovaní apriórneho gaussovského procesu v bodoch x_i a y_j. Špeciálne $\Sigma_0(x; x)$ je $\kappa(x, x)$ z definície 1.3.

V každom kroku teda vyčíslime f najprv v nejakom novom bode a po zistení $f(x)$ aktualizujeme momentálne apriórne rozdelenie na aposteriórne (a to také, ktoré zodpovedá procesu nadobúdajúcemu v bode x hodnotu $f(x)$ s pravdepodobnosťou 1), ktoré predstavuje náš hľadaný model. Následne za pomoci aposteriórneho rozdelenia vypočítame akvizičnú funkciu (angl. *acquisition function*), o ktorej si povieme v sekcii 1.5.

Potom maximalizujeme našu akvizičnú funkciu a dostaneme novú hodnotu x, v ktorej vykonáme ďalšiu evaluáciu. Tento cyklus opakujeme určitý počet iterácii, ktorý zadáme, alebo pokiaľ nezískame najlepšiu hodnotu pre naše hľadané x.

Základný algoritmus Bayesovskej optimalizácie:

1. Uvažujeme apriórny gaussovský proces $\pi(.)$ na skutočnú účelovú funkciu f a celkový počet evaluácií funkcie f, ktorý označíme N.
2. Zvolíme n_0 počiatočných bodov $x_1, ..., x_{n_0}$.
3. Vypočítame funkčné hodnoty $f(x_1), ..., f(x_{n_0})$.
4. Nastavíme $n = n_0$.

5. Pokiaľ $n \leq N$ urobíme:

   5.1. Aktualizujeme aposteriórne rozdelenie pomocou všetkých doterajších údajov, pričom funkčné hodnoty $f(x_1), ..., f(x_n)$ chápeme ako pozorovania apriórneho gaussovského náhodného poľa v bodoch $x_1, ..., x_n$.

   5.2. Vypočítame $a(x|\pi, f(x_0), ..., f(x_n))$, kde a je takzvaná akvizičná funkcia závislá od aposteriórneho rozdelenia, z ktorej určíme ďalší bod na evaluáciu.

   5.3. Nech x^* je hodnota, ktorá maximalizuje a.

   5.4. Vypočítame $f(x^*)$, ktoré z hľadiska nášho modelu považujeme za nové pozorovanie realizácie gaussovského poľa.

   5.5. Zväčšíme n na $n + 1$.

   5.6. Ukončíme cyklus.

6. Nájdeme hodnotu x, kde bola $f(x)$ najlepšia a táto hodnota je výstupom algoritmu.[4]

1.5 Akvizičná funkcia

V tejto sekcii si povieme o akvizičných funkciách, ktoré boli spomenuté v algoritme Bayesovskej optimalizácie. Akvizičná funkcia sa používa na konštrukciu účelovej funkcie z aposteriórneho rozdelenia, ktoré nám umožňuje určiť ďalší bod na evaluáciu. Výber vhodnej akvizičnej funkcie je veľmi dôležitý, pretože jej voľba môže zásadne ovplyvniť náš cieľ - nájsť extrém účelovej funkcie na čo najmenší počet iterácií.

Vo všeobecnosti keď konštuujeme algoritmus na nekonvexnú, multimodálnu optimalizáciu, mali by sme sa snažiť brať do úvahy takzvanú exploitáciu, ale aj takzvanú exploráciu.

Exploitácia znamená snahu využívať známu informáciu na okamžitý výber takých bodov, v ktorých očakávame vysokú (pri maximalizačnom probléme), alebo nízku (pri minimalizačnom probléme) hodnotu účelovej funkcie.

Naopak, explorácia je snaha o prehľadávanie takých častí základného priestoru X, ktoré nám poskytnú čo najviac informácie potenciálne užitočnej z dlhodobejšieho, „strategického" hľadiska. Poznamenajme, že bežne používané akvizičné funkcie sa zdanlivo

[4]Hodnotu x vyberáme spomedzi všetkých tých hodnôt x, v ktorých sme v priebehu algoritmu evaluovali f.

zameriavajú na exploitáciu, nie na exploráciu. Avšak v oblastiach, ktoré v danom kroku algoritmu ešte neboli dostatočne prehľadávané, narastá neurčitosť ohľadom hodnôt účelovej funkcie, čo sa prenesie do vyššej hodnoty akvizičnej funkcie. Tento efekt zabezpečuje exploračnú zložku Bayesovskej optimalizácie.

Medzi najpoužívateľnejšie akvizičné funkcie patrí pravdepodobnosť zlepšenia (angl. *probability of improvement*), očakávané zlepšenie (angl. *expected improvement*) a horná hranica spoľahlivosti (angl. *upper confidence bound*).

1.5.1 Pravdepodobnosť zlepšenia

Označíme si $f' = \min f$ ako minimálnu hodnotu f, ktorú sme zatiaľ dostali. Ide nám o nájdenie minimálnej hodnoty, teda čo najmenšej. Stratégia konštrukcie akvizičnej funkcie je v tomto prípade nájdenie takého bodu x^*, v ktorom nám momentálne aposteriórne rozdelenie gaussovského procesu poskytuje najvyššiu pravdepodobnosť dosiahnutia lepšej (čiže menšej) funkčnej hodnoty než je f'.

To zodpovedá nasledujúcej úžitkovej funkcii, kde vypočítame f v bode x:

$$u(x) = \begin{cases} 0 & f(x) > f' \\ 1 & f(x) \leq f'. \end{cases} \tag{3}$$

Inak povedané dostaneme jednotkovú odmenu, ak $f(x)$ je menšia ako f' a žiadnu odmenu, ak je väčšia.

Pravdepodobnosť zlepšenia akvizičnej funkcie je potom očakávaná funkcia užitočnosti od x:

$$a_{PI}(x) = \int_{-\infty}^{f'} \mathcal{N}(y; \mu(x), \kappa(x,x))\ dy = \phi(f'; \mu(x), \kappa(x,x)), \tag{4}$$

kde $\mathcal{N}$ je hustota normálneho rozdelenia s parametrami $\mu(x)$ a $\kappa(x,x)$, f je argument tejto hustoty, kde $\mu(x)$ a $\kappa(x,x)$ sú stredná hodnota a rozptyl aposteriórneho rozdelenia daného gaussovského procesu vzhľadom na všetky dosiaľ evaluované hodnoty účelovej funkcie (chápané ako získané pozorovania daného náhodného procesu) a ϕ je distribučná funkcia uvedeného aposteriórneho rozdelenia v bode x.

Vyberie sa teda bod na ďalšie pozorovanie s najvyššou pravdepodobnosťou zlepšenia, teda s maximálnou očakávanou užitočnosťou od pôvodnej najlepšej hodnoty.

1.5.2 Očakávané zlepšenie

Pri pravdepodobnosti zlepšenia sme dostali odmenu za zlepšenie súčasného minima nezávisle od veľkosti zlepšenia. To môže niekedy viesť k situácii, že uviazneme v lokálnom optime a nedostatočne preskúmame celý priestor X globálne. Pri tomto prípade by teda prevažovala exploitácia nad exploráciou.

Je však možné skonštruovať aj takú akvizičnú funkciu, ktorá berie do úvahy nie len pravdepodobnosť zlepšenia, ale aj magnitúdu zlepšenia. Tú opíšeme v tejto časti práce.

Akvizičná funkcia, ktorá sa pozerá na pravdepodobnosť zlepšenia a aj veľkosť zlepšenia sa nazýva očakávané zlepšenie. Znovu si označíme f' ako minimálnu hodnotu doposiaľ nameranú na f. Očakávané zlepšenie evaluuje f v bode, kde sa očakáva najlepšie zlepšenie na f, čo môžeme zapísať pomocnou úžitkovou funkciou vyjadrujúca „úžitok" z merania v bode x:

$$u(x) = \max(0; f' - f(x)). \tag{5}$$

V tomto prípade dostaneme odmenu o veľkosti zlepšenia, ak nová hodnota $f(x)$ je menšia ako pôvodná najlepšia hodnota f', inak nezískame žiadnu odmenu. Očakávané zlepšenie vieme zapísať nasledovne:

$$\begin{aligned} a_{EI}(x) &= \int_{-\infty}^{f'} (f' - y)\mathcal{N}(y; \mu(x), \kappa(x,x))\ dy \\ &= (f' - \mu(x))\phi(f'; \mu(x), \kappa(x,x)) + \kappa(x,x))\phi(f'; \mu(x), \kappa(x,x)). \end{aligned} \tag{6}$$

Interpretácia jednotlivých symbolov je rovnaká ako vo vzťahu (4).

Následne bod, ktorý má najväčšie očakávané zlepšenie, je vybraný ako ďalší bod na evaluáciu.

Očakávané zlepšenie má dve zložky. Prvú možno zvýšiť znížením funkcie strednej hodnoty $\mu(x)$. Druhú možno zvýšiť zväčšením variancie $\kappa(x,x)$. Tieto dve zložky môžeme interpretovať ako „rovnováhu" medzi hodnotením v bodoch s nízkou strednou hodnotou (exploitácia) a hodnotením v bodoch s vysokou neistotou (explorácia).

1.5.3 Horná hranica spoľahlivosti

Horná hranica spoľahlivosti sa väčšinou využíva pri maximalizovaní f, zadefinujeme si akvizičnú funkciu pre minimalizáciu f:

$$a_{UCB}(x;\beta) = \mu(x) - \beta\kappa(x,x),$$

kde parameter β slúži na vyrovnávanie pomeru medzi exploitáciou a exploráciou. Akvizičná funkcia horná hranica spoľahlivosti je pri minimalizácii rozdielom strednej predikcie a neistoty. Symboly $\mu(x)$ a $\kappa(x,x)$ majú význam ako vyššie, čiže ide o strednú hodnotu a rozptyl aposteriórneho rozdelenia odvodeného z apriórneho rozdelenia uvažovaného gaussovského procesu a všetkých dosiaľ „pozorovaných" hodnôt tohto procesu.

1.6 Optimalizácia Rosenbrockovej funkcie

V tejto časti si na jednoduchom príklade ukážeme, ako Bayesovská optimalizácia efektívne nájde hľadaný extrém funkcie na menší počet iterácií ako gradientné metódy. V našom prípade budeme hľadať minimum Rosenbrockovej funkcie. Využijeme naprogramované metódy v programovacom prostredí *R-Studio* a porovnáme výsledky a počty evaluácií oproti gradientým metódam (BFGS, Nelder-Mead).

Predpis Rosenbrockovej funkcie vyzerá nasledovne:

$$f(x,y) = 100(y-x^2)^2 + (1-x)^2. \tag{7}$$

Rosenbrockova funkcia má minimum v bode [1,1] a jej funkčná hodnota je v tomto bode rovná 0. Všetky výpočty a vizualizácie v tejto práci sme realizovali v programovacom jazyku *R-Studio* [11]. Za pomoci funkcie *optim* v programovacom prostredí *R-Studio* sme optimalizovali Rosenbrockovu funkciu s predpisom (7) metódami BFGS a Nelder-Mead a našli sme minimum v bode [1,1] a príslušnú funkčnú hodnotu rovnú 0.

Následne pomocou balíčka *rBayesianOptimization* a funkcie *BayesianOptimization* v tomto balíčku [18] sme implementovali Bayesovskú optimalizáciu na Rosenbrockovu funkciu. Do tejto optimalizácie sme zadali hranice, v ktorých má hodnoty prehľadávať, na intervaly [-2,2] pre obe súradnice. Akvizičnú funkciu sme použili hornú hranicu spoľahlivosti z časti 1.5.3, kde parameter β sme nastavili na hodnotu 2.576, ktorá sa používa ako predvolená hodnota tohto parametra. V tomto balíčku sa značí tento parameter ako

kappa a jeho účelom je vyrovnávať exploráciu a exploitáciu. S rastúcou hodnotou parametra kappa sa bude viac sústrediť na exploráciu. Zadali sme do optimalizácie 10 počiatočných bodov funkcie, ktoré si algoritmus sám náhodne vyberie a vypočíta ako začiatok optimalizácie. Následne po nájdení počiatočných bodov vypočíta algoritmus 20 iterácii. Výsledok, ktorý našla táto metóda a priebeh Bayesovskej optimalizácie môžeme vidieť na Obr. 3.

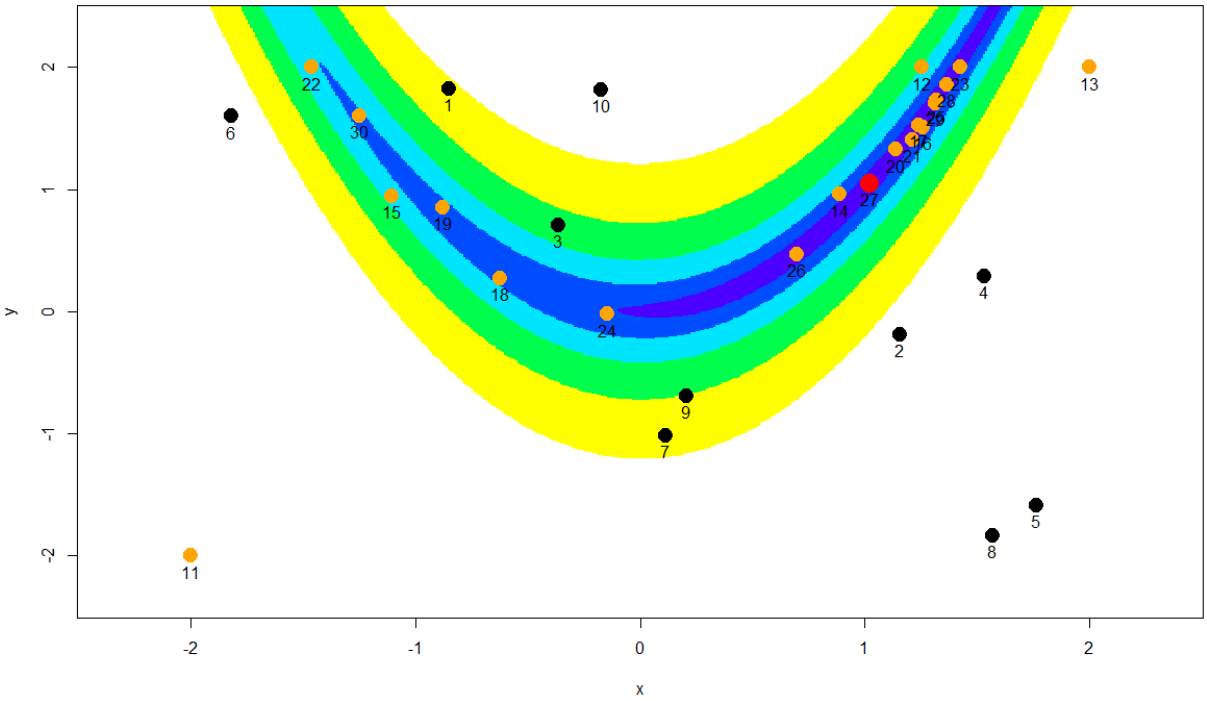
```
> bayes <- BayesianOptimization(FUN = objective,
+                               bounds = search_bound,
+                               init_grid_dt = NULL,
+                               init_points =10,
+                               n_iter = 20,
+                               kappa = 2.576,
+                               acq = "ucb" )
elapsed = 0.00  Round = 1       x1 = -0.8496899 x2 = 1.827333   Value = -125.6035
elapsed = 0.00  Round = 2       x1 = 1.153221   x2 = -0.1866634 Value = -230.0253
elapsed = 0.00  Round = 3       x1 = -0.3640923 x2 = 0.7102825  Value = -35.23671
elapsed = 0.00  Round = 4       x1 = 1.53207    x2 = 0.2905336  Value = -423.2861
elapsed = 0.00  Round = 5       x1 = 1.761869   x2 = -1.588301  Value = -2202.5212
elapsed = 0.00  Round = 6       x1 = -1.817774  x2 = 1.5993     Value = -298.6432
elapsed = 0.00  Round = 7       x1 = 0.112422   x2 = -1.015649  Value = -106.5254
elapsed = 0.00  Round = 8       x1 = 1.569676   x2 = -1.831762  Value = -1845.5813
elapsed = 0.00  Round = 9       x1 = 0.2057401  x2 = -0.6883171 Value = -54.01522
elapsed = 0.00  Round = 10      x1 = -0.1735411 x2 = 1.818015   Value = -321.0352
elapsed = 0.00  Round = 11      x1 = -2.0000    x2 = -2.0000    Value = -3609.0000
elapsed = 0.00  Round = 12      x1 = 1.254096   x2 = 2.0000     Value = -18.31814
elapsed = 0.00  Round = 13      x1 = 2.0000     x2 = 2.0000     Value = -401.0000
elapsed = 0.00  Round = 14      x1 = 0.8849292  x2 = 0.9611795  Value = -3.184483
elapsed = 0.00  Round = 15      x1 = -1.106249  x2 = 0.9492474  Value = -11.97343
elapsed = 0.00  Round = 16      x1 = 1.256237   x2 = 1.507832   Value = -0.5598483
elapsed = 0.00  Round = 17      x1 = 1.240494   x2 = 1.526945   Value = -0.07195251
elapsed = 0.00  Round = 18      x1 = -0.6250644 x2 = 0.2724529  Value = -4.039204
elapsed = 0.00  Round = 19      x1 = -0.8790817 x2 = 0.8522588  Value = -4.162561
elapsed = 0.00  Round = 20      x1 = 1.138636   x2 = 1.32713    Value = -0.1130952
elapsed = 0.00  Round = 21      x1 = 1.212658   x2 = 1.410623   Value = -0.4042329
elapsed = 0.00  Round = 22      x1 = -1.462466  x2 = 2.0000     Value = -7.990431
elapsed = 0.00  Round = 23      x1 = 1.424819   x2 = 2.0000     Value = -0.2711324
elapsed = 0.00  Round = 24      x1 = -0.1446696 x2 = -0.02030091        Value = -1.480261
elapsed = 0.00  Round = 25      x1 = 1.316896   x2 = 1.726631   Value = -0.106175
elapsed = 0.00  Round = 26      x1 = 0.696848   x2 = 0.4685777  Value = -0.1208671
elapsed = 0.00  Round = 27      x1 = 1.021958   x2 = 1.05115    Value = -0.005041525
elapsed = 0.00  Round = 28      x1 = 1.366199   x2 = 1.861539   Value = -0.1365636
elapsed = 0.00  Round = 29      x1 = 1.3147     x2 = 1.708316   Value = -0.1395136
elapsed = 0.00  Round = 30      x1 = -1.25069   x2 = 1.604956   Value = -5.231501

 Best Parameters Found:
Round = 27      x1 = 1.021958   x2 = 1.05115    Value = -0.005041525
```

Obr. 3: Výsledok a priebeh optimalizácie Rosenbrockovej funkcie pomocou Bayesovskej optimalizácie, kde *searchbound* predstavuje hranice, v ktorých hľadal algoritmus hodnoty x a y (na obrázku súradnice x=x1, y=x2), *initpoints* predstavuje počet počiatočných bodov a *niter* počet iterácií algoritmu, *acq* predstavuje akvizičnú funkciu a typ *ucb* predstavuje hornú hranicu spoľahlivosti a kappa je parameter pri tejto akvizičnej funkcii. *Value* predstavuje funkčnú hodnotu maximalizovanej zápornej funkcie v nájdenom bode.

Z Obr. 3 vidíme, že algoritmus Bayesovskej optimalizácie sa dopracoval k približne

optimálnemu výsledku. Najlepšie hodnoty sa blížili k optimálnej hodnote tejto funkcie. Najlepšie hodnoty, ktoré našla Bayesovská optimalizácia, je bod [x,y] = [1.021958, 1.05115] a funkčná hodnota maximalizovanej zápornej funkcie $Value$ = -0.005041525 v tomto konkrétnom bode.

Na Obr. 4 sme zobrazili Rosenbrockovu funkciu a priebeh samotných iterácii Bayesovskej optimalizácie. Všimnime si, že väčšina z tých bodov, ktorých polohu počíta algoritmus z princípu Bayesovskej optimalizácie (oranžové body), je práve v modrom „údolí" nízkych hodnôt účelovej funkcie, v ktorom sa nachádza aj optimálna hodnota. Zároveň si môžeme všimnúť, že pri niektorých iteráciach algoritmus volí bod evaluácie účelovej funkcie mimo tohto údolia a skúša hľadať, či v inej oblasti nebude ešte lepšia hodnota účelovej funkcie. Tento jav predstavuje spomínanú exploračnú zložku, kde algoritmus skúma aj v oblastiach, ktoré v danom kroku algoritmu ešte neboli dostatočne prehľadávané.



Obr. 4: Zobrazenie hodnôt algoritmu Bayesovskej optimalizácie na Rosenbrockovej funkcii. Čiernymi bodkami je znázornených 10 štartovacích bodov, ktoré sme zadali do algoritmu a následne oranžovými bodkami sú znázornené postupné iterácie algoritmu. Červený bod predstavuje najlepšiu hodnotu, ktorú našla Bayesovská optimalizácia.

Pomocou všetkých použitých metód sa nám podarilo dosiahnuť žiadaného výsledku, avšak metódy BFGS a Nelder-Mead potrebovali viackrát evaluovať funkciu, aby dospeli

k podobnému výsledku.

Pre porovnanie na výpočet optima pri Bayesovskej optimalizácii sme dospeli k blízkemu výsledku za pomoci algoritmu, ktorý si náhodne vybral a vypočítal 10 zadaných bodov a vypočítal 20 iterácií. Pri metóde BFGS sme museli až 110-krát vyrátať funkčnú hodnotu a 43-krát vypočítať gradient funkcie. Pri metóde Nelder-Mead sme museli vyrátať až 195-krát funkčnú hodnotu na nájdenie optimálnej funkčnej hodnoty a bodov, kde je minimum. Bayesovská optimalizácia teda na menší počet iterácií dokázala nájsť približne rovnaké riešenie.

2 Metóda oporných bodov

V tejto kapitole si predstavíme základné definície metódy oporných bodov (angl. *support vector machine*) v strojovom učení a princíp, na akom táto metóda funguje. Taktiež v tejto časti riešime problém ako pomocou binárneho klasifikátora (metóda oporných bodov je vo svojej prirodzenej verzii binárna klasifikačná metóda) vytvoriť multikategorický klasifikátor. Tento problém budeme riešiť na dátovom súbore *Pen-Based Recognition of Handwritten Digits*, ktorý si predstavíme bližšie v časti 3.1. Budeme vychádzať najmä zo zdrojov [5], [6], [13], [15] a vybraných poznámkam k prednáškam [4], a [16].

2.1 Metódy strojového učenia

Strojové učenie je jeden zo spôsobov, akým sa snažíme dosahovať „umelú inteligenciu", ktorá poskytuje systému schopnosť automaticky sa učiť a zlepšovať sa, na základe existujúcich príkladov z minulosti alebo z vlastných skúseností. Hlavný zámer umelej inteligencie je, aby sa počítač dokázal učiť samostatne, bez ľudskej asistencie.

Na základe toho, ako prebieha proces učenia, môžeme algoritmy strojového učenia rozdeliť do nasledujúcich troch skupín:

- Učenie s učiteľom (angl. *Supervised machine learning*) - používa sa na učenie z datasetu, ktorý obsahuje tzv. dáta s označeniami (angl. *labeled data*). Dataset je súbor cvičných dát, z ktorých sa funkcia učí vzory a súvislosti v dátach. Predpokladom pre jeho použitie je existencia dostatočne veľkého trénovacieho datasetu, ktorý obsahuje správne označené dvojice vstup – výstup. V datasete na rozdiel od učenia bez učiteľa už máme dáta vopred klasifikované a vieme do akých kategórií patria. Na týchto dátach sa potom algoritmus natrénuje tak, aby vedel pre daný vstup vyprodukovať správny výstup. Učenie s učiteľom je závislé na dostatočnom množstve kvalitných dát. Medzi hlavné problémy, ktoré pomocou neho vieme riešiť, sú klasifikačné a predikčné regresné problémy.

- Učenie bez učiteľa (angl. *Unsupervised machine learning*) - používa sa pri dátach, ktoré neboli vopred klasifikované. Chýba nám teda „správna odpoveď“, ktorou bolo v príklade pri učení s učiteľom označenie, aké sú skutočné (správne) kategórie objektov v databáze. Môžeme mať totiž prípady, kedy túto správnu odpoveď nie je

možné odpozorovať alebo ju ani sami nevieme jednoznačne určiť. Algoritmy učenia bez učiteľa sa potom v týchto neklasifikovaných dátach snažia objaviť, modelovať a určiť vzory, s cieľom dozvedieť sa o týchto dátach niečo viac. Algoritmus teda nepríde na správny výstup, ale preskúma dáta a určí skryté štruktúry v týchto neoznačených dátach. Najčastejším problémom, ktoré rieši učenie bez učiteľa, je zhlukovanie. Úlohou zhlukovania je spájať dáta do skupín objektov, ktoré „majú niečo spoločné“.

- Učenie formou posilňovania (angl. *Reinforcement learning*) - učenie prebieha tak, že vytvoríme systém tzv. agenta, ktorého nasadíme do prostredia a necháme ho nech sa učí prostredníctvom interakcie s prostredím. Jediné čo agentovi musíme určiť, sú pravidlá, ako sa v danom prostredí môže správať a takzvanú odmeňovaciu funkciu. Pomocou nej vie agent vyhodnotiť, či rozhodnutie, ktoré práve vykonal bolo preňho prospešné. Následne metódou pokus-omyl podobne ako človek skúša jednotlivé možnosti a naučí sa, ako sa má ideálne správať v jednotlivých situáciách.

2.2 Metóda oporných bodov

V tejto časti si predstavíme hlavnú myšlienku a algoritmus metódy oporných bodov. Vysvetlíme si pojem maximálnej odchýlky, pomocou ktorej definujeme tzv. oporné body. Pre jednoduchosť sú v tejto práci definované binárne klasifikátory [5], o multikategorických klasifikátoroch si bližšie povieme v časti 2.6.

Metóda oporných bodov (angl. *support vector machines*), skrátene SVM, je metódou strojového učenia, ktorá patrí do kategórie učenia s učiteľom a využíva sa najmä na klasifikáciu a regresiu. V tejto práci budeme využívať metódu SVM na klasifikačné problémy.

Cieľom SVM je, zhruba vyjadrené, nájsť separačnú varietu (vo všeobecnosti nelineárnu) v priestore, ktorá by oddeľovala vstupné dáta na dve triedy tak, že všetky dáta jednej triedy sú na jednej strane tejto variety a všetky dáta druhej triedy sú na druhej strane variety a aby táto varieta bola čo najďalej od najbližšieho dátového bodu z ktorejkoľvek triedy.

[5] Existujú aj viactriedne klasifikátory, ale základný princíp metódy oporných bodov sa týka binárnej klasifikácie.

2.3 Lineárna klasifikácia SVM

Pre vstupné dáta do SVM klasifikácie budeme brať tréningovú vzorku so vstupnými vektormi $x_1, x_2, ..., x_n \in \mathcal{R}^p$ a príslušnými klasifikáciami $c_1, c_2, ..., c_n \in \{+1, -1\}$ také, že množiny $C_- := \{x_i : c_i = -1\}$ a $C_+ := \{x_i : c_i = +1\}$ budú neprázdne.

Nadrovina v k-rozmernom priestore je afinný podpriestor dimenzie $k-1$, ktorý definujeme ako množinu bodov $(x_1, ..., x_k)$, ktorá spĺňa rovnosť:

$$\beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_k x_k = 0,$$

pre nejaké parametre $\beta_0, \beta_1, ..., \beta_k \in \mathcal{R}$, pričom aspoň jedna z hodnôt $\beta_1, ..., \beta_k$ je nenulová.

Keby prvky trénovacej množiny boli lineárne separovateľné, teda by sme ich vedeli rozdeliť pomocou oddeľovacej nadroviny, čo vieme ekvivaletne zapísať ako:

$$c_i = -1 \Rightarrow \beta_0 + \beta_1^T x_i < 0,$$
$$c_i = +1 \Rightarrow \beta_0 + \beta_1^T x_i > 0,$$

pre všetky *i=1,...,n* a $\beta = (\beta_0, \beta_1^T)^T \in \mathcal{R}^{p-1}, \beta_1 \neq 0_p$.

Potom existuje nadrovina

$$H_\beta^0 := \{x \in \mathcal{R}^p : \beta_0 + \beta_1^T x = 0\},$$

ktorá oddeľuje C_- a C_+. Takáto separačná nadrovina je prirodzenou rozhodovacou hranicou (angl. *decision boundary*) medzi C_- a C_+, ktorú chceme skonštruovať.

Všimnime si však, že v lineárne separovateľnom prípade existuje celé spektrum separačných nadrovín. Základným princípom SVM klasifikácie je klasifikácia pomocou separačnej nadroviny, ktorá má najväčšiu vzdialenosť od dátových bodov [6].

Nájsť separačnú nadrovinu, ktorá maximalizuje vzdialenosť od dátových bodov, znamená riešiť špeciálny optimalizačný problém na množine všetkých dostupných separačných nadrovín. Keď je nové pozorovanie klasifikované, čím ďalej leží od rozhodovacej hranice, tým sme si istejší, že pozorovanie je klasifikované správne. Pre každého kandidáta na nadrovinu sa vypočítajú kolmé vzdialenosti k bodom trénovacích dát. Najmenšia

[6] Je prirodzené, že čím je klasifikovaný dátový bod ďalej od optimálnej separačnej nadroviny, tým je klasifikácia spoľahlivejšia. To je možné využiť na priradenie číselnej miery dôvery v správnosť klasifikácie (čiže na získanie detailnejšej informácie akou je len informácia o triede, do ktorej klasifikátor daný dátový bod zaraďuje).

z týchto vzdialeností sa nazýva *hraničná vzdialenosť* (angl. *margin*), ktorú si zadefinujeme ako:

$$d_\beta := \min_{i=1,\dots,n} \rho(H^0_\beta, x_i),$$

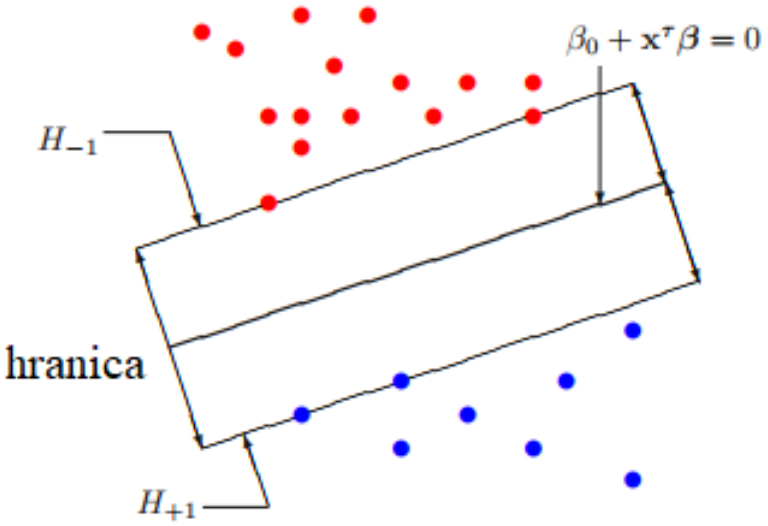
kde $\rho(H^0_\beta, x_i)$ je Euklidovská vzdialenosť medzi H^0_β a x_i.

Následne je zvolené optimálne β^*, ktoré maximalizuje d_β pri existencii oddeľovacej nadroviny H^0_β. Toto úplne definuje problém optimalizácie, ktorý musíme vyriešiť.

Pozorovania najbližšie k nadrovine, ktoré definujú hraničnú vzdialenosť sa nazývajú oporné vektory (angl. *support vectors*). Bod x_{i^*} budeme rozumieť ako oporný bod, ak spĺňa:

$$\rho(H^0_{\beta^*}, x_{i^*}) = \min_{i=1,\dots,n} \rho(H^0_{\beta^*}, x_i).$$

Ilustráciu lineárne separovateľného prípadu spolu s opornými vektormi a oddeľovacou nadrovinu môžeme vidieť na Obr. 5.



Obr. 5: Príklad lineárne separovateľného prípadu. Červené body zodpovedajú dátovým bodom s $c_i = -1$ a modré body zodpovedajú dátovým bodom s $c_i = +1$. Oddeľujúcou nadrovinou je priamka $\beta_0 + \beta_1^T x = 0$. Oporné vektory sú tie body, ktoré ležia na nadrovinách H_{-1} a H_{+1}. Zdroj obrázku: [6].

Pretože $H^0_{\beta^*}$ je oddeľujúca nadrovina, ktorá má najväčšiu vzdialenosť od oporných bodov medzi dvoma klasifikáciami, tak sa nachádza priamo v strede medzi opornými

vektormi. To znamená, že existuje $\epsilon > 0$ také, že spĺňa

$$\begin{aligned} c_i = -1 \Rightarrow \beta_0^* + (\beta_1^*)^T x_i \leq -\epsilon, \\ c_i = +1 \Rightarrow \beta_0^* + (\beta_1^*)^T x_i \geq \epsilon, \end{aligned} \tag{8}$$

pre všetky $i = 1, ..., n$. Je tiež zrejmé, že

$$d_{\beta^*} := \min_{i=1,...,n} \frac{|\beta_0^* + (\beta_1^*)^T x_i|}{||\beta_1^*||} = \frac{\epsilon}{||\beta_1^*||}. \tag{9}$$

Teda v našom prípade aby sme našli nadrovinu s najväčšou hraničnou vzdialenosťou potrebujeme nájsť β^* a $\epsilon > 0$ také, ktoré budú maximalizovať (9) a spĺňať podmienky (8).

Ak je riešením (β^*, ϵ), potom aj $(\frac{\beta^*}{\epsilon}, 1)$ bude tiež riešením pre rovnaký problém. Zafixujeme si $\epsilon = 1$ a našou novou úlohou bude maximalizovať $\frac{1}{||\beta^*||}$. Tento maximalizačný problém vieme prepísať na maximalizačnú úlohu, kde budeme maximalizovať

$$\frac{1}{2}||\beta||^2,$$

následne podmienky (8) môžeme zapísať dokopy ako

$$c_i(\beta_0 + \beta^T x_i) \geq 1.$$

Preto na vypočítanie nadroviny s najväčšou hraničnou vzdialenosťou, stačí vypočítať riešenie β^* nasledujúceho problému lineárneho kvadratického programovania

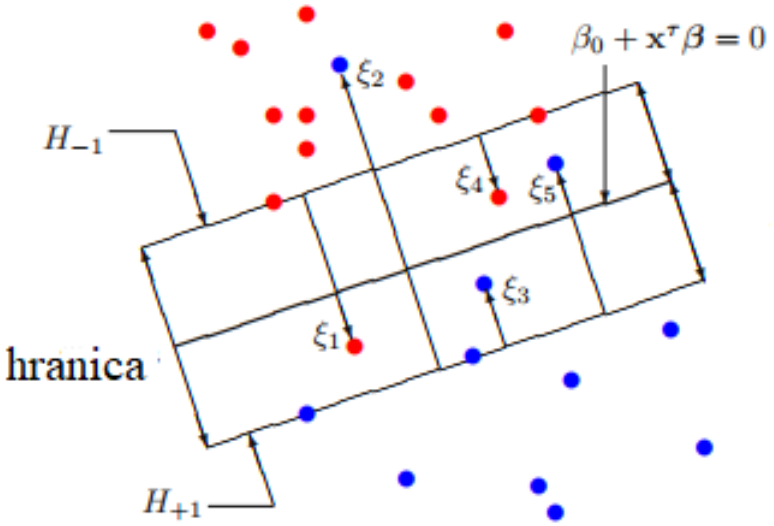
$$\begin{aligned} &\min_{\beta_0 \in \mathcal{R}, \beta \in \mathcal{R}^p} \frac{1}{2}||\beta||^2 \\ &\text{st. } c_i(\beta_0 + (\beta)^T x_i) \geq 1, \\ &\qquad i = 1, ..., n. \end{aligned} \tag{10}$$

Optimalizačný problém (10) predstavuje úlohu kvadratického programovania, pretože cieľová funkcia je kvadratická v neznámych a všetky obmedzenia sú lineárne v neznámych. Kvadratická úloha má jedno globálne minimum, ktoré možno efektívne nájsť so súčasnými optimalizačnými balíkmi. Lineárny klasifikátor, ktorý dostaneme v tomto prípade pre lineárne separovateľné triedy sa nazýva *klasifikátor maximálnej vzdialenosti.*

2.4 Klasifikácia metódou oporných bodov

Väčšina dátových súborov avšak nebude lineárne separovateľná. V prípade, ak by C_- a C_+ neboli lineárne separovateľné, je potrebné zmierniť podmienky, aby sa umožnila nesprávna klasifikácia v niektorých dátových bodoch v trénovacej sade. Metóda SVM v

tejto situácii rieši analogický optimalizačný problém ako v prípade lineárne separovateľných dát. V tomto upravenom probléme sú však lineárne ohraničenia zmenené tak, aby boli splniteľné. To vieme dosiahnuť pridaním nezáporných premenných tzv. *penalizačnou premennou* (angl. *slack variable*) $\xi = (\xi_1, \xi_2, ..., \xi_n)$. Tieto premenné, jedna pre každý dátový bod, reprezentujú „mieru chyby" klasifikácie daného dátového bodu. Zároveň sa tieto premenné včlenia do účelovej funkcie takým spôsobom, aby sa pri optimalizovanie účelovej funkcie samy minimalizovali.



Obr. 6: Príklad lineárne separovateľného prípadu. Červené body zodpovedajú dátovým bodom s $c_i = -1$ a modré body zodpovedajú dátovým bodom s $c_i = +1$. Oddeľujúcou nadrovinou je čiara $\beta_0 + \beta_1^T x = 0$. Oporné vektory sú tie body, ktoré ležia na nadrovinách H_{-1} a H_{+1}. Penalizačné premenné ξ_1 a ξ_4 sú spojené s červenými bodmi, ktoré porušujú obmedzenie nadroviny H_{-1}, a bodmi označené ξ_2, ξ_3 a ξ_5 sú spojené s modrými bodmi, ktoré porušujú obmedzenie nadroviny H_{+1}. Body, ktoré spĺňajú obmedzenia vhodnej nadroviny majú $\xi_i = 0$. Zdroj obrázku: [6].

Matematicky, nový optimalizačný problém môžeme zapísať ako

$$\begin{aligned} \min_{\beta_0 \in \mathcal{R}, \beta \in \mathcal{R}^p, \xi_1 \geq 0, \ldots, \xi_n \geq 0} & \frac{1}{2}||\beta||^2 + C\sum_{i=0}^{n} \xi_i \\ \text{st. } & c_i(\beta_0 + (\beta)^T x_i) \geq 1 - \xi_i, \\ & \xi_i \geq 0, \\ & i = 1, ..., n, \end{aligned} \tag{11}$$

kde C je parameter, ktorý predstavuje mieru penalizácie za nesprávne zaradenia bodov trénovacej sady dát. Menšie hodnoty parametra C zodpovedajú mäkšej hranici, ktorý dáva algoritmu väčšiu voľnosť pri ignorovaní niektorých nesprávnych zaradení dátových bodov. Veľké hodnoty parametra zabezpečujú, aby bol algoritmus v tomto smere prísnejší.

V tomto prípade optimálna β^* nebude definovať separačnú nadrovinu, lebo neexistuje žiadna taká nadrovina v lineárne neseparovateľnom prípade. Avšak nadrovina $H^0_{\beta^*}$ bude tvoriť rozumnú hranicu klasifikačných tried.

Tento optimalizačný problém z (11) má vždy realizovateľné riešenie a stále je to problém lineárneho kvadratického programovania, ktorý sa dá previesť na duálnu úlohu, ktorú spomenieme v ďalšej časti.

2.5 Nelineárna klasifikácia SVM a kernelový trik

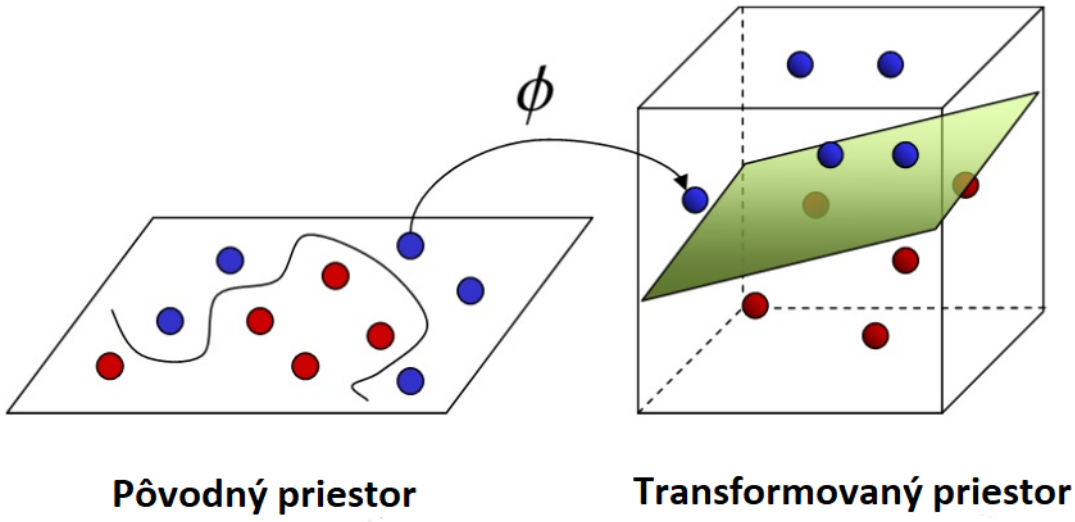
Doteraz metódy nájdenia oddeľujúcej nadroviny nám ponúkli iba možnosť zostrojiť lineárnu rozhodovaciu hranicu. Avšak vo väčšine prípadov skutočná rozhodovacia hranica medzi triedami nie je lineárneho charakteru.

SVM rieši tento problém nelineárnej klasifikácie za pomoci takzvaného jadrového triku (angl. *kernel trick*), ktorý bol prvýkrát použitý Cortesom a Vapnikom v roku 1995 [15].

Jadrový trik transformuje pôvodné dáta za pomoci vybranej funkcie $\phi(x)$ do viacrozmerného priestoru, kde sa klasifikácia spraví jednoduchšie ako v pôvodnom priestore. Táto trasformácia zvýši dimenziu našich dát.

Príklad transformácie môžeme vidieť na Obr. 7, kde z pôvodného nelineárne separovateľného priestoru sa transformáciou dostaneme do priestoru s väčšou dimenziou, kde už dáta vieme lineárne separovať.

Úlohu (11) môžeme vyriešiť ako úlohu konvexného programovania, kde Lagrangeova



Obr. 7: Transformácia pôvodného priestoru do nového priestoru s väčšou dimenziou pomocou jadrové triku ϕ. Zdroj obrázku: [17].

primárna úloha bude vyzerať nasledovne:

$$L_P = \frac{1}{2}||\beta||^2 + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\alpha_i[c_i(\beta_0 + (\beta)^T x_i) - (1-\xi_i)] - \sum_{i=1}^{n}\mu_i\xi_i, \tag{12}$$

kde $\alpha_i \geq 0$ a $\mu_i \geq 0$ sú Lagrangeove multiplikátory.

Primárny optimalizačný problém pre (12) vieme formulovať ako

$$\min_{\beta_0,\beta,\xi_i}\max_{\alpha_i,\beta,\xi_i} L_P. \tag{13}$$

Výraz (13) predstavuje konvexný problém kvadratického programovania, preto môžeme miesto toho riešiť duálny problém. Vypočítaním príslušných derivácií a položením týchto derivácií rovných nule dostaneme nasledujúce vzťahy:

$$\beta = \sum_{i=1}^{n}\alpha_i c_i x_i,$$
$$0 = \sum_{i=1}^{n}\alpha_i c_i,$$
$$\alpha_i = C - \mu_i, \forall i,$$
$$\alpha_i,\ \mu_i,\ \xi_i \geq 0\ \forall i.$$

Dosadením do pôvodnej primárnej Lagrangeovej úlohy vieme zostrojiť Lagrangeovu

duálnu funkciu s nasledujúcim tvarom:

$$L_D = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j c_i c_j \langle \phi(x_i)^T, \phi(x_j) \rangle.$$

Skutočne kľúčovým postrehom však je, že problém je založený iba na skalárnych súčinoch $\phi(x_i)$ a $\phi(x_j)$. Aj keď nepoznáme skutočné vektory, môžeme priamo numericky počítať tieto skalárne súčiny, a to bez toho, aby sme počítali vektory $\phi(x_i)$, dokonca bez toho, aby sme ϕ explicitne definovali. Na výpočet parametra β^* nadroviny tvoriacej rozhodovaciu hranicu medzi R_- a R_+ je potrebné poznať vektory $\phi(x_i)$. Avšak na praktické využitie nepotrebujeme poznať β^*, ale potrebujeme vedieť určiť či hodnota funkcie $f(x) = (\beta^*)^T x + \beta_0^*$ je kladná alebo záporná.

Hodnota funkcie

$$f(x) = (\beta^*)^T x + \beta_0^* = \sum_{i=0}^{n} c_i \alpha_i^* \langle \phi(x_i)^T, \phi(x) \rangle + c_j - \sum_{i=0}^{n} c_i \alpha_i^* \langle \phi(x_j)^T, \phi(x_i) \rangle \tag{14}$$

je plne určená skalárnymi súčinmi $\langle \phi(x_i)^T, \phi(x) \rangle$ a $\langle \phi(x_j)^T, \phi(x_i) \rangle$. Z toho vidíme, že na to, aby sme mohli použiť lineárnu triedu SVM v úlohe tejto pomocnej lineárnej triedy, musíme byť schopní vypočítať skalárne súčiny $\phi(x)$, $\phi(y)$ pre ľubovoľné $x, y \in \mathcal{R}^p$. V skutočnosti vôbec nemusíme explicitne počítať vektory $\phi(x)$, ale stačí nám len znalosť *funkcie jadra* (angl. *kernel function*).

Funkcia jadra

$$K(x, y) = \langle \phi(x), \phi(y) \rangle,$$

počíta skalárne súčiny v transformovanom priestore. Táto metóda tiež známa ako „kernel trik", výrazne urýchľuje proces učenia sa klasifikátora a umožňuje manipuláciu s vysokorozmerným priestorom.

Výber funkcie jadra je základnou úlohou pri nelineárnych SVM. Podľa teórie by sme si mali vybrať funkciu, vďaka ktorej sú triedy objektov čo najlepšie lineárne oddeliteľné vo vysokorozmernom priestore, do ktorého zobrazuje funkcia ϕ.

Vo všeobecnosti zvýšenie zložitosti funkcie jadra zvyšuje šancu na to, že transformovaný súbor dát bude lineárne oddeliteľný. Neodporúča sa však používať príliš zložité funkcie jadra. Okrem predĺženia času potrebného na trénovanie, používanie zložitých funkcií jadra môže viesť k tzv. pretrénovaniu (angl. *overfitting*). Pretrénovaniu sa dá predísť vhodným výberom jadra a výberom hyperparametrov, ktoré sú pri určitých jadrách.

Medzi najjednoduchšie jadro patrí lineárne jadro (angl. *linear kernel*), ktoré je dané skalárnym súčinom $\langle x, y\rangle$ plus voliteľná konštanta c:

$$K(x, y) = x^T y + c.$$

Funkcie jadra využívajúce lineárne jadro sú často ekvivalentné lineárnej klasifikácii bez jadra.

Ďalším typom jadra je Gaussovo jadro (angl. *Gaussian kernel*), alebo tiež známe ako (angl. *Radial Basis Function*). Funkcia Gaussovho jadra pre dva body x a y vypočítava podobnosť alebo ako blízko sú pri sebe tieto dva body. Toto jadro možno matematicky zapísať nasledovne

$$K(x, y) = \exp(-\frac{||x - y||^2}{2\sigma^2}),$$

kde σ predstavuje rozptyl a zároveň hyperparameter pre Gaussovo jadro a $||x - y||^2$ je Euklidovská norma medzi dvomi bodmi x a y.

Ďalším jadrom je polynomiálne jadro (angl. *polynomial kernel*), ide o zovšeobecnenie lineárneho jadra. Predpis polynomiálneho jadra vyzerá nasledovne:

$$K(x, y) = (k\langle x, y\rangle + c)^d, \tag{15}$$

kde k, c, d sú hyperparametre polynomiálneho jadra.

Hyperparameter k sa nazýva aj mierka (angl. *scale*), ktorý slúži na preškálovania a predstavuje pohodlný spôsob normalizácie bez potreby úpravy samotných údajov. Hyperparameter c sa nazýva posun (angl. *offset*), ktorý pre prípad $c = 0$ vytvára takzvané homogénne polynomiálne jadro. Hyperparameter d sa nazýva stupeň (angl. *degree*), ktorý je vždy kladný a predstavuje stupeň polynómu.

Tieto hyperparametre polynomiálneho jadra budeme za pomoci Bayesovskej optimalizácie optimalizovať za účelom získania kvalitného klasifikátora pre klasifikáciu čislic od 0 do 9 (viac v kapitole 3).

2.6 Prevod multikategorickej klasifikácie na binárnu

Štandardný SVM klasifikátor je naučený rozlišovať medzi dvoma triedami. Ak dáta obsahujú viacero tried, musíme zvoliť iný prístup klasifikácie dát. Existujú dva používané prístupy.

Jeden všeobecný prístup k aplikácii metódy binárnej klasifikácie na problém s $K > 2$ triedami sa nazýva jeden na jedného (angl. *one vs. one*): zostavíme súbor $\binom{K}{2}$ binárnych klasifikátorov medzi všetkými pármi tried. Nové pozorovanie x je klasifikované všetkými binárnymi flasifikátormi zo súboru. Klasifikácia x sa potom vyberie ako tá, ktorá sa najčastejšie vyskytuje vo výsledkoch týchto binárnych klasifikácií.

Druhý prístup porovnávania sa nazýva jeden voči ostatným (angl. *one vs. rest*) , kde by klasifikátor určil pre každú triedu samostatne, či sa jedná o tú jednu triedu, alebo inú. Vyberie sa jedna trieda a nájde sa separačná varieta, ktorá túto triedu oddeľuje od ostatných tried. Procedúra sa opakuje pre každú triedu, výsledkom čoho je jeden model SVM na triedu. Nové pozorovania sú klasifikované podľa toho, ktorý model je z klasifikácie „najistejší“. Úroveň istoty je reprezentovaná vzdialenosťou dátového bodu k oddeľujúcej variete. V tejto práci používame prístup jedného voči všetkým.

2.7 Mriežkové prehľadávanie

Mriežkové prehľadávanie (angl. *Grid search*) je proces, ktorý pomáha pri výbere hyperparametrov. Na začiatku sa manuálne nastavia možné hodnoty hyperparametrov, teda určíme hranice, v ktorých pre každý parameter sa zadá konečná, obyčajne artimetická postupnosť hodnôt, ktoré chceme prehľadávať. Body zodpovedajúce takto zadanému m-rozmernému hyperparametru tvoria v priestore R^m mriežku. Potom sa na trénovanie modelov používajú všetky možné kombinácie hyperparametrov. Vyberie sa model s najlepším výkonom (najmenšou chybovosťou klasifikácie) a kombinácia hyperparametrov tohto modelu sa považuje za najvhodnejšiu. Mriežkové prehľadávanie je sprevádzané krížovou validáciou, aby sa predišlo nadmernému trénovaniu (angl. *overfittingu*). Pre každú kombináciu hyperparametrov sa model trénuje k-krát v závislosti od výberu hodnoty k pri krížovej validácii, o ktorej si povieme v ďalšej časti. Výsledkom je mriežka s hodnotami parametrov a výslednou chybovosťou trénovania.

2.8 Krížová validácia

Krížová validácia (angl. *Crossvalidation*) je štatistická metóda, ktorá rozdelí náhodne počiatočný súbor údajov na k častí. Metóda má jeden parameter s názvom k, ktorý odkazuje na počet skupín, do ktorých sa má daná vzorka dát rozdeliť. Ako taký sa postup

často nazýva k-násobná krížová validácia. Krížová validácia je metóda, ktorá sa používa na vyhodnotenie modelov strojového učenia na vzorke údajov. Algoritmus sa potom ohodnotí tak, aby sa na trénovanie použilo $k-1$ častí a jedna časť z tréningového procesu sa používa na overenie modelu. Tento proces sa opakuje pre všetky časti, pričom sa vždy jedna k-tica dát ponechá na overenie modelu, zatiaľ čo sa model trénujete na zvyšku dát. Po k iteráciach sa výsledky spriemerujú a dostaneme výslednú chybovosť modelu.

3 Aplikácia Bayesovskej optimalizácie

V tejto kapitole sa budeme zaoberať aplikáciou Bayesovskej optimalizácie na výber vhodných hyperparametrov metódy SVM. Konkrétnym cieľom je nájsť efektívny klasifikátor rukou písaných cifier do tried 0,1,...,9, pričom za tréningové dáta sme zvolili súbor *Pen-Based Recognition of Handwritten Digits*, ktorý sme čerpali z repozitára [2].

3.1 Pen-Based Recognition of Handwritten Digits

Dátový súbor *Pen-Based Recognition of Handwritten Digits* sme získali z archívu [2], odkiaľ parafrázujeme opis dátového súboru.

Dátový súbor obsahuje údaje o rukou napísaných cifrách 0 - 9 na tablet s rozlíšením 500×500 pixelov, na ktorý napísalo 44 ľudí po 250 cifier v náhodnom poradí. Dátový súbor je rozdelený na trénovaciu a testovaciu sadu. V trénovacej sade sú údaje o cifrách napísaných 30 ľuďmi, v testovacej sade sú údaje o cifrách napísaných zvyšnými 14 ľuďmi.

Dáta boli získané z tabletu, kde zariadenie zaznamenalo každých 100 milisekúnd pre každú cifru hodnoty x-ovej a y-ovej súradnice. Hodnoty x-ovej a y-ovej súradnice autori normalizovali, aby boli invariantné na posun a veľkosť. Autori ďalej dáta preškálovali, aby z pôvodných hodnôt z intervalu $[0, 500]$ vznikli nové hodnoty, ktoré pochádzali z intervalu $[0, 100]$. Popri preškálovaní sa hodnoty x-ovej a y-ovej súradnice menili menej pri x-ovej súradnici v porovnaní s y-ovou súradnicou, pretože prirodzene číslice boli napísané viac do výšky ako do šírky. Dáta autori transformovali za pomoci algoritmu, ktorý používal jednoduchú lineárnu interpoláciu medzi pármi bodov, aby mohli reprezentovať číslice ako vektory s konštantnou dĺžkou. V dátovom súbore sú pre každú cifru zaznamenané hodnoty x-ovej a y-ovej súradnice pre 8 bodov. Body boli vybrané tak, aby boli rovnomerne vzdialené. Dátový súbor obsahuje 17 premenných, prvých 16 premenných sú kvantitatívne premenné o x-ovej a y-ovej súradnici 8 bodov a posledná premenná je kategóriálna premenná, ktorá nám hovorí o akú cifru ide.

V tejto práci sme pracovali iba s trénovacou sadou, ktorá obsahuje dáta od 14 ľudí po 250 cifier. V nej sa nachádza 7494 pozorovaní, čo je o 6 menej, ako by sme očakávali. V popise uvedenom v [2] nie je uvedené prečo tieto údaje chýbajú.

3.2 Vizualizácia dát za pomoci metódy hlavných komponentov

Na náš dátový súbor aplikujeme metódu strojového učenia, ktorá nám pomôže lepšie pochopiť dáta, prípadne znížiť dimenziu bez veľkej straty dôležitých informácií týkajúcich sa niektorej charakteristiky pôvodných premenných a pomôcť vizualizovať viacrozmerné dáta, ktoré by nám vo veľkom rozmere nedávali veľkú výpovednú hodnotu. V tejto podkapitole budeme vychádzať z [4], [6], [8] a [16].

Metóda hlavných komponentov (PCA) (angl. *principal component analysis*) je metódou strojového učenia bez učiteľa, ktorej základným cieľom je redukovať dimenziu mnohorozmených dát, pochopiť a identifikovať prečo a ako sú premenné navzájom korelované, t.j. ako sa navzájom ovplyvňujú a vizualizovať mnohorozmerné dáta do nižšej dimenzie bez podstatnej straty informácie o pôvodných dátach. Ak sú premenné navzájom korelované, možno rovnaký objem informácií vystihnúť menším počtom premenných – zníženie dimenzie.

PCA je metóda, ktorá umožňuje transformovať veľký počet korelovaných premenných na omnoho menší počet premenných, ktoré sú navyše navzájom nekorelované.

Vezmime si náhodný p-rozmerný vektor $X = (X_1, ..., X_p)^T$, ktorý má strednú hodnotu μ a kovariančnú maticu Σ. V prvom kroku je cieľom nájsť vektor $Y = (Y_1, ..., Y_p)^T$, ktorý je transformáciou vektora X, ktorého zložky budú navzájom nekorelované a ich variancia bude postupne klesať, čo môžeme matematicky zapísať ako

$$Var(Y_1) \geq Var(Y_2) \geq ... \geq Var(Y_p).$$

V druhom kroku zvolíme $k < p$ a z pôvodných $(Y_1, ..., Y_p)$ zoberieme iba $(Y_1, ..., Y_k)$, čím zmenšíme dimenziu dát. Redukcia dimenzie dát má zo štatistického pohľadu veľký význam. Umožňuje nám znížiť počet premenných potrebných na efektívny popis dát. Takto môžeme zanedbať smery, v ktorých majú dáta len malú varianciu a všímať si len tie s veľkou varianciou. V tomto prípade je teda nositeľom informácie rozptyl, ktorý sa snaží táto metóda maximalizovať.

Nové nekorelované premenné sa nazývajú *hlavné komponenty*, ktoré sú lineárnymi funkciami pôvodných premenných. Každú zložku Y_i vyjadríme prostredníctvom lineárnej kombinácie prvkov vektora X, teda

$$Y_i = \alpha_i^T X, \tag{16}$$

kde $\alpha_i^T = (\alpha_{1i}, \alpha_{2i}, ..., \alpha_{pi})$ je vektor konštánt. Pridaním podmienky pre normalizáciu

$$\alpha_i^T \alpha_i = 1 \tag{17}$$

a ak vektory α_i budú navzájom kolmé, potom celková transformácia je ortonormálna. Takže vzdialenosti v p-rozmernom euklidovskom priestore sú zachované. Tieto jednotkové ortognálne vektory, ak ich zvolíme ako vlastné vektory kovariančnej matice pôvodných dát, sa nazývaju hlavné komponenty a sú vlastnými vektormi kovariančnej matice pôvodných dát.

Prvý hlavný komponent Y_1 nájdeme tak, že α_1 položíme také, aby Y_1 malo čo najväčší možný rozptyl. Inými slovami, vyberieme α_1 so snahou maximalizovať varianciu $\alpha_i^T X$ za podmienky, že platí (17) a vektory α_i sú navzájom kolmé.

Štandardným postupom na maximalizáciu funkcie za určitých podmienok je využitie Lagrangeových multiplikátorov. Pre α_1 si zadefinujeme Lagrangeovu funkciu

$$L(\alpha_1) = \alpha_1^T \Sigma \alpha_1 - \lambda(\alpha_1^T \alpha_1 - 1),$$

kde λ predstavuje Lagrangeov multiplikátor.

Funkciu zderivujeme podľa α_1:

$$\frac{\partial L}{\partial \alpha_1} = 2\Sigma\alpha_1 - 2\lambda\alpha_1.$$

Deriváciu položíme rovnú nule a dostávame vzťah

$$2(\Sigma - \lambda I_p)\alpha_1 = 0, \tag{18}$$

kde I_p predstavuje jednotkovú maticu rozmeru $p \times p$.

Aby táto rovnica zo vzťahu (18) mala nenulové riešenie pre α_1, tak musí byť matica $(\Sigma - \lambda I_p)$ singulárna. Teda inak povedané, existuje nenulové riešenie práve vtedy, ak λ je vlastné číslo kovariančnej matice Σ. Kovariančná matica Σ má p nezáporných vlastných čísiel $\lambda_1, ..., \lambda_p$, pretože Σ je pozitívne semidefinitná matica. Zoberme si usporiadané vlastné hodnoty $\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_p \geq 0$. Potom platí, že

$$\alpha_1^T \Sigma \alpha_1 = \alpha_1^T \lambda I_p \alpha_1 = \lambda \alpha_1^T \alpha_1 = \lambda. \tag{19}$$

Teda vlastné číslo, ktoré maximalizuje variaciu $\alpha_i^T X$ bude λ_1. Potom zo vzťahu (18) vyplýva, že α_1 je vlastný vektor kovariančnej matice Σ pre najväčšiu vlastnú hodnotu λ_1.

Druhý hlavný komponent Y_2 určíme podobným spôsobom, a to výberom α_2 tak, aby Y_2 malo druhú najväčšiu možnú varianciu. Pribudne k tomu podmienka aby bolo Y_2 nekorelované s pôvodnými hlavnými komponentami, teda Y_1, čo môžeme matematicky vyjadriť ako

$$Cov(Y_2, Y_1) = Cov(\alpha_2^T X, \alpha_1^T X) = \alpha_2^T \Sigma \alpha_1 = 0. \tag{20}$$

Vieme však, že $\Sigma\alpha_1 = \lambda_1\alpha_1$, čoho použitím zjednodušíme výraz (20) a dostaneme podmienku

$$\alpha_2^T \alpha_1 = 0. \tag{21}$$

Nová Lagrangeova funkcia pre α_2 bude vyzerať nasledovne

$$L(\alpha_2) = \alpha_2^T \Sigma \alpha_2 - \lambda(\alpha_2^T \alpha_2 - 1) - \delta \alpha_2^T \alpha_1,$$

kde λ a δ sú Lagrangeove multiplikátory. Funkciu zderivujeme podľa α_2:

$$\frac{\partial L}{\partial \alpha_2} = 2\Sigma\alpha_2 - 2\lambda\alpha_2 - \delta\alpha_1. \tag{22}$$

Vzťah (22) položime rovný nule a upravíme prenásobením α_1^T

$$2\alpha_1^T(\Sigma\alpha_2 - \lambda\alpha_2) - \delta = 0.$$

Využitím rovníc zo vzťahu (20), podmienky (21) a vzťahu $\alpha_1^T\alpha_1 = 1$ dostaneme, že $\delta = 0$ a teda vzťah

$$2\alpha_1^T(\Sigma\alpha_2 - \lambda\alpha_2) = 0,$$

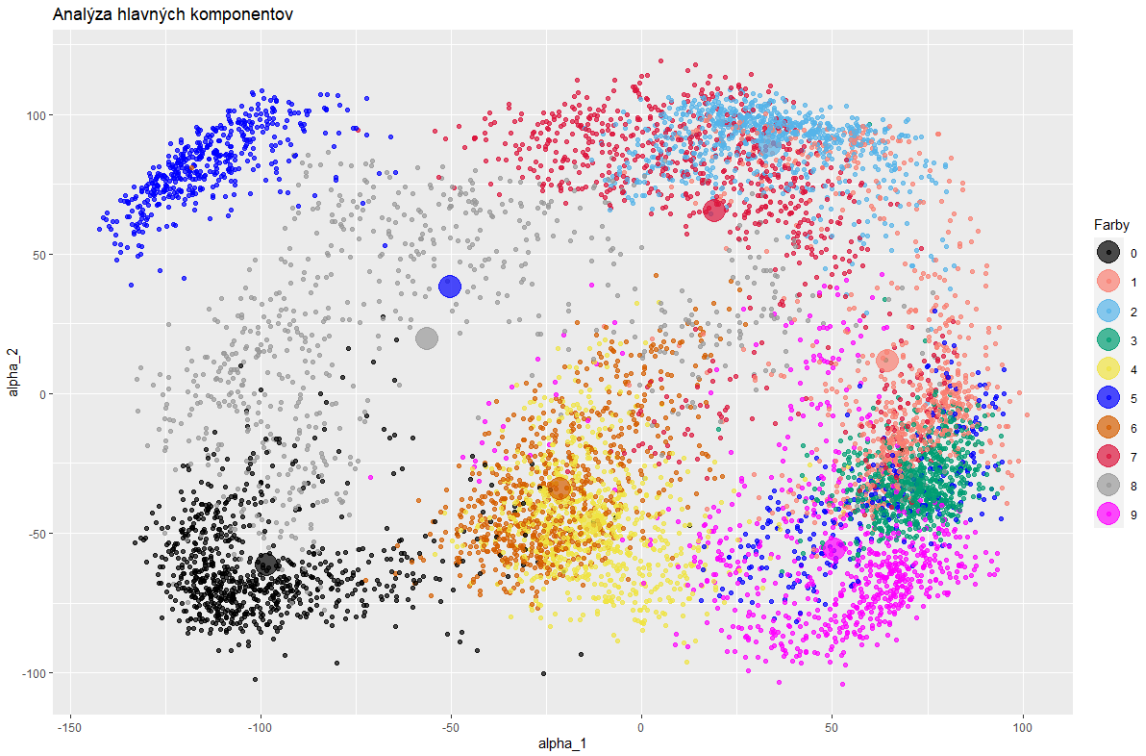
ktorý pripomína vzťah (18) pri hľadaní prvého hlavné komponentu. V tomto prípade vyberáme za λ druhé najväčšie vlastné číslo matice Σ a α_2 je príslušný vlastný vektor. λ sa už nemôže rovnať λ_1, pretože potom by sme dostali rovnosť $\alpha_1 = \alpha_2$ a nebola by splnená podmienka $\alpha_1^T\alpha_2 = 0$.

Takýmto postupom vieme vypočítať aj nasledujúce hlavné komponenty $Y_3, ..., Y_p$, kde $\lambda_3, ..., \lambda_p$ by boli vlastné čísla kovariančnej matice Σ a $\alpha_3, ..., \alpha_p$ ich vlastné vektory.

V praxi sa to dosiahne výpočtom kovariančnej matice pre celý súbor údajov. Ďalej sa vypočítajú vlastné vektory a vlastné hodnoty kovariančnej matice a zoradia sa podľa klesajúcej vlastnej hodnoty. Vo všeobecnosti využívajú, že *i*-ty jednotkový vektor je kolmý na *i-1*.

Namiesto hlavných komponentov sa na vizualizáciu použijú výberové hlavné komponenty, ktoré vzniknú nahradením priemeru μ výberovým priemerom $\overline{\mathbf{X}} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{X}_i$ a kovariančnej matice Σ výberovou kovariančnou maticou $\mathbf{S} = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{X}_i - \overline{\mathbf{X}})(\mathbf{X}_i - \overline{\mathbf{X}})^T$, kde $\mathbf{X}_i$ je realizácia p-rozmerného náhodného vektora pozorovaných premenných na i-tom objekte pre $i = 1, ..., n$.

Na náš dátový súbor sme aplikovali metódu PCA. V tomto dátovom súbore majú všetky kvantitatívne premenné rovnakú jednotku, preto sme hodnoty pozorovaní pred aplikovaním metódy PCA nepreškálovali. Analýzu hlavných komponentov a vizualizáciu dátového súboru do dvojrozmeru spolu s výberovými priemermi pre každú kategóriu môžeme vidieť na Obr. 8. Informácie o dôležitosti hlavných komponentov ako miery celkových rozptylov dát, smerodajné odchýlky a kumulatívne rozptyly sú uvedené v tabuľke 1.



Obr. 8: Vizualizácia dátového súboru metódou PCA.

Pri vizualizácií našich dát do dvojrozmeru pomocou metódy PCA na Obr. 8 nám vznikli rozličné „mraky" dát, kde vidno isté prekryvy medzi jednotlivými kategóriami, čo naznačuje, že niektoré číslice ľudia písali podobne a pri týchto kategóriach by mohol mať klasifikátor problém správne určiť o akú číslicu ide. [7] Zaujímavou kategóriou je kategória

[7] Avšak aj ak sú prvé dve hlavné komponenty dvoch rôznych cifier podobné, neznamená to nutne, že

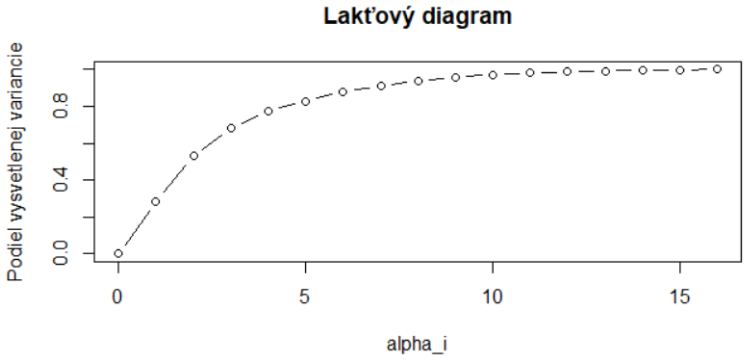
α_i	Smerodajná odchýlka	Podiel rozptylu	Kumulatívny rozptyl
α_1	65.3149	0.2851	0.2851
α_2	60.4275	0.2440	0.5291
α_3	47.9528	0.1537	0.6827
α_4	36.93311	0.09115	0.77388
α_5	28.65259	0.05486	0.82874
α_6	27.59586	0.05089	0.87962
α_7	21.11449	0.02979	0.90941
α_8	20.47350	0.02801	0.93742
α_9	17.3012	0.0200	0.9574
α_{10}	14.6263	0.0143	0.9717
α_{11}	11.46555	0.00878	0.98050
α_{12}	10.50446	0.00737	0.98788
α_{13}	8.25215	0.00455	0.99243
α_{14}	7.68500	0.00395	0.99638
α_{15}	5.39746	0.00195	0.99832
α_{16}	5.01091	0.00168	1.00000

Tabuľka 1: Miery celkového rozptylu dát, smerodajné odchýlky a kumulatívne rozptyly lineárnej transformácie metódy PCA na dátovom súbore *Pen-Based Recognition of Handwritten Digits*.

5, kde máme dva veľmi odlišné mraky v ľavom hornom a v pravom dolnom rohu pre rovnakú kategóriu. Toto bude pravdepodobne spôsobené dvomi rozdielnymi postupmi, ktoré ľudia používajú pri písaní cifry 5 na tablete.

Z tabuľky 1 a z lakťového diagramu na Obr. 9 vidíme, že prvý smer maximálneho rozptylu α_1 vysvetľuje iba 28.51% celkového rozptylu. Druhý smer maximálneho rozptylu vysvetľuje ďalších 24.40% celkového rozptylu. Spolu tieto dva smery zachytávajú len 52.91% informácie z pôvodných dát. Na ešte väčší obnos informácie o pôvodných dátach by sme museli použiť viac a vizualizovať dáta do vyšších dimenzií. V lakťovou diagrame

sú podobné aj celé p-rozmerné vektory čŕt daných dvoch cifier. Ide totiž len o priemet mnohorozmerných dát do dvojrozmerného priestoru.



Obr. 9: Lakťový diagram pre hodnoty vysvetlenej variancie pre α_i.

na Obr. 9 sa nenachádza žiaden hlavný komponent, ktorý by mal výrazný „skok" vo vysvetlení variancie. Na vysvetlenie aspoň 80% variancie pôvodných dát by sme potrebovali použiť až 5 hlavných komponentov.

3.3 Predstavenie problému klasifikácie a výberu hyperparametrov

Našim hlavným cieľom v tejto práci je využiť Bayesovskú optimalizáciu pri výbere hyperparametrov metód strojového učenia. Našou zvolenou metódou strojového učenia je v tejto práci metóda oporných bodov. Na prácu s SVM sme využili balíček *library(kernlab)* [9], ktorý sa dá implementovať do programovacieho prostredia *R-Studio* nainštalovaním tohto balíčka.

Pri metóde SVM sa hyperparametre vyskytujú pri výbere jadra. Zvolili sme si polynomiálne jadro (15), ktoré obsahuje tri hyperparametre: *stupeň*, *mierka* a *posun*. Z tejto trojice hyperparametrov sme sa rozhodli zafixovať *stupeň = 1*, čoho príčinou je, že budeme hľadať len lineárne separačné priamky, ktoré sa budú snažiť rozdeľovať do tried jednotlivé číslice. Hodnoty hyperparametrov *mierky* a *posunu* budeme skúmať a snažiť sa nájsť najlepšie hodnoty týchto hyperparametrov, ktoré budú dávať najlepšiu úspešnosť pri trénovaní klasifikátora.

Dátový súbor *Pen-Based Recognition of Handwritten Digits* obsahuje kategoriálnu premennú s číslicami od 0 po 9, takže máme 10 klasifikačných tried, medzi ktorými budeme hľadať správnu klasifikáciu. Riešime multikategoriálny problém na rozpoznanie desiatich číslic medzi sebou. Ako sme spomínali v podkapitole 2.6, tak existujú dva prístupy, ako

previesť multikategoriálny problém na binárny problém.

Prístup *jeden na jedného* predstavuje v našom prípade $\binom{10}{2}$ binárnych tried, teda 45 samostatných optimalizačných problémov. Pri tomto prístupe by sa pre každú dvojicu cifier natrénovali pomocou Bayesovskej optimalizácie hodnoty hyperparametrov, ktoré by mali pre každý optimalizačný problém najlepšiu percentuálnu úspešnosť pre jednotlivé dvojice. Nové pozorovanie x by bolo klasifikované všetkými binárnymi triedami z kolekcie. Klasifikácia x by sa vybrala tá, ktorá by sa najčastejšie vyskytovala vo výsledkoch týchto binárnych klasifikácií. Celkovo by sme museli nájsť 45 dvojíc optimálnych hodnôt hyperparametrov, ktoré by mali najväčšiu percentuálnu úspešnosť klasifikácie.

Druhý prístup *jeden voči ostatným* predstavuje len 10 optimalizačných problémov, kde by klasifikátor určil pre každú číslicu samostatne, či sa jedná o tú konkrétnu číslicu, alebo si myslí, že to je nejaká iná. Nové pozorovania sú klasifikované podľa toho, ktorý model je z klasifikácie „najistejší“, teda pri ktorom z desiatich modelov vyšla najväčšia pravdepodobnosť, že ide práve o tú číslicu. Celkovo by nám stačilo nájsť 10 dvojíc optimálnych hodnôt hyperparametrov, ktoré by mali najväčšiu percentuálnu úspešnosť klasifikácie. V tejto práci použijeme prístup jedného voči všetkým kvôli efektívnejšiemu prístupu k problému a aj kvôli počtu optimalizácií, ktoré by sme museli vykonať.

Problém pri tejto klasifikácii môže nastať pri trénovaní dostatočne dobrého klasifikátora, kedže niektoré číslice sa môžu písať podobne a klasifikátor by si ich mohol mýliť. Už pri vizualizácii dátového súboru pomocou PCA v časti 3.2 sme videli isté prekryvy medzi „mrakmi" dát a práve v týchto prekryvoch by sa mohli vyskytovať chybné klasifikácie.

3.4 Optimalizácia hyperparametrov pri SVM

Cieľom optimalizácie hyperparametrov *mierky* a *posunu* je nájdenie najlepších hodnôt hyperparametrov, pri ktorých dostaneme najlepšiu percentuálnu úspešnosť pri natrénovaní klasifikátora pri SVM a výslednú úspešnosť otestujeme na predikciách na testovacej sade.

Dátový súbor si náhodne rozdelíme na trénovaciu množinu a testovaciu množinu v pomere 80 : 20 z celkových 7494 pozorovaní. Náhodným rozdelením dát na trénovaciu a testovaciu vzorku znížime tzv. „bias", teda aby nebol náš model príliš zaujatý a testoval sa na dátach, ktoré ešte nevidel. Inak by mohol model dospieť k pretrénovaniu a mohol by byť príliš zaujatý na dáta, na ktorých sa trénoval. Potom by bol daný natrénovaný

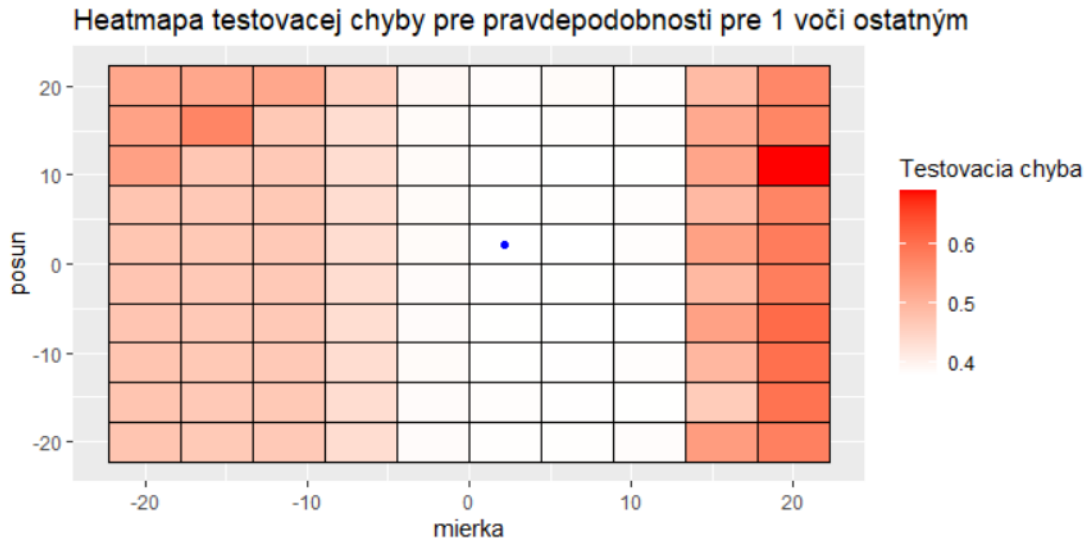
model veľmi dobrý na konkrétnu sadu dát, no na nových dátach by už nemusel byť tak dobrý, keby sme napríklad chceli zobrať dodatočné dáta zo zdroja [2] na ďalšie testovanie klasifikácie.

Do Bayesovskej optimalizácie potrebujeme zadať rôzne hodnoty podobne ako v časti 1.6 pri optimalizácii Rosenbrockovej funkcie. Na začiatku však netušíme v akom rozmedzí by sme mali hľadať naše hodnoty hyperparametrov. Použijeme mriežkové prehľadávanie z časti 2.7, kde prehľadáme priestor hodnôt hyperparametrov, za účelom nájdenia hraníc, v ktorých by sa nám oplatilo prehľadávať najlepšie hodnoty hyperparametrov. Táto metóda sa dá implementovať v programovacom prostredí *R-Studio* pomocou knižnice *library(mlr)* [1]. Využijeme metódu k-násobnej krížovej validácie na výber trénovacej sady a testovacej sady. Na trénovacej sade sa natrénuje pre rôzne kombinácie hodnôt hyperparametrov. Následne sa na testovacej vzorke určí testovacia chyba modelu, akú mali rôzne hodnoty hyperparametrov v mriežkovom prehľadávaní.

Postupným prehľadávaním priestoru a skúmaním výsledkov testovacej chyby sme našli približné hranice, v ktorých hodnoty hyperparametrov dávali najmenšiu chybovosť pri klasifikácii jedného voči ostatným.

Na Obr. 10 môžeme vidieť príklad výslednej mriežky rozmeru 10×10 pri prehľadávaní úspešnosti hyperparametrov pri klasifikácii jednotky voči ostatným, kde k pri k-násobnej krížovej validácii sme nastavili na $k = 3$. Pre $k = 3$ krížová validácia rozdelila náhodne dátový súbor na 3 časti a vždy zobrala *k-1*-tice dát na trénovanie modelu a na zvyšnej časti sa otestovala. Tento postup sa opakoval k krát a výsledné chybovosti modelov sa spriemerovali na jeden výsledok testovacej chyby. Testovaciu chybu sme na Obr. 10 transformovali funkciou $chyba \rightarrow chyba^{1/4}$, kvôli lepšej interpretácii výsledkov (táto transformácia zvýrazní rozdiely medzi malými hodnotami chyby). Taktiež pri nastavení mriežky a prehľadávaní sme využili transformácie hodnôt hyperparametrov cez funkciu 2^x, aby sme využili priestor spojitého prehľadávania. Tesne pred odovzdaním hodnôt hyperparametrov do učiaceho sa algoritmu sa použije transformačná funkcia, ktorá spraví prehľadávanie jemnejším. Výsledná úspesnosť klasifikátora jednotky voči ostatným mriežkovým prehľadávaním na testovacej vzorke bola 95.38565%.

Odhadli sme výsledné hranice [-20,20] pre obe hodnoty hyperparametrov, v ktorých testovacia chyba modelov mala malé hodnoty testovacej chyby a oplatilo by sa tieto oblasti



Obr. 10: Výsledné mriežkové prehľadávanie pre hodnoty hyperparametrov pri klasifikácií jednotky voči ostatným pri krížovej validácii pre $k = 3$. Modrá bodka predstavuje hodnoty hyperparametrov *mierky* = 2.2222 a *posunu* = 2.2222, kde bola najmenšia testovacia chyba = 0.02068335 v mriežkovom prehľadávaní. Hodnoty testovacej chyby sú zobrazené na obrázku ako $(chyba)^{1/4}$, kvôli lepšej interpretácii výsledkov.

preskúmať lepšie.

Keď už vieme približné hranice, v ktorých by sme mohli hodnoty hyperparametrov hľadať, tak aplikujeme Bayesovskú optimalizáciu na výber hyperparametrov pri metóde SVM.

Na začiatok pri Bayesovskej optimalizácii potrebujeme zadať funkciu, ktorú chceme optimalizovať. V našom prípade chceme optimalizovať funkciu od dvoch hodnôt hyperparametrov, ktorých hodnoty dostanú rôzne úspešnosti pri trénovaní klasifikácie jedného voči ostatným pre danú cifru. Snažíme sa maximalizovať úspešnosť klasifikátora a nájsť hodnoty hyperparametrov, v ktorých dosahuje túto najväčšiu úspešnosť.

V každom kroku optimalizácie Bayesovská optimalizácia nájde hodnoty hyperparametrov, s ktorými sa následne natrénuje klasifikátor na trénovacej sade, ktorý použije hodnoty hyperparametrov pri polynomiálnom jadre. Natrénovaný klasifikátor sa otestuje na testovacej sade, kde klasifikátor pri každej vzorke z testovacej sady určí pravdepodobnosť, s akou si myslí, že sa jedná práve o tú cifru. Výsledkom funkcie od dvoch hodnôt hyperparametrov je suma všetkých hodnôt pravdepodobností pri vzorkách z testovacej sady, kde správne predikoval danú cifru, pre dané hodnoty hyperparametrov.

Ďalej zadáme hranice pre hodnoty hyperparametrov, aby Bayesovská optimalizácia vedela, v ktorých hodnotách môže prehľadávať. Použijeme rovnaké hranice [-20,20] pre hodnoty hyperparametrov ako sme použili pri mriežkovom prehľadávaní.

Ako akvizičnú funkciu zvolíme pravdepodobnosť zlepšenia z časti 1.5.1, kde sa vyberá bod na ďalšie pozorovanie s najvyššou pravdepodobnosťou zlepšenia, teda s maximálnou očakávanou užitočnosťou od pôvodnej najlepšej hodnoty.

Ďalej do Bayesovskej optimalizácie zadáme počet náhodných počiatočných bodov, ktoré si Bayesovská optimalizácia náhodne vyberie a vypočíta v nich funkčné hodnoty. Zadáme aj počet iterácii, v ktorých sa pokúsi nájsť najlepšiu hodnotu pomocou algoritmu Bayesovskej optimalizácie. Zvolíme 10 počiatočných bodov a 20 iterácií pre každý optimalizačný problém.

Výsledkom tohto procesu je nájdenie desiatich dvojíc hodnôt hyperparametrov, ktoré dávali najlepšiu úspešnosť klasifikácie pre jednotlivé optimalizačné problémy prístupu jeden voči ostatným. Výsledky dvojíc hodnôt hyperparametrov, ktoré našla Bayesovská optimalizácia ako najlepšie, sú zobrazené v tabuľke 2 a uvádzame aj príklad iterácií algoritmu Bayesovskej optimalizácie pre klasifikátor kategórie 1 voči ostatným, ktorý je na Obr. 11.

Získané hodnoty hyperparametrov z tabuľky 2 využijeme pri trénovaní desiatich finálnych klasifikátorov jedného voči ostatným pre každú cifru. Postup trénovania a testovania finálnych klasifikátorov je rovnaký, akurát teraz budeme klasifikátory trénovať s najlepšími hodnotami hyperparametrov, ktoré sme dostali z Bayesovskej optimalizácie.

Výsledné úspešnosti klasifikácie jednotlivých klasifikátorov jedného voči ostatným sme vypočítali rovnako ako sumu pravdepodobností predikcií, v ktorých predikoval správne na testovacej sade. Tieto úspešnosti jednotlivých klasifikátorov sú zobrazené v tabuľke 3.

Na Obr. 12 a Obr. 13 vidíme hodnoty pravdepodobností pre každý klasifikátor jedného voči ostatným pre všetky vzorky v testovacej sade. Pre každú cifru z testovacej sady je určená pravdepodobnosť, s akou si jednotlivý klasifikátor myslí, že ide o to číslo. Čím nižšia pravdepodobnosť, tým viac si myslí, že sa ide o nejakú inú číslicu. Čím viac sú hodnoty pravdepodobností na obrázkoch „rozhádzané", tým sú jednotlivé klasifikátory neistejšie pri určovaní o akú cifru ide. Hodnoty pravdepodobností na Obr. 12 a Obr. 13 pri jednotlivých klasifikátoroch 1,5,8,9 sú najviac „rozhádzané", čo vidíme aj pri úspešnosti

	Hyperparametre	
Bayesovská optimalizácia	mierka	posun
0 voči ostatným	6.772092	-13.507220
1 voči ostatným	4.838229	16.749783
2 voči ostatným	4.502901	10.892654
3 voči ostatným	5.235159	-11.405021
4 voči ostatným	1.770210	-8.642625
5 voči ostatným	3.376477	19.158447
6 voči ostatným	12.882974	-5.676696
7 voči ostatným	5.353264	5.018647
8 voči ostatným	5.210141	7.947518
9 voči ostatným	5.931111	11.259977

Tabuľka 2: Výsledné najlepšie hodnoty hyperparametrov *mierky* a *posunu* pre jednotlivé optimalizácie typu "jeden voči ostatným" nájdené pomocou Bayesovskej optimalizácie.

Klasifikátory	Úspešnosti v %
0 voči ostatným	97.98079
1 voči ostatným	95.80445
2 voči ostatným	99.03507
3 voči ostatným	98.70732
4 voči ostatným	98.83540

Klasifikátory	Úspešnosti v %
5 voči ostatným	94.82321
6 voči ostatným	99.55431
7 voči ostatným	98.11030
8 voči ostatným	95.79669
9 voči ostatným	95.94809

Tabuľka 3: Úspešnosti jednotlivých klasifikátorov jedného voči ostatným na testovacej sade.

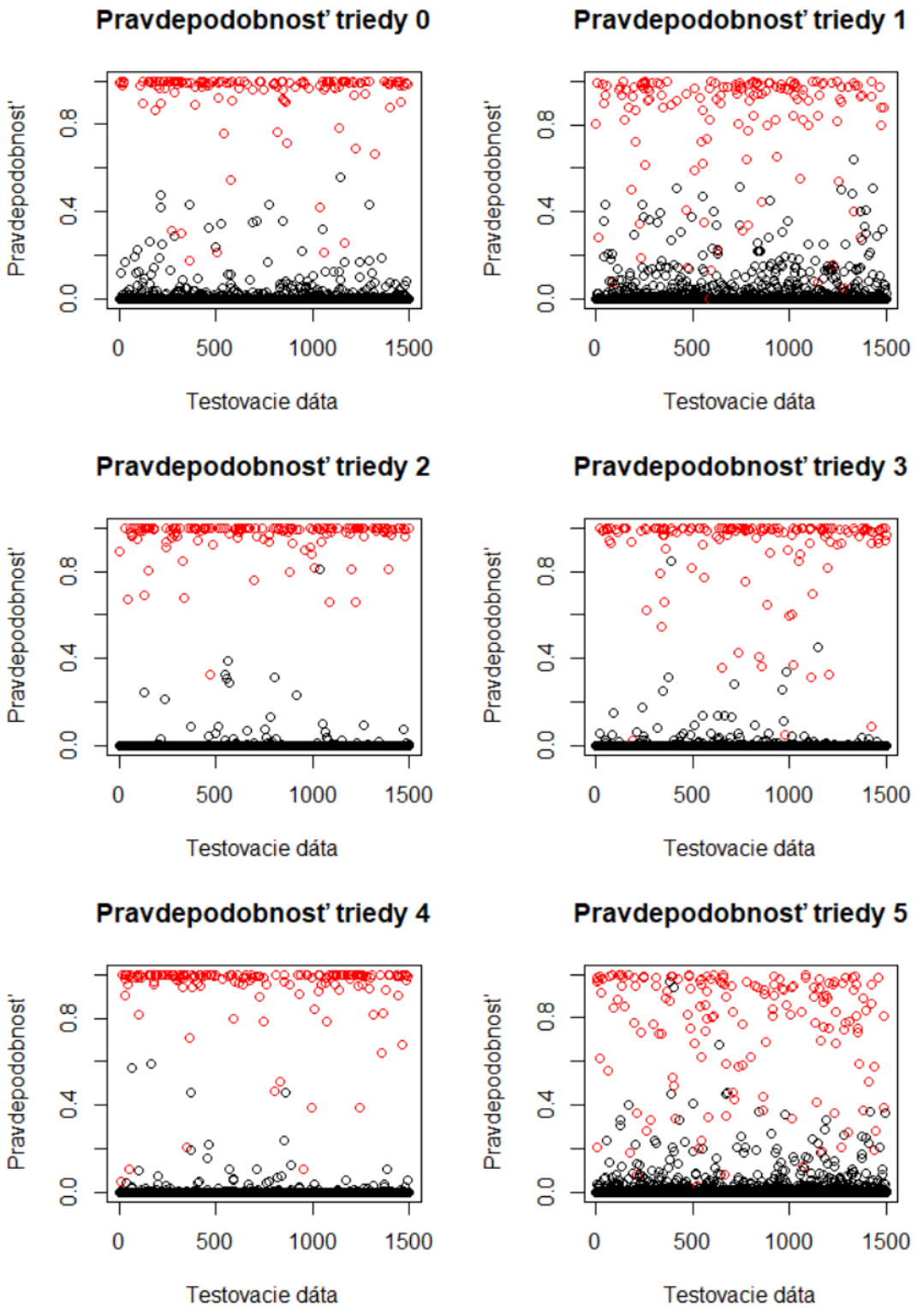
```
>     bayesall1 <- BayesianOptimization(FUN = BayOptall1,
+                                       bounds = search_bound,
+                                       init_grid_dt = NULL,
+                                       init_points =10,
+                                       n_iter = 20,
+                                       acq = "poi" )
elapsed = 1.56  Round = 1       scale = -17.22556       offset = -4.292112      Value = 938.1059
elapsed = 1.56  Round = 2       scale = 12.71101        offset = 12.55522       Value = 1435.3149
elapsed = 1.81  Round = 3       scale = 17.70487        offset = -4.950062      Value = 1434.6589
elapsed = 1.54  Round = 4       scale = -9.224725       offset = -4.767513      Value = 938.1059
elapsed = 1.55  Round = 5       scale = -13.22608       offset = -9.403265      Value = 938.1059
elapsed = 1.54  Round = 6       scale = -18.64418       offset = -2.426627      Value = 938.1059
elapsed = 1.53  Round = 7       scale = -12.8486        offset = -1.695714      Value = 938.1059
elapsed = 1.16  Round = 8       scale = 5.666615        offset = 1.628302       Value = 1435.6808
elapsed = 1.64  Round = 9       scale = -19.08489       offset = 6.627193       Value = 938.1059
elapsed = 1.56  Round = 10      scale = -19.66701       offset = -15.49204      Value = 938.1059
elapsed = 1.17  Round = 11      scale = 5.772159        offset = -9.510327      Value = 1435.5782
elapsed = 1.67  Round = 12      scale = 13.47569        offset = -15.91743      Value = 1435.2881
elapsed = 1.17  Round = 13      scale = 5.569869        offset = -16.01464      Value = 1435.7548
elapsed = 1.12  Round = 14      scale = 5.464163        offset = -16.02828      Value = 1435.8677
elapsed = 1.13  Round = 15      scale = 5.372491        offset = -6.596514      Value = 1435.9239
elapsed = 1.16  Round = 16      scale = 5.310754        offset = -6.127182      Value = 1435.9763
elapsed = 1.14  Round = 17      scale = 5.159285        offset = -6.399603      Value = 1436.0309
elapsed = 1.12  Round = 18      scale = 5.055266        offset = -6.440149      Value = 1436.0134
elapsed = 1.75  Round = 19      scale = 16.29953        offset = 5.896458       Value = 1434.7134
elapsed = 1.14  Round = 20      scale = 5.061667        offset = -6.301141      Value = 1436.0189
elapsed = 1.31  Round = 21      scale = 5.060303        offset = -6.479914      Value = 1436.0003
elapsed = 1.11  Round = 22      scale = 5.117411        offset = -6.357432      Value = 1436.0013
elapsed = 1.89  Round = 23      scale = 20.0000 offset = 16.95923       Value = 1434.6077
elapsed = 1.14  Round = 24      scale = 5.157595        offset = -6.3996        Value = 1435.9886
elapsed = 1.11  Round = 25      scale = 5.096021        offset = -15.75624      Value = 1436.0076
elapsed = 1.12  Round = 26      scale = 5.103539        offset = 10.71411       Value = 1436.0406
elapsed = 1.13  Round = 27      scale = 5.120978        offset = -15.88975      Value = 1436.0188
elapsed = 1.12  Round = 28      scale = 5.103361        offset = -11.15567      Value = 1435.9746
elapsed = 1.11  Round = 29      scale = 4.838229        offset = 16.74978       Value = 1436.1086
elapsed = 1.10  Round = 30      scale = 4.957181        offset = 14.28887       Value = 1436.0434

 Best Parameters Found:
Round = 29      scale = 4.838229        offset = 16.74978       Value = 1436.1086
```

Obr. 11: Postup algoritmu Bayesovkej optimalizácie pri hľadaní najlepších hodnôt hyperparametrov pri klasifikácii jednotky voči ostatným s 10 počiatočnými bodmi a 20 vykonanými iteráciami. *Value* predstavuje funkčnú hodnotu, ktorú chceme maximalizovať.

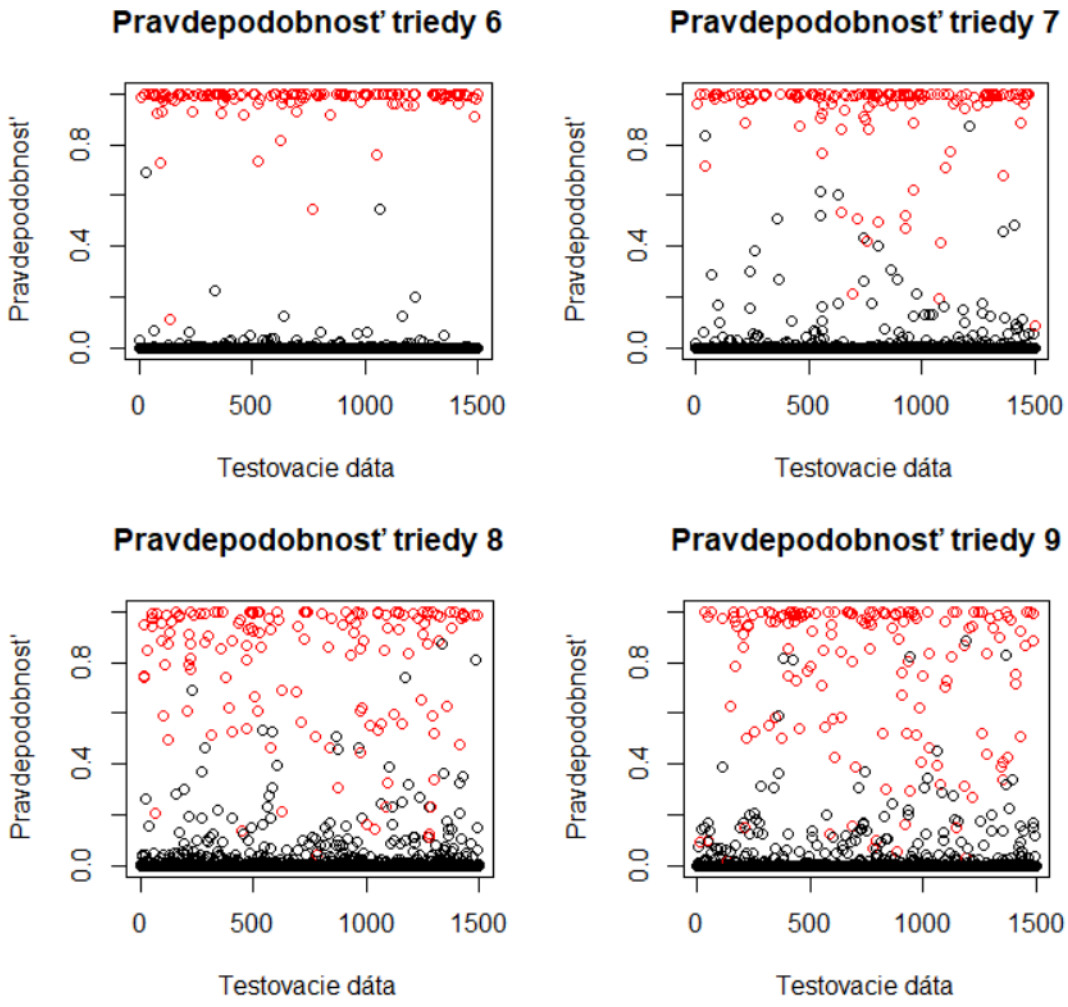
týchto klasifikátorov v tabuľke 3. Naopak hodnoty pravdepodobností pri klasifikácii 6 voči ostatným sú takmer jasne rozdelené na 1 (je to číslica 6) alebo 0 (je to iná číslica).

Teraz z týchto 10 klasifikátorov pre každý optimalizačný problém potrebujeme spraviť jeden finálny klasifikátor, ktorý bude vedieť povedať o akú cifru ide pre hocijakú vzorku z testovacej sady. Celkový klasifikátor vyhodnotí 10 klasifikácií, v ktorých už máme pravdepodobnosti pri predikcii na každú cifru z testovacej sady a zoberie si ako finálnu predikciu tú cifru, pri ktorej bola najväčšia pravdepodobnosť pri klasifikácii.

Celkový klasifikátor sme natrénovali a otestovali na testovacej sade. Z 1499 testovacích vzoriek v testovacej sade bola úspešnosť celkového klasifikátora 94.92995%, pričom sa pri klasifikácii pomýlil len 76 krát, zatiaľ čo 1423 krát sa nepomýlil a klasifikoval správne danú testovaciu vzorku. Natrénovaný celkový klasifikátor pomocou Bayesovskej optimalizácie dokáže s takmer 95% úspešnosťou správne klasifikovať novú vzorku.

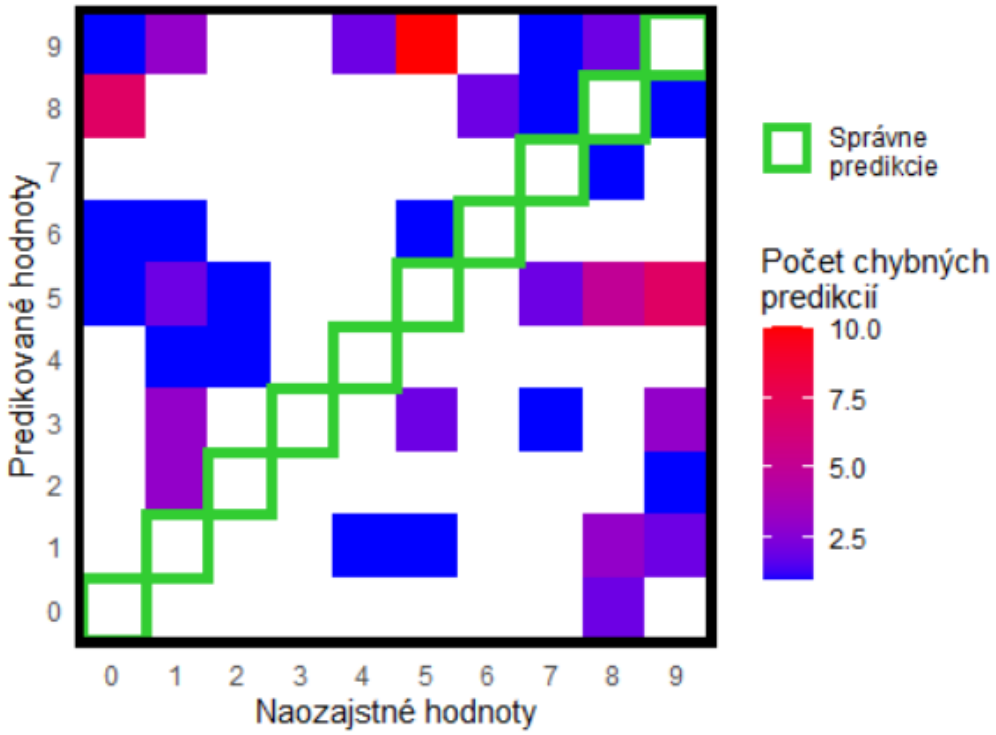


Obr. 12: Výsledné pravdepodobnosti pre každú vzorku z testovacej sady určené natrénovanými klasifikátormi pre triedy 0 až 5 pri SVM. SVM odhaduje, či je to daná cifra, alebo nejaká iná. Červenými bodkami sú znázornené objekty (rukou písané cifry), ktoré zodpovedajú tej jednej klasifikátorom detekovanej cifre.



Obr. 13: Výsledné pravdepodobnosti pre každú vzorku z testovacej sady určené natrénovanými klasifikátormi pre triedy 6 až 9 pri SVM. SVM odhaduje, či je to daná cifra, alebo nejaká iná. Červenými bodkami sú znázornené objekty (rukou písané cifry), ktoré zodpovedajú tej jednej klasifikátorom detekovanej cifre.

Pri metóde PCA z časti 3.2 sme videli isté prekryvy medzi „mrakmi" dát. Tieto prekryvy nám signalizovali, že niektoré cifry písali ľudia na tablet podobne a práve v týchto prekryvoch by sa mohol celkový klasifikátor mýliť. Pozrieme sa na klasifikácie z testovacej sady, v ktorých sa celkový klasifikátor pomýlil. Z týchto nesprávnych klasifikácii určíme, v ktorých cifrách sa najviac mýlil a či spolu nejako súvisia s prekryvmi dát na Obr. 8 pri metóde PCA. Na Obr. 14 sme zobrazili počty chybných predikcií pre jednotlivé kategórie cifier a aj hodnoty, za ktoré si ich celkový klasifikátor pomýlil.



Obr. 14: Zobrazenie počtu chybných predikcií pre jednotlivé dvojice naozajstnej hodnoty a zlej predikovanej hodnoty.

Z obrázka Obr. 14 jasne vidíme, že najčastejšou chybnou predikciou bolo pomýlenie si cifry päťky za deviatku, deviatky za päťku a aj nuly za osmičku. Kategória päťky bola pri vizualizácii pomocou PCA na Obr. 8 zobrazená do dvoch rozličných a odľahlých „mrakov" dát, kde práve druhý menší „mrak" dát tejto kategórie bol rozložený v okolí „mraku" deviatky.

3.5 Porovnanie časovej náročnosti a dosiahnutých výsledkov

V tejto časti sa pozrieme na dosiahnuté výsledky, časovú a výpočtovú náročnosť pri optimalizácii hyperparametrov pri metóde SVM pomocou mriežkového prehľadávania s krížovou validáciou a Bayesovskou optimalizáciou. Už v časti 1.6 sme porovnávali efektívnosť Bayesovskej optimalizácie oproti gradientným metódam BFGS a Nelder-Mead pri hľadaní minima a optimálnej funkčnej hodnoty na Rosenbrockovej funkcii. Bayesovská optimalizácia sa ukázala ako efektívny prístup, ktorý na podstatne menší počet evaluácii dokázal nájsť približne rovnaké výsledky ako gradientné metódy.

Pri optimalizácii hyperparametrov za pomoci mriežkového prehľadávania a aj Bayesovskej optimalizácie sme použili rovnaké hranice pre výber hodnôt hyperparametrov [-20,20] a aj rovnaké rozloženie dát na trénovaciu a testovaciu sadu.

Pri mriežkovom prehľadávaní pre každý klasifikátor jedného voči ostatným algoritmus musel vypočítať pre každú bunku mriežky 10×10 testovaciu chybu modelu pre dvojicu hodnôt hyperparametrov. Na vypočítanie tejto hodnoty musel 3 krát spraviť krížovú validáciu a vždy vypočítať testovaciu chybu modelu, z ktorej dostal zpriemerovanú výslednú chybovosť modelu pre danú dvojicu hodnôt hyperparametrov. Celkovo na trénovanie jedného najlepšieho klasifikátora jedného voči ostatným potreboval 300 krát natrénovať model a vypočítať príslušné testovacie chyby.

Veľký počet výpočtov a samotných evaluácii algoritmu vie byť náročný aj z časovej perspektívy. Trénovanie jedného najlepšieho klasifikátora jedného voči ostatným mriežkovým prehľadávaním trvalo rôzne dlho, najmenší čas bol 15 minút pre natrénovanie klasifikátora pre šestku a najviac až 5 hodín trvalo natrénovanie pre jednotku. Celkový čas procesov pri optimalizácii hyperparametrov pomocou mriežkového prehľadávania by sa hýbal okolo 16 hodín.

Pri použití Bayesovskej optimalizácie algoritmus musel vypočítať 10 algoritmom náhodne zvolených bodov na začiatok optimalizácie a následne vypočítať 20 iterácii, v ktorých sa snažil nájsť najväčšiu úspešnosť klasifikátora. Celkovo na trénovanie jedného najlepšieho klasifikátora jedného voči ostatným potreboval 30 krát natrénovať model a vypočítať funkčné hodnoty. Tento spôsob optimalizácie predstavuje 10 krát menší počet celkových trénovaní modelu. Z pohľadu časovej náročnosti celková Bayesovská optimalizácia všetkých desiatich klasifikátorov trvala dokopy menej ako hodinu.

Po nájdení najlepších hodnôt hyperparametrov a nátrenovaní modelov sme otestovali dané modely s najlepšími hodnotami hyperparametrov na testovacej sade. V tabuľke 4 môžeme vidieť porovnanie výsledných úspešností natrénovaných klasifikátorov mriežkovým prehľadávaním a Bayesovskou optimalizáciou spolu s nájdenými najlepšími hodnotami hyperparametrov pre jednotlivé kategórie. Všimnime si, že pri mriežkom prehľadávaní sa hodnoty niektorých hyperparametrov často opakujú, čo je spôsobené nastavením mriežky. Samotná mriežka by sa mohla spraviť väčšia, napríklad rozmeru 20×20, čoho dôsledkom by sme mali viac prehľadávaných hodnôt hyperparametrov, ale aj počet trénovaní by stúpol z 300 iterácií na 1200 trénovaní modelu a počítania testovacej chyby pre jeden klasifikátor.

Tabuľka 4

	Mriežkové prehľadávanie			**Bayesovská optimalizácia**		
Klasifikátory	**Mierka**	**Posun**	**Úspešnosť v %**	**Mierka**	**Posun**	**Úspešnosť v %**
0 voči ostatným	11.1111	-2.2222	95.81522	6.772092	-13.507220	97.98079
1 voči ostatným	2.2222	2.2222	95.38565	4.838229	16.749783	95.80445
2 voči ostatným	2.2222	-15.5556	99.06512	4.502901	10.892654	99.03507
3 voči ostatným	2.2222	20	98.88592	5.235159	-11.405021	98.70732
4 voči ostatným	11.1111	2.2222	95.53769	1.770210	-8.642625	98.83540
5 voči ostatným	11.1111	2.2222	92.54231	3.376477	19.158447	94.82321
6 voči ostatným	-2.2222	15.5556	99.71685	12.882974	-5.676696	99.55431
7 voči ostatným	11.1111	20	96.57539	5.353264	5.018647	98.11030
8 voči ostatným	11.1111	20	94.71695	5.210141	7.947518	95.79669
9 voči ostatným	-2.2222	20	95.50112	5.931111	11.259977	95.94809

Z tabuľky 4 vidíme, že Bayesovská optimalizácia dosahovala vo väčšine prípadov lepšie alebo nanajvýš veľmi podobné úspešnosti jednotlivých klasifikátorov oproti mriežkovému prehľadávaniu. Avšak v rámci nákladového hľadiska, či už „šetrením" počtu evaluácii alebo časovej náročnosti je Bayesovskej optimalizácie efektívnejšia.

4 Možnosti rozšírenia

Priestoru na zlepšenie úspešnosti predstaveného finálneho klasifikátora je viacero. Samotné hľadanie pomocou Bayesovskej optimalizácie predstavuje proces, ktorý by sa mohol skúmaním a skúšaním vylepšiť. Niektoré nastavenia pri algoritme Bayesovskej optimalizácie by sa dali zmeniť a lepšie peskúmať ich možnosti. Napríklad zvýšenie počtu iterácií alebo počtu počiatočných bodov, rozšírenie alebo naopak zúženie hraníc hľadaných hyperparametrov či použitie rôznych akvizičných funkcií pri algoritme.

Našou prioritou pri tejto práci bolo využitím Bayesovskej optimalizácie nájsť hodnoty hyperparametrov, s ktorými sa natrénuje dostatočne dobrý klasifikátor na klasifikáciu cifier.

Finálny klasifikátor vyhodnocoval výslednú predikciu na základe najväčšej pravdepodobnosti dosiahnutej pri jednotlivých desiatich natrénovaných klasifikátoroch jedného voči ostatným.

Ako možnosť prípadného zlepšenia celkovej úspešnosti klasifikátora by mohlo byť rozšírenie o záverečnú kontrolnú binárnu klasifikáciu. Finálny klasifikátor by sa nepozrel len na najväčšiu pravdepodobnosť, podľa ktorej určil predikovanú cifru, ale zobral by si dve cifry s najväčšími pravdepodobnosťami. Potom by medzi týmito dvoma ciframi spravil binárnu klasifikáciu jedného na jedného a víťaz tejto klasifikácie by bola finálna predikcia. Jednotlivé klasifikátory jedného na jedného by sa dali dopredu natrénovať, no ako sme spomínali pri prevode multikategoriálnej klasifikácie na binárnu v časti 2.6, tak prístup jeden na jedného by predstavoval 45 optimalizačných problémov. (To by predstavovalo výpočtovo náročný problém, v ktorom by ale význam Bayesovskej optimalizácie voči mriežkovému prehľadávaniu mohol byť ešte výraznejší.)

Ako ďalšia možnosť rozšírenia by mohla byť aplikácia vedomostí o zhlukovaní. Zhlukovanie (angl. *clustering*) je metóda hľadania súvislostí v celkovej štruktúre a následné utriedenie objektov na základe podobných vlastností do zhlukov. Na dátový súbor sme už aplikovali jednu z metód, a to metódu hlavných komponentov. Pri tejto metóde sme dospeli k vizualizácii na Obr. 8, kde sme zobrazili „mraky" dát pre každú kategóriu.

Toto rozdelenie by sa dalo rozšíriť, kde by sme tieto dátové zhluky rozdelili na viac zhlukov a vytvorili podkategórie, napríklad pri kategórii cifry 5 by sme rozdelili túto kategóriu na dve, kde jedna podkategória by bola zhluk na obrázku úplne vľavo hore a

druhá podkategória zhluk vpravo dole. Dôsledkom tejto aplikácie by bolo viac kategórií, resp. podkategórií, medzi ktorými by sa klasifikátor rozhodoval.

Záver

Hlavným cieľom tejto diplomovej práce bola aplikácia Bayesovskej optimalizácie na výber hyperparametrov pri metódach strojového učenia. Diplomová práca je členená do štyroch kapitol.

V prvej kapitole sme sa venovali potrebnej teórii k definovaniu Bayesovskej optimalizácie. Vysvetlili sme si na akom princípe funguje a v akých prípadoch sa Bayesovská optimalizácia používa. Zadefinovali sme si základný algoritmus Bayesovskej optimalizácie a rôzne typy akvizičných funkcií, ktoré sa v tomto algoritme používajú. Na konci prvej kapitoly v časti 1.6 sme optimalizovali Rosenbrockovu funkciu za účelom nájdenia minima a optimálnej funkčnej hodnoty pomocou gradietných metód BFGS a Nelder-Mead a aj pomocou Bayesovskej optimalizácie. Pomocou všetkých použitých metód sa nám podarilo dosiahnuť žiadaného výsledku, avšak metódy BFGS a Nelder-Mead potrebovali viakrát evaluovať funkciu, aby dospeli k podobnému výsledku.

Druhú kapitolu sme venovali metódam strojového učenia, kde sme na začiatku uviedli čitateľa do problematiky metódy oporných bodov a vysvetlili sme základný prístup lineárnej klasifikácie. Ukázali sme, ako funguje klasifikácia metódou oporných bodov a kernelový trik, vďaka ktorému vieme z nelineárnej klasifikácie spraviť lineárnu. Vysvetlili sme na akom princípe funguje mriežkové prehľadavánie a krížová validácia. Taktiež sme v tejto časti ukázali, ako previesť multikategoriálnu klasifikáciu na binárnu, ktorú sme aplikovali v tretej kapitole.

Tretia kapitola obsahovala samotnú aplikáciu Bayesovkej optimalizácie na výber hyperparametrov metódy SVM. Na začiatok sme si dáta vizualizovali za pomoci metódy hlavných komponentov, kde sme sa za pomoci redukcie dimenzie dát snažili zobraziť mnohorozmerné dáta do dvojrozmerého priestoru. Na dátovom súbore *Pen-Based Recognition of Handwritten Digits* sme riešili klasifikačný problém, kde hlavným cieľom bolo natrénovanie klasifikátora, ktorý bude vedieť dostatočne dobre predikovať správnu cifru od 0 po 9 na testovacej vzorke. Tento multikategoriálny problém sa nám podarilo previesť na binárny, kde sme natrénovali 10 klasifikátorov pre každú cifru a na základe najväčšej pravdepodobnosti z týchto klasifikátorov sme určovali výslednú klasifikáciu. Pre porovnanie sme tento klasifikačný problém riešili dvoma prístupmi. Jedným prístupom bolo mriežkové prehľadávanie s krížovou validáciou, kde sa nám podarilo natrénovať jednotlivé

klasifikátory s úspešnosťou nad 92 %. Druhým prístupom bola Bayesovská optimalizácia, ktorá dosahovala trošku lepšie percentuálne úspešnosti a celkový klasifikátor natrénovaný pomocou Bayesovkej optimalizácie dosiahol 94.92995 % úspešnosť správnej klasifikácie na testovacej sade. Okrem lepšej úspešnosti pri klasifikácii, ktorú sme vypísali do tabuľky 4, sme sa pozerali aj na časovú a výpočtovú náročnosť týchto prístupov. Mriežkové prehľadávanie s krížovou validáciou potrebovalo na natrénovanie podobne úspešného klasifikátora z časového hľadiska približne 16 hodín, zatiaľ čo Bayesovská optimalizácia dosahovala podobných výsledkov okolo 1 hodiny. Výpočtová náročnosť bola tiež v prospech Bayesovskej optimalizácie, kde pri mriežkovom prehľadávaní bolo potrebných 300 evaluácii a trénovaní modelu, zatiaľ čo pri Bayesovskej optimalizácii stačilo 10 počiatočných bodov a 20 iterácii.

S výsledkami práce sme spokojní, no zároveň vidíme potenciál na zlepšovanie celkovej úspešnosti klasifikátora. V poslednej časti práce spomíname možnosti rozšírenia práce a spôsoby, ako by sa dala zlepšiť úspešnosť klasifikátora. Jedným zo spôsobov rozšírenia úspešnosti bolo rozšírenie o binárnu klasifikáciu, kde by klasifikátor nezobral len jednu najväčšiu pravdepodobnosť, ale pozrel by sa dve najväčšie a medzi nimi by spravil ďalšiu klasifikáciu. Druhým spomenutým spôsobom rozšírenia bolo použitie zhlukovania, ktoré by prinieslo rozdelenia dát do viac kategórií a podkategórií.

Zoznam použitej literatúry

[1] Bischl, B., Lang, M., Kotthoff, L., Schiffner, J., Richter, J., Studerus, E., Casalicchio, G., Jones, Z.: *mlr: Machine Learning in R.*, Journal of Machine Learning Research 17(170), 1-5., 2016, dostupné na internete (27.4.2022) https://jmlr.org/papers/v17/15-066.html

[2] Dua, D., Graff, C.: *Machine Learning repository*, University of California, School of Information and Computer Sciences, Irvine, 2019, dostupné na internete (9.1.2022): http://archive.ics.uci.edu/ml/datasets/Optical+Recognition+of+Handwritten+Digits

[3] Frazier, P. I.: *A tutorial on bayesian optimization*, arXiv preprint arXiv:1807.02811, 2018, dostupné na internete (9.1.2022): https://arxiv.org/pdf/1807.02811.pdf

[4] Harman, R.: *Multivariate Statistical Analysis: Selected Lecture Notes*, učebné texty, FMFI UK Bratislava, 2021, dostupné na internete (9.1.2022): http://www.iam.fmph.uniba.sk/ospm/Harman/VSAp.pdf

[5] Hastie, T., Tibshirani, R., Friedman, R.: *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, (Springer Science Business Media, Second Edition, 2009), ISBN: 978-0-387-84858-7

[6] Izenman, A. I.: *Modern Multivariate Statistical Techniques: Regression, classification, and manifold learning*, Springer, New York, USA, 2008, ISBN: 978-0-387-78189-1

[7] Janková, K., Pázman, A.: *Pravdepodobnosť a štatistika*, Univerzita Komenského, Bratislava, 2011.

[8] Jolliffe, I.T.: *Principal Component Analysis*, New York: Springer-Verlag New York, second edition, 2002, ISBN: 978-0-387-22440-4, https://link.springer.com/book/10.1007/b98835

[9] Karatzoglou, A., Smola, A., Hornik, K., Zeileis, A.: *kernlab - An S4 Package for Kernel Methods in R.* Journal of Statistical Software 11(9), 1-20., 2004, dostupné na internete (27.4.2022) http://www.jstatsoft.org/v11/i09/

[10] Rasmussen, C. E., Williams, Ch. K. I.: *Gaussian processes for machine learning*, MIT Press, 2005, ISBN: 9780262182539, dostupné na internete (15.3.2022): http://gaussianprocess.org/gpml/chapters/RW.pdf

[11] RStudio Team: *RStudio: Integrated Development Environment for R. RStudio*, Boston, 2021, dostupné na internete (27.4.2022) http://www.rstudio.com/

[12] Snoek J., Larochelle H., Adams P. R.: *Practical Bayesian Optimization of Machine Learning Algorithms*, University of Toronto, 2012, arXiv:1206.2944, dostupné na internete (9.1.2022): https://arxiv.org/abs/1206.2944

[13] Souza, C.: *Kernel Functions for Machine Learning Applications*, 2010, dostupné na internete (15.3.2022): http://crsouza.com/2010/03/17/kernel-functions-for-machine-learning-applications/#kernel_functions

[14] Tong, Y. L.: *The Multivariate Normal Distribution*, Springer Series in Statistics, 1990, ISBN: 978-1-4613-9655-0

[15] Vapnik, V. N.: *The Nature of Statistical Learning Theory*, Springer-Verlag, Berlin, 1995, ISBN: 978-1-4757-3264-1

[16] Vinař, T.: *Prednášky zo strojového učenia*, učebné texty, FMFI UK Bratislava, 2021, dostupné na internete (9.1.2022): http://compbio.fmph.uniba.sk/vyuka/ml/handouts/?fbclid=IwAR2gPYSA7c8n6D7QM6ozCqfufaGYQawBtwsTtrs5cZWLKF2f8r0FPiyJ10I

[17] Wilimitis, D.: *The Kernel Trick in Support Vector Classification*, dostupné na internete (15.3.2022): https://towardsdatascience.com/the-kernel-trick-c98cdbcaeb3f

[18] Yan, Y.: *rBayesianOptimization: Bayesian Optimization of Hyperparameters.*, 2021, dostupné na internete (27.4.2022) https://CRAN.R-project.org/package=rBayesianOptimization

Príloha

V tejto prílohe neuvádzame celý zdrojový kód, ale uvádzame iba niektoré príklady kódu, ktoré môžu slúžiť čitateľovi na prípadnú reprodukciu výsledkov a vyskúšanie si aplikácie Bayesovskej optimalizácie. Všetky kódy boli použité v programovacom prostredí *R-Studio*.

V časti I. uvádzame použité knižnice, ktoré je potrebné nainštalovať na použitie týchto častí kódu. Časť II. obsahuje načítanie dát a rozdelenie dát na trénovaciu a testovaciu sadu. V časti III. uvádzame optimalizáciu Rosenbrockovej funkcie pomocou Bayesovskej optimalizácie, ktorá bola spomenutá v časti 1.6. Časť IV. obsahuje metódu hlavných komponentov spolu s vizualizáciou dát pomocou prvých dvoch hlavných komponentov, ktorá bola na Obr. 8. V časti V. uvádzame príklad mriežkového prehľadávania pre trénovanie klasifikátora 1 voči ostatným spolu s vykreslením obrázka Obr. 10. V časti VI. uvádzame natrénovanie klasifikátora 1 voči ostatným za pomoci Bayesovskej optimalizácie (Obr. 11) a následné natrénovanie modelu už s najlepšími hodnotami hyperparametrov ($bayesall1\$Best\_Par$).

```
# I. Pouzite kniznice
library(rBayesianOptimization); library(ggplot2);
library(kernlab); library(mlr); library(tidyverse)

# II. Nacitanie dat, trenovacia a testovacia sada
setwd("␣") # cesta k suboru pendigits.txt
set.seed(12) # reprodukcia vysledkov
X <- read.table("pendigits.txt",header = FALSE,sep=",")
sample <- sample.int(n = nrow(X), size = floor(.80*nrow(X)),
replace = F)
train <- X[sample, ]
test <- X[-sample, ]

# III. Optimalizacia Rosenbrockovej funkcie

objective <- function(x1,x2) {
```

```
  fitness =  -(100 * (x2 - x1 * x1)^2 + (1 - x1)^2)
  result <- list(Score = fitness)
  return(result)
}

search_bound <- list(x1 = c(-2,2),
                     x2 = c(-2,2))
set.seed(123) # set.seed na reprodukciu vysledkov
bayes <- BayesianOptimization(FUN = objective,
                              bounds = search_bound,
                              init_grid_dt = NULL,
                              init_points =10,
                              n_iter = 20,
                              kappa = 2.576,
                              acq = "ucb" )
#  IV. PCA
#vyber farieb
cbPalette <- c("#000000","#FA8072","#56B4E9","#009E73",
               "#F0E442","#0000FF","#D55E00","#DC143C",
               "#999999","#FF00FF")

pca <- prcomp(X[,-cc],scale = FALSE)
matica <- matrix(rep(0,2 * k),ncol = 2)
for (i in 1:k){
  for (j in 1:2){
    cat <- as.character(unique(X[,cc])[i])
    matica[i,j] <- apply(pca$x[X[,cc] == cat,],2,mean)[j]}}
(ggplot()
  + geom_point(data = data.frame(pca$x),
    aes(x = data.frame(pca$x)[,1],y = data.frame(pca$x)[,2],
        color = as.factor(X[,cc])),shape = 19,alpha = 0.7)
```

```
  + geom_point(data = data.frame(matica),
    aes(x = data.frame(matica)[,1], y = data.frame(matica)[,2],
        color = as.factor(unique(X[,cc]))),size = 8,shape = 19,
        alpha = 0.7)
  + scale_colour_manual(values = cbPalette,name = "Farby")
  + ggtitle("Analyza hlavnych komponentov")
  + xlab("alpha_1")
  + ylab("alpha_2"))
summary(pca) # podiel vysvetleneho rozptylu

#laktovy diagram
plot(0:16, c(0, summary(pca)$importance[3, ]), type = "b",
ylim = c(0, 1), main="Laktov diagram", xlab="alpha_i",
ylab="Podiel vysvetlenej variancie" )

# V. Mriezkove prehladavanie pre 1 voci ostatnym

set.seed(123) #reprodukcia vysledku
X0 <- X
X0["V17"][X0["V17"] == "0"] <- "-1"
X0["V17"][X0["V17"] == "2"] <- "-1"
X0["V17"][X0["V17"] == "3"] <- "-1"
X0["V17"][X0["V17"] == "4"] <- "-1"
X0["V17"][X0["V17"] == "5"] <- "-1"
X0["V17"][X0["V17"] == "6"] <- "-1"
X0["V17"][X0["V17"] == "7"] <- "-1"
X0["V17"][X0["V17"] == "8"] <- "-1"
X0["V17"][X0["V17"] == "9"] <- "-1"
X0[, 'V17'] <- as.factor(X0[, 'V17'])

sampled1 <- sample.int(n = nrow(X0), size = floor(.80*nrow(X0)),
```

```
replace = F)
train1 <- X0[sampled1, ] # trenovacia sada pre 1
test1 <- X0[-sampled1, ] # testovacia sada pre 1

set.seed(123) # reprodukcia vysledku
start.time <- Sys.time() # pocitadlo casu

taskSVM <- makeClassifTask(data = train1,
                           target = "V17")

svmlearner <- makeLearner("classif.ksvm", predict.type = "prob")

# k=3 nasobna krizova validacia
rdesc = makeResampleDesc("CV", iters = 3L)

getParamSet("classif.ksvm") # nastavenie parametrov
ps = makeParamSet(
  makeDiscreteParam("kernel", values = "polydot"),
  makeIntegerParam("degree", lower = 1, upper = 1),
  makeNumericParam("scale", lower = -20, upper = 20,
  trafo =  function(x) 2^x),
  makeNumericParam("offset", lower = -20, upper = 20,
  trafo =  function(x) 2^x)
)
ctrl = makeTuneControlGrid(resolution = 10L) # 10x10 mriezka

#trenovanie modelu
tunemodel <- train(svmlearner, taskSVM)
tunedsvm = tuneParams(svmlearner, task = taskSVM,
           resampling = rdesc, par.set = ps, control = ctrl)
```

```
df = as.data.frame(tunedsvm$opt.path)

# natrenovany model
tuned <- setHyperPars(svmlearner, par.vals = tunedsvm$x)

# ukoncenie casomiery
end.time <- Sys.time()
time.taken <- end.time - start.time
time.taken # 4.6h +-

# predikcia na testovacej sade
taskSVMtest <- makeClassifTask(data = test1,
                               target = "V17")
tunedSvmModel = train(tuned, taskSVM)
datapredicted  <-predict(tunedSvmModel, task = taskSVM)
dp <- as.data.frame(datapredicted)
Pfp1 <- ifelse(dp$truth == dp$response & dp$truth == -1 ,
dp[ ,3],0)
Pfp2 <- ifelse(dp$truth == dp$response & dp$truth == 1 ,
dp[ ,4],0)
fp1 <- (sum(Pfp1)+ sum(Pfp2))/nrow(dp)
fp1 # finalna pravdepodobnost 1 voci ostatnym

# vykreslenie - heatmapa
ggplot(df, aes(x = scale, y = offset,
  fill = sqrt(sqrt(mmce.test.mean)))) +
  geom_tile(color = "black") +
  xlab("mierka")+
  ylab("posun")+
  geom_point(aes(x=2.2222, y=2.2222), colour="blue") +
  scale_fill_gradient(low = "white", high = "red")  +
```

```
  ggtitle(paste("Heatmapa testovacej chyby pre pravdepodobnosti
 pre 1 voci ostatnym" ) ) +
  guides(fill = guide_colourbar(title = "Testovacia chyba"))
  coord_fixed()

# VI. Bayesovska optimalizacia 1 voci ostatnym

BayOptall1 <- function (scale, offset) {

  set.seed(12)
  pd <- list(degree = 1, scale = scale, offset = offset)

  X["V17"][X["V17"] == "0"] <- "-1"
  X["V17"][X["V17"] == "2"] <- "-1"
  X["V17"][X["V17"] == "3"] <- "-1"
  X["V17"][X["V17"] == "4"] <- "-1"
  X["V17"][X["V17"] == "5"] <- "-1"
  X["V17"][X["V17"] == "6"] <- "-1"
  X["V17"][X["V17"] == "7"] <- "-1"
  X["V17"][X["V17"] == "8"] <- "-1"
  X["V17"][X["V17"] == "9"] <- "-1"
  X[, 'V17'] <- as.factor(X[, 'V17'])

  sample <- sample.int(n = nrow(X), size = floor(.80*nrow(X)),
  replace = F)
  train <- X[sample, ]
  test <- X[-sample, ]

  kernfit <- ksvm(V17~., train,
                  kernel= "polydot",
                  type = "C-svc",
```

```
                    kpar = pd,
                    prob.model=TRUE)

  prediction = predict(kernfit, test, type="probabilities")
  data <- as.data.frame(prediction)
  P <- ifelse(test$V17 == -1, data[ ,1], data[ ,2])
  final_percentage <-sum(P)
  result <- list(Score = final_percentage)
  return(result)
}
# hranice parametrov
lower_bounds <- c(scale = -20, offset = -20)
upper_bounds <- c(scale = 20, offset = 20)
search_bound <- list(scale = c(lower_bounds[1], upper_bounds[1]),
                offset = c(lower_bounds[2], upper_bounds[2]))

set.seed(12)
    bayesall1 <- BayesianOptimization(FUN = BayOptall1,
                                      bounds = search_bound,
                                      init_grid_dt = NULL,
                                      init_points =10,
                                      n_iter = 20,
                                      acq = "poi" )
bayesall1$Best_Par # najlepsie hodnoty

# trenovanie s najlepsimi hodnotami
X1 <- X
X1["V17"][X1["V17"] == "0"] <- "-1"
X1["V17"][X1["V17"] == "2"] <- "-1"
X1["V17"][X1["V17"] == "3"] <- "-1"
X1["V17"][X1["V17"] == "4"] <- "-1"
```

```
X1["V17"][X1["V17"] == "5"] <- "-1"
X1["V17"][X1["V17"] == "6"] <- "-1"
X1["V17"][X1["V17"] == "7"] <- "-1"
X1["V17"][X1["V17"] == "8"] <- "-1"
X1["V17"][X1["V17"] == "9"] <- "-1"
X1[, 'V17'] <- as.factor(X1[, 'V17'])
set.seed(12)
sampled1 <- sample.int(n = nrow(X1), size = floor(.80*nrow(X1)),
replace = F)
train1 <- X1[sampled1, ]
test1 <- X1[-sampled1, ]

polydot1 <- list(degree= 1, scale = bayesall1$Best_Par[1],
                 offset = bayesall1$Best_Par[2] )

kernfit1 <- ksvm(V17~., train1,
                 kernel= "polydot",
                 type = "C-svc",
                 kpar = polydot1,
                 prob.model=TRUE)
kernfit1

prediction1 = predict(kernfit1, test1, type="probabilities")
data1 <- as.data.frame(prediction1)
P1 <- ifelse(test1$V17 == -1, data1[ ,1], data1[ ,2])
final_percentage1 <-sum(P1)/nrow(data1)
final_percentage1
```