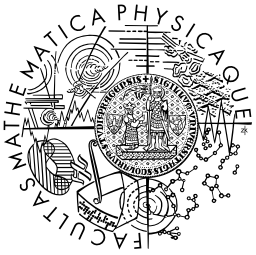


MATEMATICKO-FYZIKÁLNÍ FAKULTA
Univerzita Karlova

DIPLOMOVÁ PRÁCE

Peter Kostovčík

Bayesovská optimalizace

Katedra pravděpodobnosti a matematické statistiky

Vedoucí diplomové práce:	doc. RNDr. Petr Lachout, CSc.
Studijní program:	Matematika
Studijní obor:	Pravděpodobnost, matematická statistika a ekonometrie

Praha 2017

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Univerzita Karlova má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle §60 odst. 1 autorského zákona.

V ..... dne ..... Podpis autora

Ďakujem vedúcemu diplomovej práce doc. RNDr. Petrovi Lachoutovi, CSc. za ústretovosť, jeho cenné rady a pripomienky, ktorými mi bol nápomocný pri tvorbe tejto práce.

Název práce: Bayesovská optimalizace

Autor: Peter Kostovčík

Katedra: Katedra pravděpodobnosti a matematické statistiky

Vedoucí diplomové práce: doc. RNDr. Petr Lachout, CSc., Katedra pravděpodobnosti a matematické statistiky

Abstrakt: Optimalizácia je veľmi dôležitou súčasťou matematiky často využívanou v praxi. Existuje mnoho metód, ktoré dokážu riešiť špecifické typy účelových funkcií. Akú metódu použiť ak je funkcia neznáma a/alebo výpočtovo náročná? Jednou z odpovedí je bayesovská optimalizácia, ktorá namiesto priamej optimalizácie vytvorí pravdepodobnostný model účelovej funkcie a využije ho na zostrojenie ľahko optimalizovateľnej pomocnej funkcie. Ide o iteračný prístup, ktorý s využitím informácií z predchádzajúcich krokov nájde nový bod pre výpočet účelovej funkcie a snaží sa priblížiť optimu pri čo najmenej iteráciách. V tejto práci predstavíme bayesovskú optimalizáciu, zosumarizujeme jej rôzne prístupy v nízkych aj vysokých dimenziách a ukážeme kedy je vhodné ju použiť. Dôležitou súčasťou práce je vlastný algoritmus, ktorý aplikujeme na rôzne praktické problémy – napr. na optimalizáciu parametrov metód strojového učenia.

Klíčová slova: bayesovská optimalizácia, veľká dimenzionalita dát, redukcia dimenzie, regresia podľa gaussovského procesu

Title: Bayesian Optimization

Author: Peter Kostovčík

Department: Department of Probability and Mathematical Statistics

Supervisor: doc. RNDr. Petr Lachout, CSc., Department of Probability and Mathematical Statistics

Abstract: Optimization is an important part of mathematics and is mostly used for practical applications. For specific types of objective functions, a lot of different methods exist. A method to use when the objective is unknown and/or expensive can be difficult to determine. One of the answers is bayesian optimization, which instead of direct optimization creates a probabilistic model and uses it to constructs easily optimizable auxiliary function. It is an iterative method that uses information from previous iterations to find new point in which the objective is evaluated and tries to find the optimum within a fewer iterations. This thesis introduces bayesian optimization, sumarizes its different approaches in lower and higher dimensions and shows when to use it suitably. An important part of the thesis is my own optimization algorithm which is applied to different practical problems – e.g. parameter optimization in machine learning algorithm.

Keywords: bayesian optimization, big data set, reduction of dimension, gaussian process regression

Obsah

Úvod

Optimalizácia je v súčastnosti veľmi dôležitou oblasťou matematiky, informatiky a operačného výskumu. Úlohou je nájsť „najlepšiu“ hodnotu (optimum – minimum, resp. maximum) účelovej funkcie. Okrem optimálnej hodnoty nás ďalej zaujíma prvok v ktorom má funkcia optimum. V optimalizácii existujú rôzne prístupy v závislosti od toho aké vlastnosti má účelová funkcia. Každý prístup môže využívať inú numerickú metódu – simplex pre lineárne programovanie, Lagrangeove multiplikátory a metódy subgradientov pre konvexné programovanie a podobne. Zvoliť správnu optimalizačnú metódu je veľmi dôležité, pretože nie každá dokáže nájsť globálne optimum pre špecifické účelové funkcie. Akú optimalizačnú metódu zvoliť ak je účelová funkcia neznáma, nedokážeme ju analyticky vyjadriť alebo je výpočtovo náročná?

V dnešnej dobe už existuje veľké množstvo optimalizačných metód pre takéto účelové funkcie. V tejto práci sa budeme venovať jednej z nich – bayesovskej optimalizácii. Stručne ide o globálnu optimalizáciu neznámych funkcií bez potreby použitia gradientov. Termín bayesovská optimalizácia sa objavil už v 70. a 80. rokoch 20. storočia, avšak naozajstný potenciál tejto metódy sa objavil až s nástupom lepších výpočtových technológií. Od roku 2010 sa objavujú články o bayesovskej optimalizácii, ktoré predstavujú rôzne prístupy a skúmajú ďalšie možnosti využitia.

Práca je rozdelená na tri kapitoly. Prvé dve sú viac teoretické – najskôr podrobne predstavíme základný prístup k bayesovskej optimalizácii a potom ho rozšírime na problémy vo vyšších dimenziách. Tretia kapitola obsahuje praktické aplikácie.

V prvej kapitole podrobne opíšeme základný prístup k bayesovskej optimalizácii, t.j. s využitím gaussovských procesov. Keďže účelová funkcia môže byť neznáma a/alebo výpočtovo náročná, tak optimum by sme chceli nájsť s čo najmenším počtom iterácií (výpočtov účelovej funkcie). Pomocou gaussovských procesov vytvoríme pravdepodobnostný model pre účelovú funkciu a zadefinujeme akvizičnú funkciu, ktorá bude využívať informáciu z už vypočítaných hodnôt účelovej funkcie. Akvizičná funkcia je definovaná tak, že je výpočtovo nenáročná a diferencovateľná. Maximalizáciou akvizičnej funkcie, ktorá je značne jednoduchšia, nájdeme nový vstup pre účelovú funkciu. Ďalej predstavíme vlastnú implementáciu algoritmu pre bayesovskú optimalizáciu a pridáme niekoľko rozšírení.

Druhá kapitola obsahuje nadstavbu bayesovskej optimalizácie na problémy vo vyšších dimenziách. Problém vysokých dimenzií v našom prípade znamená vysokú dimenziu vstupu do účelovej funkcie. Väčšina numerických metód založených na gradientoch pri vyšších dimenziách zlyháva, preto máme problém optimalizovať aj akvizičnú funkciu. Predstavíme niekoľko metód na redukciu dimenzie, avšak podrobne sa budeme venovať len náhodnému preloženiu, kde zadefinujeme efektívne (aktívne) dimenzie. Budeme teda predpokladať, že niektoré dimenzie nemajú na výpočet účelovej funkcie vplyv.

V tretej kapitole využijeme teoretické znalosti z prvých dvoch kapitol a vlastný algoritmus aplikujeme na rôzne praktické problémy. Optimalizujeme parametre klasifikačnej metódy strojového učenia a aplikujeme optimalizáciu aj na neznámu funkciu – výsledok back-testu obchodnej stratégie na burze.

Zoznam použitých skratiek

Textové skratky

Skratka	Význam
BFGS	Broyden-Fletcher-Goldfarb-Shanno
BO	bayesovská optimalizácia
BUNV	bez ujmy na všeobecnosti
GP	gaussovský proces
MCMC	[Markov Chain Monte Carlo]
REMBO	bayesovská optimalizácia s náhodným preložením [Random EMbedding Bayesian Optimization]

Matematická symbolika

Symbol	Význam
$\boldsymbol{x}$	stĺpcový vektor
$1{:}n$	$\{1,\ldots,n\}$
$x_{1:n}$	$\{x_1,\ldots,x_n\}$
$\mathsf{E}(\cdot), \mathsf{P}(\cdot)$	stredná hodnota a pravdepodobnosť
$\mathcal{N}(\boldsymbol{\mu},\boldsymbol{\Sigma})$	viacrozmerné normálne rozdelenie
$\Phi(\cdot)$	distribučná funkcia rozdelenia $\mathcal{N}(0,1)$
$\phi(\cdot)$	hustota rozdelenia $\mathcal{N}(0,1)$
$\mathcal{U}(a,b)$	rovnomerné rozdelenie na intervale (a,b)
$\|\|\cdot\|\|_2$	norma v euklidovskom priestore
$A \perp\!\!\!\perp B$	nezávislé náhodné veličiny A a B

Použitý software

V práci používame voľne prístupný software Python™ (verzia 3.6.1) a nasledujúce balíčky (verzia):

- od Jones a kol. (2001–): NumPy (1.13.0) – numerické výpočty a viacrozmerné vektorové objeky, SciPy (0.19.0) – základná knižnica na vedecké výpočty a Matplotlib (1.5.3) – knižnica na tvorbu 2D a 3D obrázkov;
- od Pedregosa a kol. (2011): scikit-learn (0.18.1) – základná knižnica pre strojové učenie;
- od GPy (2012–): GPy (1.6.1) – špeciálny balíček na prácu s GP;
- od skupiny ML4AAD: RoBO (0.2.1) – nová knižnica na robustnú BO, stále vo vývoji;
- od Halls-Moore (2016): qstrader (0.0.1) – knižnica na back-testovanie a algoritmické obchodovanie na burze.

1. Efektívna globálna optimalizácia výpočtovo náročných funkcií

Úlohou globálnej optimalizácie je nájsť globálne maximum, resp. minimum nejakej funkcie. Ak je táto funkcia výpočtovo náročná, nevieme ju analyticky vyjadriť alebo vôbec nepoznáme jej tvar (ide o tzv. *čiernu skrinku [black-box]*), tak klasické metódy ako klesajúce gradienty [gradient descent] alebo náhodné prehľadávanie [random search] zlyhávajú. V tejto kapitole predstavíme bayesovskú optimalizáciu, ktorá patrí medzi metódy tzv. efektívnej globálnej optimalizácie. Výhodou tejto metódy je, že dokáže optimalizovať aj spomínané funkcie. Metóda sa snaží nájsť extrém, prípadne sa priblížiť k extrému s čo najnižším počtom iterácií. Táto vlastnosť je pre optimalizáciu výpočtovo náročných funkcií veľmi dôležitá, pretože výpočtový čas funkcie pre konkrétny parameter môže byť aj v hodinách.

1.1 Úvod do bayesovskej optimalizácie

Neznámu alebo výpočtovo náročnú funkciu $f\colon \mathcal{X} \to \mathbb{R}$, kde $\mathcal{X} \subset \mathbb{R}^d$, $d \in \mathbb{N}$, nazveme účelovou funkciou. Našou úlohou je nájsť globálne maximum funkcie f:

$$\boldsymbol{x}_{\text{opti}} = \arg\max_{\boldsymbol{x}\in\mathcal{X}} f(\boldsymbol{x}). \tag{1.1}$$

$\mathcal{X}$ nazývame aj dizajnový priestor záujmu, obvykle ide o kompaktnú množinu (nie nutne). V prípade minimalizácie f, maximalizujeme $-f$. Predpokladajme, že je možné vypočítať hodnotu funkcie f pre ľubovoľný bod $\boldsymbol{x} \in \mathcal{X}$. Tento výpočet nám môže dávať „šumom poškodené“ výsledky y také, že $\mathsf{E}\,[y|f(\boldsymbol{x})] = f(\boldsymbol{x})$. Toto je minimálna podmienka pre bayesovskú optimalizáciu. Brochu a kol. (2010) uvádzajú aj podmienku *Lipschitzovskej spojitosti*, t.j $\forall \boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathcal{X}$:

$$||f(\boldsymbol{x}_1) - f(\boldsymbol{x}_2)||_2 \leq C||\boldsymbol{x}_1 - \boldsymbol{x}_2||_2, \tag{1.2}$$

kde C je obvykle neznáma konštanta a $||\cdot||_2$ je norma v euklidovskom priestore. V ostatnej literatúre sa už táto podmienka neuvádza.

Pri bayesovskej optimalizácii sa snažíme zostrojiť nejaký pravdepodobnostný model pre účelovú funkciu, teda považujeme ju za náhodnú. Pravdepodobnostný model (rozdelenie pre vypočítané hodnoty) najskôr odhadneme *apriórnym* rozdelením. V každom kroku optimalizácie na základe vypočítaných hodnôt funkcie f apriórne rozdelenie aktualizujeme a získame *aposteriórne* rozdelenie – hľadaný model. Tento postup je totožný s bayesovkými metódami pre odhad parametra na základe pozorovaných dát, avšak namiesto odhadu parametra odhadujeme funkciu f.

Označme vstupy a vypočítané hodnoty funkcie f (realizácie) ako dáta $\mathcal{D}_n = \{(\boldsymbol{x}_i, f(\boldsymbol{x}_i))\colon i \in 1{:}n\}$. Po zozbieraní dát skombinujeme apriórnu hustotu $p(f)$ definovanú vzhľadom k nejakej σ-konečnej miere na valcovej σ-algebre s vierohodnosťou $p(\mathcal{D}_n|f)$ definovanou na nejakej σ-algebre a s využitím **bayesovej**

vety dostaneme aposteriórnu hustotu pre f:

$$p(f|\mathcal{D}_n) \propto p(\mathcal{D}_n|f)p(f), \tag{1.3}$$

kde $\propto$ znamená, že ide o rovnaké rozdelenie až na normalizačnú konštantu (Brochu a kol., 2010). Ako apriórne rozdelenie sa najčastejšie používa gaussovský proces, ktorý predstavíme v časti 1.2.

Na základe aktuálneho aposteriórneho rozdelenia (po n výpočtoch funkcie f) potrebujeme rozhodnúť o novom bode $\boldsymbol{x}_*$ tak, aby sme maximalizovali získanú informáciu a vypočítali novú hodnotu $f(\boldsymbol{x}_*)$. Vyberieme vhodnú úžitkovú funkciu a budeme optimalizovať jej strednú hodnotu vzhľadom k aposteriórnemu rozdeleniu funkcie f. Táto optimalizácia je jednoduchšia pretože volíme ľahko spočítateľnú úžitkovú funkciu. Túto strednú hodnotu nazývame *akvizičná funkcia* [acquisition function] a podrobne sa jej budeme venovať v časti 1.3.

Algoritmus 1 základný algoritmus BO

1: inicializuj prvý bod $\boldsymbol{x}_1$ a vypočítaj $y_1 = f(\boldsymbol{x}_1)$, $\mathcal{D}_1 = \{(\boldsymbol{x}_1,y_1)\}$
2: **for** $n = 1,2,\ldots$ **do**
3: vyber nové $\boldsymbol{x}_{n+1}$ maximalizáciou akvizičnej funkcie $\alpha(\boldsymbol{x},\mathcal{D}_n)$
4: vypočítaj hodnotu $y_{n+1} = f(\boldsymbol{x}_{n+1})$
5: rozšír dáta $\mathcal{D}_{n+1} = \mathcal{D}_n \cup (\boldsymbol{x}_{n+1},y_{n+1})$
6: aktualizuj pravdepodobnostný model

1.2 Gaussovské procesy

Definícia 1.1 (**gaussovský proces – GP**). *Hovoríme, že náhodná funkcia $f\colon \mathcal{X} \to \mathbb{R}$ je gaussovský proces charakterizovaný strednou hodnotou $\mu_0\colon \mathcal{X} \to \mathbb{R}$ a pozitívne definítnou kovariančnou funkciou $\kappa\colon \mathcal{X}\times\mathcal{X} \to \mathbb{R}$, ktorá je spojitá na diagonále (t.j. v $\boldsymbol{x} = \boldsymbol{x}'$ pre spojité premenné $\boldsymbol{x},\boldsymbol{x}' \in \mathcal{X}$), ak pre ľubovoľné $N \in \mathbb{N}$ a pre ľubovoľnú $\boldsymbol{x}_{1:N} \subset \mathcal{X}$ má vektor $(f(\boldsymbol{x}_1),\ldots,f(\boldsymbol{x}_N))^\top$ združené normálne rozdelenie so strednou hodnotou μ_0 a kovariančnou maticou $\boldsymbol{\Sigma}$, kde $\Sigma_{ij} = \kappa(\boldsymbol{x}_i,\boldsymbol{x}_j)$. Označujeme $f \sim \mathcal{GP}(\mu_0,\kappa)$.*

Majme vypočítaných n hodnôt funkcie f. Keďže ďalej predpokladáme, že f je gaussovský proces, tak vypočítané hodnoty sú realizácie GP. Za apriórne rozdelenie pre f vyberieme GP so zvolenou funkciou strednej hodnoty μ_0 a kovariančnou funkciou κ. Označme $y_i = f(\boldsymbol{x}_i)$ ako realizáciu GP f v bode $\boldsymbol{x}_i \in \mathcal{X}$, $i \in 1{:}n$ a $\boldsymbol{X} = (\boldsymbol{x}_1,\ldots,\boldsymbol{x}_n)^\top$ ako dizajnovú maticu $n \times d$. Rozdelenie pre funkciu môže znieť zložito, v praxi však ide o rozdelenie pre funkčné hodnoty.

Majme ľubovoľné pevné $\boldsymbol{x}_* \in \mathcal{X}$, z definície (1.1) máme existenciu združeného rozdelenia pre $(f(\boldsymbol{x}_1),\ldots,f(\boldsymbol{x}_n),f(\boldsymbol{x}_*))^\top$. Ďalej už pracujeme s viacrozmerným normálnym rozdelením, teda zo združeného rozdelenia vieme vyjadriť podmienené rozdelenie pre $f(\boldsymbol{x}_*)$. Ak zoberieme konkrétnu podmienku $(f(\boldsymbol{x}_1),\ldots,f(\boldsymbol{x}_n))^\top = (y_1,\ldots,y_n)^\top$, tak získame podmienené aposteriórne rozdelenie – hľadaný pravdepodobnostný model. Ide o tzv. regresiu podľa gaussovského procesu.

Predtým než sa posunieme k regresii podľa GP, ešte odvodíme niektoré potrebné vlastnosti z viacrozmerného normálneho rozdelenia.

Lemma 1.1. *(niektoré z vlastností blokových matíc) Nech* $\boldsymbol{A}_{n\times n}$, $\boldsymbol{B}_{n\times m}$, $\boldsymbol{C}_{m\times n}$, $\boldsymbol{D}_{m\times m}$ *sú bloky matice* $\boldsymbol{M}_{(n+m)\times(n+m)}$*:*

$$\boldsymbol{M} = \begin{pmatrix} \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{C} & \boldsymbol{D} \end{pmatrix},$$

matice $\boldsymbol{A}$ *a* $\boldsymbol{D}$ *sú regulárne. Potom platí:*

$$\boldsymbol{M}^{-1} = \begin{pmatrix} \boldsymbol{I}_n & \boldsymbol{0} \\ -\boldsymbol{D}^{-1}\boldsymbol{C} & \boldsymbol{I}_m \end{pmatrix} \cdot \begin{pmatrix} (\boldsymbol{M}/\boldsymbol{D})^{-1} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{D}^{-1} \end{pmatrix} \cdot \begin{pmatrix} \boldsymbol{I}_n & -\boldsymbol{B}\boldsymbol{D}^{-1} \\ \boldsymbol{0} & \boldsymbol{I}_m \end{pmatrix}, \tag{1.4}$$

$$|\boldsymbol{M}| = |\boldsymbol{M}/\boldsymbol{D}||\boldsymbol{D}|, \tag{1.5}$$

kde $\boldsymbol{M}/\boldsymbol{D} = \boldsymbol{A} - \boldsymbol{B}\boldsymbol{D}^{-1}\boldsymbol{C}$ *je Schurov komplement matice* $\boldsymbol{M}$ *vzhľadom k* $\boldsymbol{D}$ *(Murphy, 2012).*

Dôkaz. Maticu $\boldsymbol{M}$ upravíme tak aby bola po blokoch diagonálna. Najskôr vynulujeme pravý horný blok $\boldsymbol{B}$:

$$\underbrace{\begin{pmatrix} \boldsymbol{I}_n & -\boldsymbol{B}\boldsymbol{D}^{-1} \\ \boldsymbol{0} & \boldsymbol{I}_m \end{pmatrix}}_{\boldsymbol{X}} \cdot \begin{pmatrix} \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{C} & \boldsymbol{D} \end{pmatrix} = \underbrace{\begin{pmatrix} \boldsymbol{A} - \boldsymbol{B}\boldsymbol{D}^{-1}\boldsymbol{C} & \boldsymbol{0} \\ \boldsymbol{C} & \boldsymbol{D} \end{pmatrix}}_{\boldsymbol{U}},$$

podobne blok $\boldsymbol{C}$:

$$\begin{pmatrix} \boldsymbol{A} - \boldsymbol{B}\boldsymbol{D}^{-1}\boldsymbol{C} & \boldsymbol{0} \\ \boldsymbol{C} & \boldsymbol{D} \end{pmatrix} \cdot \underbrace{\begin{pmatrix} \boldsymbol{I}_n & \boldsymbol{0} \\ -\boldsymbol{D}^{-1}\boldsymbol{C} & \boldsymbol{I}_m \end{pmatrix}}_{\boldsymbol{Y}} = \underbrace{\begin{pmatrix} \boldsymbol{A} - \boldsymbol{B}\boldsymbol{D}^{-1}\boldsymbol{C} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{D} \end{pmatrix}}_{\boldsymbol{W}}.$$

Spojením predchádzajúcich odvodení dostaneme $\boldsymbol{X}\boldsymbol{M}\boldsymbol{Y} = \boldsymbol{W}$ a ak invertujeme obidve strany máme:

$$\boldsymbol{Y}^{-1}\boldsymbol{M}^{-1}\boldsymbol{X}^{-1} = \boldsymbol{W}^{-1} \iff \boldsymbol{M}^{-1} = \boldsymbol{Y}\boldsymbol{W}^{-1}\boldsymbol{X},$$

čím sme dokázali (1.4). Z predchádzajúceho máme $\boldsymbol{X}\boldsymbol{M} = \boldsymbol{U}$ a z vlastnosti determinantu vyplýva:

$$|\boldsymbol{X}\boldsymbol{M}| = |\boldsymbol{X}||\boldsymbol{M}| = 1 \cdot |\boldsymbol{M}|,$$

$$|\boldsymbol{M}| = |\boldsymbol{U}| = \begin{vmatrix} \boldsymbol{M}/\boldsymbol{D} & \boldsymbol{0} \\ \boldsymbol{C} & \boldsymbol{D} \end{vmatrix} = |\boldsymbol{M}/\boldsymbol{D}||\boldsymbol{D}|.$$

□

Veta 1.2. *Nech vektor* $\boldsymbol{x} = \begin{pmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \end{pmatrix} \in \mathbb{R}^{n_1+n_2}$ *má združené normálne rozdelenie s parametrami:*

$$\boldsymbol{\mu} = \begin{pmatrix} \boldsymbol{\mu}_1 \\ \boldsymbol{\mu}_2 \end{pmatrix}, \quad \boldsymbol{\Sigma} = \begin{pmatrix} \boldsymbol{\Sigma}_{11} & \boldsymbol{\Sigma}_{12} \\ \boldsymbol{\Sigma}_{21} & \boldsymbol{\Sigma}_{22} \end{pmatrix}.$$

Potom pre podmienené aposteriórne rozdelenie platí:

$$\begin{aligned} \boldsymbol{x}_1|\boldsymbol{x}_2 &\sim \mathcal{N}(\boldsymbol{\mu}_{1|2}, \boldsymbol{\Sigma}_{1|2}), \\ \boldsymbol{\mu}_{1|2} &= \boldsymbol{\mu}_1 + \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}(\boldsymbol{x}_2 - \boldsymbol{\mu}_2), \\ \boldsymbol{\Sigma}_{1|2} &= \boldsymbol{\Sigma}_{11} - \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}\boldsymbol{\Sigma}_{21} \end{aligned}$$

(Murphy, 2012).

Dôkaz. Rozložením združenej hustoty $p(\boldsymbol{x}_1,\boldsymbol{x}_2)$ na tvar $p(\boldsymbol{x}_1|\boldsymbol{x}_2)p(\boldsymbol{x}_2)$ získame parametre pre podmienené rozdelenie.

$$p(\boldsymbol{x}_1,\boldsymbol{x}_2) = (2\pi)^{-\frac{n_1+n_2}{2}} |\boldsymbol{\Sigma}|^{-\frac{1}{2}} \exp\left\{-\frac{1}{2}\boldsymbol{A}\right\},$$

$$\boldsymbol{A} = \begin{pmatrix} \boldsymbol{x}_1 - \boldsymbol{\mu}_1 \\ \boldsymbol{x}_2 - \boldsymbol{\mu}_2 \end{pmatrix}^\top \begin{pmatrix} \boldsymbol{\Sigma}_{11} & \boldsymbol{\Sigma}_{12} \\ \boldsymbol{\Sigma}_{21} & \boldsymbol{\Sigma}_{22} \end{pmatrix}^{-1} \begin{pmatrix} \boldsymbol{x}_1 - \boldsymbol{\mu}_1 \\ \boldsymbol{x}_2 - \boldsymbol{\mu}_2 \end{pmatrix}.$$

S využitím (1.4) pre inverznú kovariančnú maticu rozpíšeme maticu $\boldsymbol{A}$:

$$\boldsymbol{A} = \begin{pmatrix} \boldsymbol{x}_1 - \boldsymbol{\mu}_1 \\ \boldsymbol{x}_2 - \boldsymbol{\mu}_2 \end{pmatrix}^\top \begin{pmatrix} \boldsymbol{I}_{n_1} & \boldsymbol{0} \\ -\boldsymbol{\Sigma}_{22}^{-1}\boldsymbol{\Sigma}_{21} & \boldsymbol{I}_{n_2} \end{pmatrix} \begin{pmatrix} (\boldsymbol{\Sigma}/\boldsymbol{\Sigma}_{22})^{-1} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{\Sigma}_{22}^{-1} \end{pmatrix} \cdot \begin{pmatrix} \boldsymbol{I}_{n_1} & -\boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1} \\ \boldsymbol{0} & \boldsymbol{I}_{n_2} \end{pmatrix} \begin{pmatrix} \boldsymbol{x}_1 - \boldsymbol{\mu}_1 \\ \boldsymbol{x}_2 - \boldsymbol{\mu}_2 \end{pmatrix}.$$

Po vynásobení prvých dvoch matíc dostaneme:

$$\left((\boldsymbol{x}_1 - \boldsymbol{\mu}_1)^\top - (\boldsymbol{x}_2 - \boldsymbol{\mu}_2)^\top \boldsymbol{\Sigma}_{22}^{-1}\boldsymbol{\Sigma}_{21} \ , \ (\boldsymbol{x}_2 - \boldsymbol{\mu}_2)^\top\right), \tag{1.6}$$

podobne pri posledných dvoch:

$$\begin{pmatrix} \boldsymbol{x}_1 - \boldsymbol{\mu}_1 - \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}(\boldsymbol{x}_2 - \boldsymbol{\mu}_2) \\ \boldsymbol{x}_2 - \boldsymbol{\mu}_2 \end{pmatrix}.$$

Prepisom (1.6) do stĺpcového tvaru a využitím symetrickosti kovariančnej matice $\boldsymbol{\Sigma}$, t.j. $\boldsymbol{\Sigma}_{21}^\top = \boldsymbol{\Sigma}_{12}$ máme:

$$\boldsymbol{A} = \begin{pmatrix} \boldsymbol{x}_1 - \boldsymbol{\mu}_1 - \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}(\boldsymbol{x}_2 - \boldsymbol{\mu}_2) \\ \boldsymbol{x}_2 - \boldsymbol{\mu}_2 \end{pmatrix}^\top \begin{pmatrix} (\boldsymbol{\Sigma}/\boldsymbol{\Sigma}_{22})^{-1} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{\Sigma}_{22}^{-1} \end{pmatrix} \cdot \begin{pmatrix} \boldsymbol{x}_1 - \boldsymbol{\mu}_1 - \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}(\boldsymbol{x}_2 - \boldsymbol{\mu}_2) \\ \boldsymbol{x}_2 - \boldsymbol{\mu}_2 \end{pmatrix}.$$

Odkial je jasne viditeľné rozdelenie $\boldsymbol{A} = \boldsymbol{A}_1(\boldsymbol{x}_1,\boldsymbol{x}_2) + \boldsymbol{A}_2(\boldsymbol{x}_2)$, kde:

$$\begin{aligned} \boldsymbol{A}_1(\boldsymbol{x}_1,\boldsymbol{x}_2) &= (\boldsymbol{x}_1 - \boldsymbol{\mu}_1 - \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}(\boldsymbol{x}_2 - \boldsymbol{\mu}_2))^\top (\boldsymbol{\Sigma}/\boldsymbol{\Sigma}_{22})^{-1} \\ &\quad \cdot (\boldsymbol{x}_1 - \boldsymbol{\mu}_1 - \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}(\boldsymbol{x}_2 - \boldsymbol{\mu}_2)), \\ \boldsymbol{A}_2(\boldsymbol{x}_2) &= (\boldsymbol{x}_2 - \boldsymbol{\mu}_2)^\top \boldsymbol{\Sigma}_{22}^{-1}(\boldsymbol{x}_2 - \boldsymbol{\mu}_2). \end{aligned}$$

V $\boldsymbol{A}_1$ máme kvadratickú formu $\boldsymbol{x}_1, \boldsymbol{x}_2$ a v $\boldsymbol{A}_2$ kvadratickú formu $\boldsymbol{x}_2$, t.j. exponenciálnu funkciu v $p(\boldsymbol{x}_1,\boldsymbol{x}_2)$ môžeme napísať v tvare:

$$\exp\left(-\frac{1}{2}\boldsymbol{A}_1(\boldsymbol{x}_1,\boldsymbol{x}_2)\right) \exp\left(-\frac{1}{2}\boldsymbol{A}_2(\boldsymbol{x}_2)\right).$$

Pre normalizačnú konštantu využijeme vzťah (1.5) z Lemma 1.1 $|\boldsymbol{\Sigma}| = |\boldsymbol{\Sigma}/\boldsymbol{\Sigma}_{22}| \cdot |\boldsymbol{\Sigma}_{22}|$ a platí:

$$\begin{aligned} (2\pi)^{-\frac{n_1+n_2}{2}} |\boldsymbol{\Sigma}|^{-\frac{1}{2}} &= (2\pi)^{-\frac{n_1+n_2}{2}} (|\boldsymbol{\Sigma}/\boldsymbol{\Sigma}_{22}||\boldsymbol{\Sigma}_{22}|)^{-\frac{1}{2}} \\ &= (2\pi)^{-\frac{n_1}{2}} |\boldsymbol{\Sigma}/\boldsymbol{\Sigma}_{22}|^{-\frac{1}{2}} (2\pi)^{-\frac{n_2}{2}} |\boldsymbol{\Sigma}_{22}|^{-\frac{1}{2}}. \end{aligned}$$

Tým sme rozložili $p(\boldsymbol{x}_1,\boldsymbol{x}_2)$ na $p(\boldsymbol{x}_1|\boldsymbol{x}_2)p(\boldsymbol{x}_2)$ a z $\boldsymbol{A}_1(\boldsymbol{x}_1,\boldsymbol{x}_2)$ určíme $\boldsymbol{\mu}_{1|2}$ a $\boldsymbol{\Sigma}_{1|2}$. Podobne pre rozdelenie $\boldsymbol{x}_2|\boldsymbol{x}_1$.

□

1.2.1 Regresia podľa GP

Pri hľadaní aposteriórneho rozdelenia funkcie f sa snažíme náhodnú funkciu, resp. jej strednú hodnotu prispôsobiť pozorovaným hodnotám, preto sa používa názov regresia. Za apriórne rozdelenie funkcie f volíme GP:

$$f(\boldsymbol{x}) \sim \mathcal{GP}(\mu_0(\boldsymbol{x}),\kappa(\boldsymbol{x},\boldsymbol{x}))$$

kde $\mu_0(\boldsymbol{x})$ a $\kappa(\boldsymbol{x},\boldsymbol{x}')$ sú vhodne zvolené funkcie (podobne ako pri klasickej bayesovskej inferencii, určujeme parametre apriórneho rozdelenia ako vhodne zvolené konštanty, resp. náhodné veličiny s nejakým rozdelením). Zvyčajne sa volí konštantné $\mu_0(\cdot) = \mu_0 \in \mathbb{R}$ a $\kappa(\cdot,\cdot)$ ako pozitívne definitne jadro.

Definícia 1.2. *Nech $\mathcal{X} \subset \mathbb{R}^d$ je neprázdna. Symetrická funkcia $k\colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ sa nazýva pozitívne definitne jadro [kernel] na $\mathcal{X}$ ak $\forall N \in \mathbb{N}$, $\forall \boldsymbol{x}_1,\ldots,\boldsymbol{x}_N \in \mathcal{X}$ také, že $k(\boldsymbol{x}_i,\boldsymbol{x}_j) \neq 0$ pre $i \neq j$ a $\forall c_1,\ldots,c_N \in \mathbb{R} \setminus \{0\}$ platí:*

$$\sum_{i=1}^{N}\sum_{j=1}^{N} c_i c_j k(\boldsymbol{x}_i,\boldsymbol{x}_j) > 0.$$

Predpokladajme, že výpočet funkcie f nám dáva „šumom poškodené“ pozorovania, t.j. $y_i = f(\boldsymbol{x}_i) + \varepsilon_i$, kde $\varepsilon_i \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0,\sigma_y^2)$ $\forall i \in 1{:}n$ a $f(\boldsymbol{x}_j) \perp\!\!\!\perp \varepsilon_i$ $\forall i,j \in 1{:}n$. Kovariancia zašumenej odozvy je

$$\operatorname{cov}(y_i,y_j) = \kappa(\boldsymbol{x}_i,\boldsymbol{x}_j) + \mathbb{1}_{[i=j]}\sigma_y^2.$$

Ak označíme $\boldsymbol{K} = (\kappa(\boldsymbol{x}_i,\boldsymbol{x}_j))_{i,j=1}^{n}$ máme:

$$\operatorname{var}(\boldsymbol{y}|\boldsymbol{X}) = \boldsymbol{K} + \sigma_y^2\boldsymbol{I}_n = \boldsymbol{K}_y. \tag{1.7}$$

Chceme urobiť predikciu pre nové dáta $(\boldsymbol{X}_*, \boldsymbol{y}_*)$. Z definície 1.1 máme združené rozdelenie:

$$\begin{pmatrix}\boldsymbol{y}\\ \boldsymbol{y}_*\end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix}\mu_0(\boldsymbol{X})\\ \mu_0(\boldsymbol{X}_*)\end{pmatrix}, \begin{pmatrix}\boldsymbol{K}_y & \boldsymbol{K}_*\\ \boldsymbol{K}_*^\top & \boldsymbol{K}_{**}\end{pmatrix}\right), \tag{1.8}$$

kde $\mu_0(\boldsymbol{X}) = (\mu_0(\boldsymbol{x}_1),\ldots,\mu_0(\boldsymbol{x}_n))^\top$ je zvolená funkcia strednej hodnoty z apriórneho rozdelenia, $\boldsymbol{K}_* = \kappa(\boldsymbol{X},\boldsymbol{X}_*)$[1] je $n \times n_*$, $\boldsymbol{K}_{**} = (\kappa(\boldsymbol{x}_{*i},\boldsymbol{x}_{*j}))_{i,j}^{n_*}$ a $\kappa(\boldsymbol{x},\boldsymbol{x}')$ je zvolená kovariančná funkcia. S využitím vety 1.2 je podmienené aposteriórne rozdelenie typu:

$$\begin{aligned}\boldsymbol{y}_*|\boldsymbol{X}_*,\boldsymbol{X},\boldsymbol{y} &\sim \mathcal{N}(\boldsymbol{y}_*|\boldsymbol{\mu}_*,\boldsymbol{\Sigma}_*),\\ \boldsymbol{\mu}_* &= \mu_0(\boldsymbol{X}_*) + \boldsymbol{K}_*^\top\boldsymbol{K}_y^{-1}(\boldsymbol{y} - \mu_0(\boldsymbol{X})),\\ \boldsymbol{\Sigma}_* &= \boldsymbol{K}_{**} - \boldsymbol{K}_*^\top\boldsymbol{K}_y^{-1}\boldsymbol{K}_*.\end{aligned} \tag{1.9}$$

Poznámka. Ak v (1.9) dosadíme za nové dáta $\boldsymbol{X}_*$ pôvodné dáta $\boldsymbol{X}$ a budeme uvažovať výsledky bez šumu $\sigma_y^2 = 0$, tak $\boldsymbol{\mu}_* = \boldsymbol{y} = (f(\boldsymbol{x}_1),\ldots,f(\boldsymbol{x}_n))^\top$ a $\boldsymbol{\Sigma}_* = \boldsymbol{0}$, teda dostaneme deterministické hodnoty $f(\boldsymbol{x}_1),\ldots,f(\boldsymbol{x}_n)$ (viď obr. 1.1).

[1] Označenie $\kappa(\boldsymbol{X},\boldsymbol{X}_*)$ je skrátené $(\kappa(\boldsymbol{x}_i,\boldsymbol{x}_{*j}))_{i,j}$ $\forall i \in 1{:}n$ a $\forall j \in 1{:}n_*$.

Často používané pozitívne definítne jadra (Murphy, 2012):

- **SE** jadro – štvorcové exponenciálne jadro alebo gaussovské jadro:

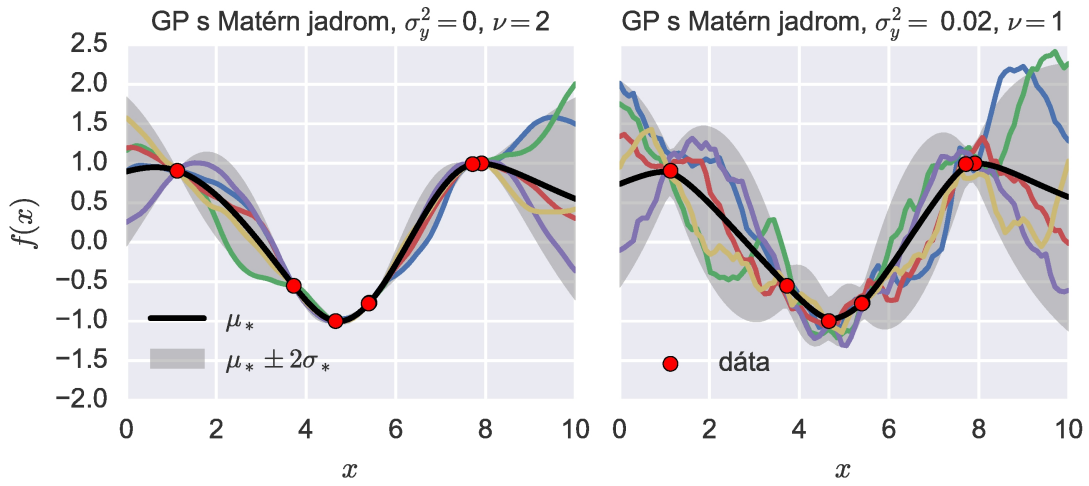
$$k(\boldsymbol{x},\boldsymbol{x}') = \theta_0 \exp\left(-(\boldsymbol{x}-\boldsymbol{x}')\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{x}')^\top\right),$$

  kde $\theta_0 > 0$ je parameter rozptylu a $\boldsymbol{\Sigma}$ je pozitívne definítna matica parametrov mierky, najčastejšie diagonálna. V prípade $\boldsymbol{\Sigma} = \ell^2$ pre $\ell > 0$ a $\theta_0 = 1$ ide o **RBF** jadro [radial basic function]. Často sa používa $\theta_0 = 1$, pretože mierka je postačujúca.

- **Matérn**ové jadro – najčastejšie používané v GP:

$$k(\boldsymbol{x},\boldsymbol{x}') = \frac{2^{1-\nu}}{\Gamma(\nu)}\left(\frac{\sqrt{2\nu||\boldsymbol{x}-\boldsymbol{x}'||_2}}{\ell}\right)^{\nu} K_\nu\left(\frac{\sqrt{2\nu||\boldsymbol{x}-\boldsymbol{x}'||_2}}{\ell}\right),$$

  kde $\nu > 0$ je parameter „hladkosti“, $\ell > 0$ je parameter mierky a $K_\nu(\cdot)$ je modifikovaná Besselová funkcia.



Obrázok 1.1: Ukážka regresie podľa GP (podmieneného GP) s Matérnovým jadrom s hyper-parametrami $\ell = 1$ a rôznymi ν. Regresia je z dát – $x_{1:7}$ vygenerovaných z $\mathcal{U}(0{,}10)$, $f = \sin$. Okrem strednej hodnoty a 95% predikčného pásu máme 5 vygenerovaných realizácií podmieneného GP. Vľavo máme výsledky bez šumu a pravo so šumom $\sigma_y^2 = 0.02$. Inšpirácia k obrázku od Murphy (2012).

V regresii podľa GP sa musí zvážiť nasledujúce:

- pri apriórnom rozdelení sa najčastejšie volí $\mu_0 = 0$, pretože GP je dostatočne flexibilný na modelovanie strednej hodnoty μ_* (z 1.9).
- Pri voľbe kovariančnej funkcie κ sa najčastejšie používa nejaké pozitívne definítne jadro.
- Ako určiť vhodné hyper-parametre v jadrových funkciách (k tejto téme sa vrátime v časti 1.3).
- Používame stacionárny gaussovský proces ako apriórne rozdelenie.

Nestacionárny GP pre apriórne rozdelenie

Hlavným predpokladom v regresii podľa GP je stacionarita procesu. V praxi to znamená, že jadro $k(\boldsymbol{x},\boldsymbol{x}')$ je funkciou $\boldsymbol{x}-\boldsymbol{x}'$. Pri reálnych problémoch často pozorujeme, že pôvodný proces (hodnoty f) je nestacionárny. Použitím stacionárneho GP je zle nastavené apriórne rozdelenie a k vytvoreniu vhodného aposteriórneho rozdelenia potrebujeme viac dát. V BO sa však snažíme nájsť optimum s čo najmenším počtom bodov. Shahriari a kol. (2016) uvádzajú dve základné metódy, ktoré si dokážu poradiť s nestacionaritou.

Prvou možnosťou je použiť tzv. nestacionárne jadro. V praxi to znamená transformáciu priestoru $\mathcal{X}$ pomocou *baliacej funkcie* w. Nech $\boldsymbol{x} = (x_1,\dots,x_d)^\top$, $w(\boldsymbol{x}) = (w_1(\boldsymbol{x}),\dots,w_d(\boldsymbol{x}))^\top$ a funkciu w sa snažíme voliť tak aby bol proces na transformovanom priestore stacionárny. Ak $\mathcal{X} = [0,1]^d$, prípadne vieme priestor $\mathcal{X}$ ľahko transformovať do jednotkovej hyper-kocky, tak w môžeme voliť ako kumulatívnu distribučnú funkciu beta rozdelenia:

$$w_j(\boldsymbol{x}) = \frac{1}{\mathrm{B}(a,b)} \int_0^{x_j} s^{a-1}(1-s)^{b-1}\mathrm{d}s \quad \forall j \in 1{:}d,$$

kde a, b sú nové hyper-parametre a $\mathrm{B}(a,b)$ je beta funkcia.

Alternatívou je *partícia* priestoru $\mathcal{X}$ tak, že v každom regióne modelujeme samostatný stacionárny proces.

1.3 Akvizičné funkcie

V predchádzajúcej časti sme sa venovali zostrojeniu pravdepodobnostného modelu pre úžitkovú funkciu f pri n-tej iterácií. Teraz opíšeme spôsob výberu nových vstupov $\boldsymbol{x}$, pre ktoré sa vypočíta hodnota funkcie f. Táto časť je pri BO veľmi dôležitá, pretože sa snažíme vypočítať extrém s čo najnižším počtom iterácií a teda musíme vyberať body, ktoré nám poskytnú čo najlepšiu informáciu, z ktorých budeme mať najlepší úžitok.

Uvažujme úžitkovú funkciu $U\colon \mathbb{R}^d \times \mathbb{R} \times \boldsymbol{\Theta} \to \mathbb{R}$, ktorej vstupy sú $\boldsymbol{x} \in \mathcal{X} \subset \mathbb{R}^d$, $y = f(\boldsymbol{x}) \in \mathbb{R}$ a hyper-parametre $\boldsymbol{\theta} \in \boldsymbol{\Theta}$ z konštrukcie GP. Na základe už získaných dát $\mathcal{D}_n$ z predchádzajúcich iterácií a daných hyper-parametroch $\boldsymbol{\theta}$ môžeme získať očakávaný úžitok v bode $\boldsymbol{x}$, ktorý nazývame akvizičná funkcia:

$$\alpha(\boldsymbol{x};\mathcal{D}_n) = \mathsf{E}_{\boldsymbol{\theta}}\,\mathsf{E}_{y|\boldsymbol{x},\boldsymbol{\theta}}[U(\boldsymbol{x},y,\boldsymbol{\theta})] \tag{1.10}$$

(Shahriari a kol., 2016). Pre zjednodušenie budeme hyper-parametre $\boldsymbol{\theta}$ považovať za pevne dané, t.j použijeme $\alpha(\boldsymbol{x};\boldsymbol{\theta},\mathcal{D}_n) = \mathsf{E}_{y|\boldsymbol{x},\boldsymbol{\theta}}[U(\boldsymbol{x},y,\boldsymbol{\theta})]$ a k ich odhadu sa dostaneme v časti 1.3.2. Ak máme určenú akvizičnú funkciu, tak nový bod $\boldsymbol{x}_{n+1}$ vypočítame z maxima akvizičnej funkcie:

$$\boldsymbol{x}_{n+1} = \arg\max_{\boldsymbol{x}\in\mathcal{X}} \alpha(\boldsymbol{x};\boldsymbol{\theta},\mathcal{D}_n). \tag{1.11}$$

Postupne si predstavíme najpoužívanejšie akvizičné funkcie. Shahriari a kol. (2016) spísali ucelený prehľad rôznych akvizičných funkcií a rozdelili ich na tri základné skupiny:

- postup založený na zlepšení maxima,

- optimistický prístup,
- postup založený na teórii informácie.

Brochu a kol. (2010) podrobnejšie opísali motiváciu a vznik prvých dvoch skupín.

Postup založený na zlepšení

Opäť predpokladajme, že máme vypočítaných n hodnôt funkcie f, t.j. sme v $n+1$ iterácii. Označme $\boldsymbol{x}^+ = \arg\max_{\boldsymbol{x}\in\boldsymbol{x}_{1:n}} f(\boldsymbol{x})$ a $y^+ = f(\boldsymbol{x}^+)$ ako maximum po n iteráciách. Chceme zlepšiť súčasné maximum a zaujímame sa iba o hodnoty $y = f(\boldsymbol{x})$ väčšie ako y^+ pri nových vstupoch $\boldsymbol{x}$. Najjednoduchší prístup je zadefinovať účelovú funkciu ako indikátor, t.j. $U(\boldsymbol{x},y,\boldsymbol{\theta}) = \mathbb{1}_{[y>y^+]}$. Akvizičná funkcia je potom pravdepodobnosťou:

$$\alpha_{\mathrm{PI}}(\boldsymbol{x};\boldsymbol{\theta},\mathcal{D}_n) = \mathsf{E}_{\,y|\boldsymbol{x},\boldsymbol{\theta}}\,\mathbb{1}_{[y>y^+]} = \mathsf{P}_{y|\boldsymbol{x},\boldsymbol{\theta}}(y > y^+) = \Phi\left(\frac{\mu(\boldsymbol{x}) - y^+}{\sigma(\boldsymbol{x})}\right), \tag{1.12}$$

ktorú nazývame pravdepodobnosť zlepšenia [probability of improvement] a $\alpha_{\mathrm{PI}} = 0$ pre $\sigma(\boldsymbol{x}) = 0$. Ide o pravdepodobnosť z podmieneného aposteriórneho rozdelenia (1.9), kde $y = f(\boldsymbol{x})$ je náhodná veličina (hodnota funkcie) z gaussovského procesu. Stredná hodnota $\mu(\boldsymbol{x})$ a štandardná odchýlka $\sigma(\boldsymbol{x}) > 0$ majú z (1.9) tvar:

$$\begin{aligned} \mu(\boldsymbol{x}) &= \boldsymbol{k}^\top \boldsymbol{K}_y^{-1}\boldsymbol{y}, \\ \sigma^2(\boldsymbol{x}) &= \kappa(\boldsymbol{x},\boldsymbol{x}) - \boldsymbol{k}^\top \boldsymbol{K}_y^{-1}\boldsymbol{k}, \end{aligned} \tag{1.13}$$

kde $\boldsymbol{K}_y$ je kovariančná matica pre $\boldsymbol{y}$ z (1.7), $\boldsymbol{y} = (f(\boldsymbol{x}_1),\ldots,f(\boldsymbol{x}_n))^\top$ a $\boldsymbol{k} = (\kappa(\boldsymbol{x},\boldsymbol{x}_1),\ldots,\kappa(\boldsymbol{x},\boldsymbol{x}_n))^\top$ pre zvolenú kovariančnú funkciu κ. Φ je distribučná funkcia rozdelenia $\mathcal{N}(0,1)$.

Ďalšia akvizičná funkcia je rozšírením pravdepodobnosti zlepšenia a nazýva sa očakávane zlepšenie [expected improvement]. Úžitková funkcia U, označovaná aj ako zlepšenie I má tvar $I(\boldsymbol{x},y,\boldsymbol{\theta}) = \max\{0,\, y - y^+\}$. Akvizičná funkcia je potom horným parciálnym momentom:

$$\alpha_{\mathrm{EI}}(\boldsymbol{x};\boldsymbol{\theta},\mathcal{D}_n) = \mathsf{E}_{\,y|\boldsymbol{x},\boldsymbol{\theta}} \max\{0,\, y - y^+\}. \tag{1.14}$$

Tvrdenie 1.3. *Pre očakávané zlepšenie platí:*

$$\alpha_{\mathrm{EI}}(\boldsymbol{x};\boldsymbol{\theta},\mathcal{D}_n) = (\mu(\boldsymbol{x}) - y^+)\Phi\left(\frac{\mu(\boldsymbol{x}) - y^+}{\sigma(\boldsymbol{x})}\right) + \sigma(\boldsymbol{x})\phi\left(\frac{\mu(\boldsymbol{x}) - y^+}{\sigma(\boldsymbol{x})}\right), \tag{1.15}$$

pre $\sigma(\boldsymbol{x}) > 0$ *a* $\alpha_{\mathrm{EI}} = 0$ *pre* $\sigma(\boldsymbol{x}) = 0$, *kde* ϕ *je hustota* $\mathcal{N}(0,1)$.

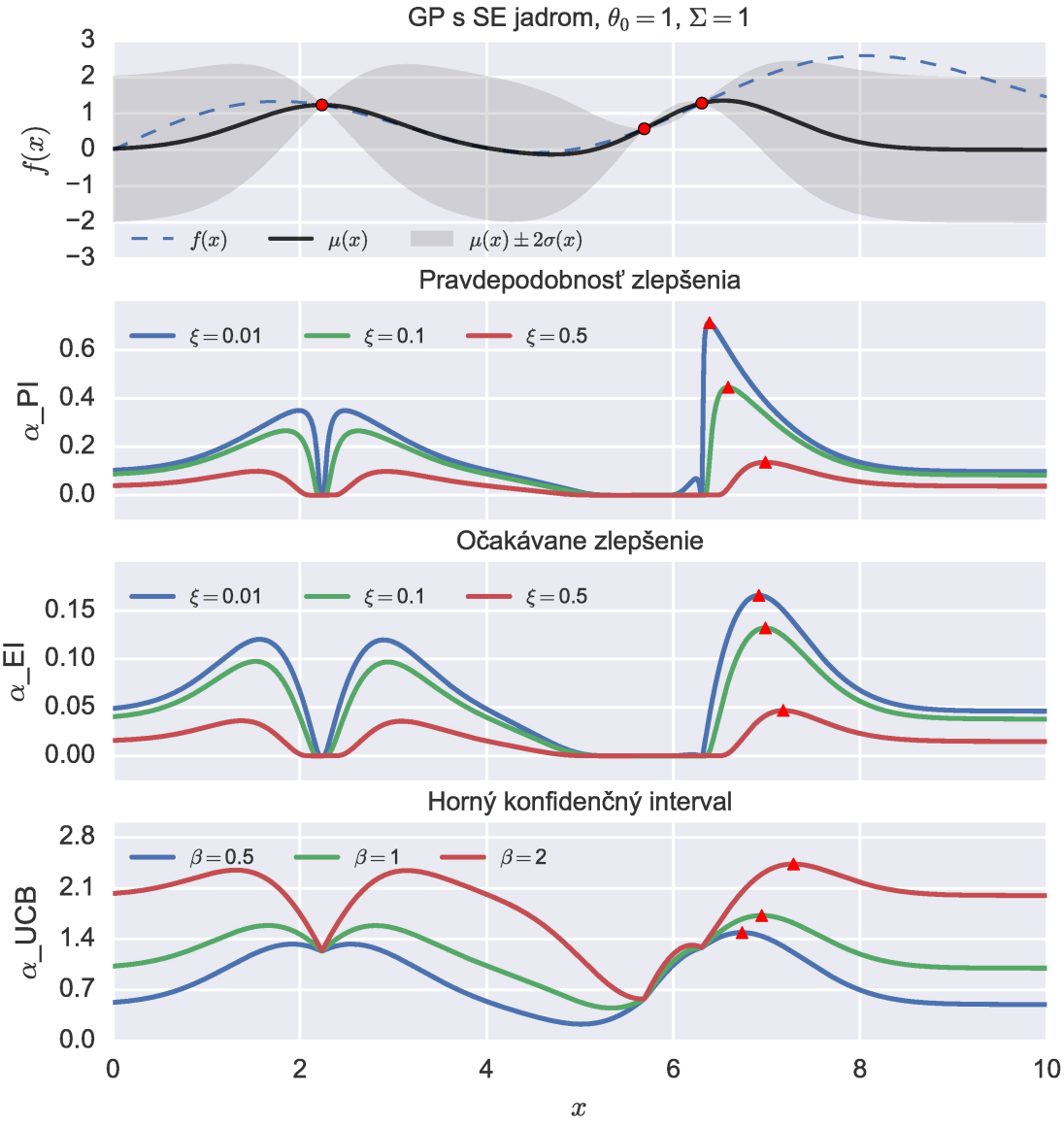
Dôkaz. Dané tvrdenie dokážeme priamo z definície strednej hodnoty pomocou vhodných substitúcií. Pre $s = y - y^+$ a $y \sim \mathcal{N}(\mu(\boldsymbol{x}),\sigma^2(\boldsymbol{x}))$ máme

$$\mathsf{E}_{\,s|\boldsymbol{x},\boldsymbol{\theta}} \max\{0,\, s\} = \int_0^\infty \frac{s}{\sqrt{2\pi}\sigma(\boldsymbol{x})} \exp\left(-\frac{(s - \mu(\boldsymbol{x}) + y^+)^2}{2\sigma^2(\boldsymbol{x})}\right)\mathrm{d}s.$$

Ďalej zoberme $t = (\mu(\boldsymbol{x}) - y^+ - s)/\sigma(\boldsymbol{x})$, teda $\frac{-1}{\sigma(\boldsymbol{x})}\mathrm{d}s = \mathrm{d}t$, $0 \to \frac{\mu(\boldsymbol{x})-y^+}{\sigma(\boldsymbol{x})}$ a $\infty \to -\infty$. Po výmene hraníc integrálu máme:

$$(\mu(\boldsymbol{x}) - y^+)\int_{-\infty}^{\frac{\mu(\boldsymbol{x})-y^+}{\sigma(\boldsymbol{x})}} \phi(t)\mathrm{d}t + \sigma(\boldsymbol{x})\int_{-\infty}^{\frac{\mu(\boldsymbol{x})-y^+}{\sigma(\boldsymbol{x})}} \phi'(t)\mathrm{d}t,$$

kde $\phi(t) = \frac{1}{\sqrt{2\pi}} \exp(-\frac{t^2}{2})$ a $\phi'(t) = \frac{-t}{\sqrt{2\pi}} \exp(-\frac{t^2}{2})$, čím sme dokázali (1.15). □



Obrázok 1.2: Porovnanie rôznych akvizičných funkcií pri rôznych hyper-parametroch. Zvolili sme jednoduchú účelovú funkciu $f(x) = \sin(x) + \frac{x}{5}$, $x \in (0{,}10)$. Pri apriórnom rozdelení sme zvolili nulovú strednú hodnotu a kovariančnú funkciu ako SE jadro s $\theta_0 = 1$ a $\Sigma = 1$. Vyznačené body (dáta) sme vygenerovali z $\mathcal{U}(0{,}10)$ a vypočítali ich funkčné hodnoty. $\mu(x)$ a $\sigma(x)$ sú definované v (1.13). Podobne ako v ľavom grafe z obrázka 1.1 máme $\sigma_y^2 = 0$, teda „predikcie“ v pôvodných dátach majú nulový rozptyl (viď poznámka pri (1.9)). Nachádzame sa v štvrtom kroku BO – maximum z akvizičnej funkcie nám určuje novú hodnotu x^*, pre ktorú vypočítame hodnotu $f(x^*)$. Inšpirácia k obrázku od Brochu a kol. (2010).

Pri aplikácií akvizičných funkcií (1.15) a (1.12) môže vzniknúť situácia keď body, ktoré majú veľkú pravdepodobnosť byť len nepatrne väčšie ako y^+ budú uprednostnené pred bodmi, ktoré ponúknu lepšie nové maximum ale s menšou

pravdepodobnosťou. Preto sa do odvodených akvizičných funkcií môže pridať hyper-parameter $\xi \geq 0$, ktorý nám zabezpečí vyššiu pravdepodobnosť, resp. zlepšenie, pre vzdialenejšie body s potencionálne vyššou hodnotou (viď obr. 1.2). Pre $\sigma(\boldsymbol{x}) > 0$ máme:

$$\alpha_{\mathrm{PI}}(\boldsymbol{x};\boldsymbol{\theta},\mathcal{D}_n) = \Phi\left(\frac{\mu(\boldsymbol{x}) - y^+ - \xi}{\sigma(\boldsymbol{x})}\right), \tag{1.16}$$

$$\alpha_{\mathrm{EI}}(\boldsymbol{x};\boldsymbol{\theta},\mathcal{D}_n) = (\mu(\boldsymbol{x}) - y^+ - \xi)\Phi\left(\frac{\mu(\boldsymbol{x}) - y^+ - \xi}{\sigma(\boldsymbol{x})}\right) + \sigma(\boldsymbol{x})\phi\left(\frac{\mu(\boldsymbol{x}) - y^+ - \xi}{\sigma(\boldsymbol{x})}\right). \tag{1.17}$$

Optimistický prístup

V optimistickom prístupe akvizičnú funkciu zostrojíme ako horný konfidenčný interval (UCB) pre každý bod $\boldsymbol{x} \in \mathcal{X}$. Týmto spôsobom efektívne využívame fixnú pravdepodobnosť najlepšieho scenára podľa daného modelu. Keďže v aposteriórnom rozdelení má každá hodnota $f(\boldsymbol{x})$ na základe gaussovského procesu podmienené rozdelenie (1.9), tak akvizičná funkcia bude tvaru:

$$\alpha_{\mathrm{UCB}}(\boldsymbol{x};\boldsymbol{\theta},\mathcal{D}_n) = \mu(\boldsymbol{x}) + \beta\sigma(\boldsymbol{x}), \tag{1.18}$$

kde $\mu(\boldsymbol{x})$ a $\sigma(\boldsymbol{x})$ sme využívali aj v predchádzajúcich akvizičných funkciách a $\beta > 0$ je nový hyper-parameter.

Ako vidíme na obrázku 1.2, čím sú hyper-parametre ξ a β väčšie, tým skôr sa dostaneme k vzdialenejším hodnotám od pôvodných dát. Pre $\beta = 2$ je akvizičná funkcia horným 95% predikčným intervalom. Keďže sme strednú hodnotu apriórneho rozdelenia zvolili nulovú tak vidíme, že aposteriórne rozdelenie vzdialenejších hodnôt je $\mathcal{N}(0,1)$. Vidíme, že akvizičné funkcie sú multimodálne a ich optimalizácia nebude jednoduchá. Sú však ľahko spočítateľné a pri použití diferencovateľnej jadrovej funkcie sú diferencovateľné.

Prístup založený na informácii

Shahriari a kol. (2016) uviedli aj akvizičné funkcie založené na teórii informácie. V tomto prístupe uvažujeme aposteriórne rozdelenie $\mathsf{P}_*(\boldsymbol{x}|\mathcal{D}_n)$ pre neznáme minimum $\boldsymbol{x}^*$, implicitne odvodené z aposteriórneho rozdelenia pre $-f$ (v tomto prístupe sa preferuje minimalizácia). Prvá akvizičná funkcia, ktorej sa nebudeme veľmi venovať, sa nazýva *Thompsonové vzorkovanie*. Metóda je založená na generovaní vzoriek z $\mathsf{P}_*(\boldsymbol{x}|\mathcal{D}_n)$ a výbere vzorky s najnižšou hodnotou. V n-tom kroku optimalizácie je úlohou minimalizovať spojitú funkciu $f^{(n)}$. Funkcia $f^{(n)}$ musí byť pevne zvolená aby sme ju vedeli vypočítať v ľubovoľných bodoch a nájsť minimum. Táto metóda však zlyháva pri vyšších dimenziách.

Namiesto vzorkovania z $\mathsf{P}_*(\boldsymbol{x}|\mathcal{D}_n)$, technika *prehľadávania entropie* [ES – entropy search] sa sústredí na redukovanie neistoty v neznámom bode $\boldsymbol{x}^*$ výberom bodu, v ktorom očakávame najväčší pokles entropie pre rozdelenie $\mathsf{P}_*(\boldsymbol{x}|\mathcal{D}_n)$ (Villemonteix a kol., 2009).

Definícia 1.3 (**Entropia**). *Nech Z je náhodná veličina s hustotou p (v diskrétnom prípade s pravdepodobnostnou funkciou P) definovanou na priestore $\mathcal{Z}$. Entropiu (Shanonnovu) definujeme:*

$$H(Z) = -\sum_{z\in\mathcal{Z}} \mathsf{P}(Z=z)\ln \mathsf{P}(Z=z), \tag{1.19}$$

resp. spojitú (diferenciálnu) entropiu:

$$h(Z) = -\int_{\mathcal{Z}} p(z)\ln p(z)\mathrm{d}z. \tag{1.20}$$

Podmienenú entropiu pre náhodné veličiny Z a W definujeme:

$$H(Z|W) = -\sum_{z\in\mathcal{Z},w\in\mathcal{W}} \mathsf{P}(Z=z,W=w)\ln\left(\frac{\mathsf{P}(W=w)}{\mathsf{P}(Z=z,W=w)}\right), \tag{1.21}$$

resp. podmienenú spojitú entropiu:

$$h(Z|W) = -\iint_{\mathcal{W}\times\mathcal{Z}} p(z,w)\ln p(z|w)\mathrm{d}z\mathrm{d}w = h(Z,W) - h(W), \tag{1.22}$$

kde $h(Z,W)$ je podobné (1.20) *– použijeme viacnásobný integrál a hustotu združeného rozdelenia. Často sa používa aj logaritmus so základom* 2.

S využitím značenia ako pri predchádzajúcich akvizičných funkciách máme:

$$\alpha_{\mathrm{ES}}(\boldsymbol{x};\boldsymbol{\theta},\mathcal{D}_n) = h(\boldsymbol{x}^*|\mathcal{D}_n) - \mathsf{E}_{\,y|\boldsymbol{x},\boldsymbol{\theta}}\left[h(\boldsymbol{x}^*|\mathcal{D}_n\cup\{\boldsymbol{x},y\})\right],$$

kde h je spojitá entropia na základe rozdelenia $\mathsf{P}_*(\boldsymbol{x}|\mathcal{D}_n)$, t.j. pre minimum $\boldsymbol{x}^*$, $y = f(\boldsymbol{x})$ ako v predchádzajúcich prípadoch a $y \sim \mathcal{N}(\mu(\boldsymbol{x}),\sigma^2(\boldsymbol{x}))$. Inými slovami ES meria očakávaný informačný prínos vzhľadom k ľubovoľnému bodu $\boldsymbol{x}\in\mathcal{X}$ a vyberá bod, ktorý nám dáva najlepšiu informáciu o neznámom minime $\boldsymbol{x}^*$. Shahriari a kol. (2016) upozorňujú, že daná akvizičná funkcia musí byť aproximovaná.

1.3.1 Optimalizácia akvizičnej funkcie

Definovali sme rôzne akvizičné funkcie, ktoré sú ľahko spočítateľné a vo väčšine prípadov aj diferencovateľné. Ďalej potrebujeme nájsť globálne maximum akvizičnej funkcie. Ako sme videli na obrázku 1.2, akvizičné funkcie sú multimodálne a pri vyššom počte dát aj zväčša ploché a teda globálna optimalizácia nie je jednoduchá.

Ak zvolíme diferencovateľné akvizičné funkcie, tak môžeme použiť metódy využívajúce gradienty, ktoré nájdu lokálne maximum. Pre nájdenie globálneho maxima danú metódu spustíme niekoľko-krát s vždy inou počiatočnou hodnotou.

Ďalšou možnosťou je vyhnúť sa lokálnym technikám pri optimalizácii multimodálnej funkcie a použiť techniky vetvenia priestoru založené na stromoch. Wang a kol. (2014) predstavili BaMSOO [Bayesian Multi-Scale Optimistic Optimization] – *bayesovská viac-škálová optimistická optimalizácia*. Metóda je odvodená z SOO (simultánna optimistická optimalizácia), ktorá hierarchicky rozdeľuje priestor $\mathcal{X}$ konštrukciou stromu. Táto metóda sa môže použiť priamo na optimalizáciu čiernej-skrinky f, avšak nevyužíva informácie z predchádzajúcich výpočtov f a je veľmi výpočtovo náročná.

BaMSOO je nadstavbou na SOO a využíva priamo horné a dolné konfidenčné intervaly aposteriórneho rozdelenia (podobne ako $\alpha_{UCB}(\boldsymbol{x};\boldsymbol{\theta},\mathcal{D}_n)$ z (1.18)) s meniacim sa hyper-parametrom β. Ak sa pri konštrukcii stromu dostaneme do oblasti, kde sú hodnoty akvizičnej funkcie menšie ako súčasné maximum, tak za hodnotu f určíme dolný konfidenčný interval. Tým zúžime priestor záujmu $\mathcal{X}$ a vyhneme sa výpočtu funkcie f v danom bode. Táto metóda už má niekoľko predpokladov. Najdôležitejšie sú: použitie dva-krát diferencovateľnej jadrovej funkcie pre GP, lokálna hladkosť danej funkcie a existencia globálneho maxima. Ďalšie predpoklady a vlastnosti uviedli Wang a kol. (2014) vo svojej práci.

Metóda BaMSOO by sa mohla použiť aj priamo pre funkciu f, ak by boli splnené potrebné predpoklady. Akvizičná funkcia spĺňa dané predpoklady, avšak na jej optimalizáciu nie je vhodné použiť BaMSOO a konštruovať ďalší GP, preto budeme používať prehľadávanie medzi lokálnymi maximami.

Preferované numerické metódy pre vlastný algoritmus BO

Existujú rôzne numerické algoritmy na hľadanie lokálneho extrému pomocou klesajúcich gradientov. Budeme používať metódu L-BFGS-B, ktorá je implementovaná v balíčku SciPy (Jones a kol., 2001–). Ide o kvázi-Newtonovú metódu, ktorej názov sú iniciály mien autorov, L označuje limitovanú pamäť (t.j. nevyužíva toľko pamäte ako pôvodná metóda BFGS) a B označuje uzatvorené intervaly [box]. V praxi sa väčšinou zaujímame o vstupy z viacrozmerných ohraničených intervalov. V prípade, že máme v úlohe (1.1) aj nejaké ďalšie obmedzenia, použijeme metódu SLSQP – sekvenčné programovanie najmenších štvorcov (ide o inú nadstavbu na BFGS).

Týmito numerickými metódami dokážeme nájsť lokálne extrémy v okolí počiatočného bodu. Výberom rôznych počiatočných bodov získame aj ostatné lokálne extrémy akvizičnej funkcie. Počiatočné hodnoty môžeme zvoliť tak aby každý bod sa nachádzal medzi dvoma už vypočítanými bodmi. Týmto spôsobom získame všetky lokálne maximá akvizičnej funkcie (prehľadávanie medzi všetkými už vypočítanými bodmi môže byť pri vyššom počte bodov zbytočne výpočtovo náročné).

V balíčku SciPy je implementovaná *preskakovacia metóda* [basin-hopping], ktorú opísali Wales a Doye (1997). Táto metóda je vhodná na hľadanie globálneho extrému pre viacrozmerné multimodálne funkcie. Metóda vhodným spôsobom určuje nové počiatočné body, z ktorých nájdeme lokálny extrém. Pre dosiahnutie lepších výsledkov je vhodné metódu spustiť viackrát v rôznych štartoch.

1.3.2 Odhad hyper-parametrov

Doteraz sme sa venovali akvizičným funkciám $\alpha(\boldsymbol{x};\boldsymbol{\theta},\mathcal{D}_n)$ s pevnými hyper-parametrami $\boldsymbol{\theta} \in \boldsymbol{\Theta}$, kde $\boldsymbol{\Theta}$ je vhodný parametrický priestor podľa zvoleného jadra. Častým prístupom k odhadu hyper-parametrov je bodový odhad $\boldsymbol{\theta}$ a teda odhad akvizičnej funkcie:

$$\hat{\alpha}(\boldsymbol{x};\mathcal{D}_n) = \alpha(\boldsymbol{x};\hat{\boldsymbol{\theta}},\mathcal{D}_n),$$

kde $\hat{\boldsymbol{\theta}}$ je najčastejšie maximálny vierohodný odhad $\hat{\boldsymbol{\theta}}_n^{\mathrm{MLE}}$ alebo maximálnym aposteriórnym odhadom $\hat{\boldsymbol{\theta}}_n^{\mathrm{MAP}}$. MLE získame z marginálnej vierohodnosti pre gaus-

sovský proces, t.j. pre $\boldsymbol{y}$ z (1.8):

$$\hat{\boldsymbol{\theta}}_n^{\mathrm{MLE}} = \arg\max_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} \ln p(\boldsymbol{y}|\boldsymbol{X},\boldsymbol{\theta}) = \arg\max_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} \ln \phi(\boldsymbol{y}|\mu_0(\boldsymbol{X}),\boldsymbol{K}_y(\boldsymbol{\theta})),$$

kde ϕ je hustota normálneho rozdelenia, $\boldsymbol{y}$ sú hodnoty účelovej funkcie f pre dostupné dáta $\boldsymbol{X} = (\boldsymbol{x}_1,\ldots,\boldsymbol{x}_n)^\top$, $\boldsymbol{K}_y(\boldsymbol{\theta})$ je definované v (1.7) a $\boldsymbol{\theta}$ sú hyper-parametre z jadrovej funkcie (Snoek a kol., 2012). MAP vypočítame pomocou bayesovej vety:

$$\hat{\boldsymbol{\theta}}_n^{\mathrm{MAP}} = \arg\max_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} p(\boldsymbol{\theta}|\mathcal{D}_n) = \arg\max_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} \frac{p(\boldsymbol{y}|\boldsymbol{X},\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\mathcal{D}_n)} = \arg\max_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} p(\boldsymbol{y}|\boldsymbol{X},\boldsymbol{\theta})p(\boldsymbol{\theta}),$$

kde $p(\boldsymbol{\theta})$ je zvolené apriórne rozdelenie pre $\boldsymbol{\theta}$ a $p(\mathcal{D}_n|\boldsymbol{\theta}) = p(\boldsymbol{y}|\boldsymbol{X},\boldsymbol{\theta})$.

Bodové odhady však nie sú vhodné a aby sme zabezpečili úplný bayesovský prístup k danému problému zvolíme *integrovanú akvizičnú funkciu*:

$$\alpha(\boldsymbol{x};\mathcal{D}_n) = \mathsf{E}_{\boldsymbol{\theta}|\mathcal{D}_n}[\alpha(\boldsymbol{x};\boldsymbol{\theta},\mathcal{D}_n)] = \int_{\boldsymbol{\Theta}} \alpha(\boldsymbol{x};\boldsymbol{\theta},\mathcal{D}_n)p(\boldsymbol{\theta}|\mathcal{D}_n)\mathrm{d}\boldsymbol{\theta}. \tag{1.23}$$

Integrál aproximujeme Monte Carlo metódou s použitím M vzoriek $\{\boldsymbol{\theta}_n^{(i)}\}_{i=1}^M$, t.j.:

$$\mathsf{E}_{\boldsymbol{\theta}|\mathcal{D}_n}[\alpha(\boldsymbol{x};\boldsymbol{\theta},\mathcal{D}_n)] \approx \frac{1}{M}\sum_{i=1}^{M} \alpha(\boldsymbol{x};\boldsymbol{\theta}_n^{(i)},\mathcal{D}_n). \tag{1.24}$$

V praxi však nedokážeme vzorkovať priamo z aposteriórneho rozdelenia, teda použijeme techniku Markovský reťazec Monte Carlo (MCMC), prípadne sekvenčné Monte Carlo (Shahriari a kol., 2016).

Wang a de Freitas (2014) riešili konvergenciu bayesovskej optimalizácie s neznámymi hyper-parametrami pre akvizičnú funkciu $\alpha_{\mathrm{EI}}(\boldsymbol{x};\boldsymbol{\theta},\mathcal{D}_n)$ a ukázali typy účelovej funkcie, pri ktorých BO nekonverguje k globálnemu optimu. Ďalej odvodili vhodný spôsob voľby hyper-parametrov $\boldsymbol{\theta}$ pre GP a taktiež spôsob voľby parametra ξ z (1.17) pre každý krok BO. Pre danú voľbu hyper-parametrov autori dokázali konvergenciu s pravdepodobnosťou $1-\delta$ pre $\delta \in (0,1)$. Podrobnejšie sa týmito odhadmi nebudeme zaoberať, pretože sú mimo hlavného záujmu tejto práce.

1.4 Algoritmus BO

Zostrojíme vlastný algoritmus pre bayesovskú optimalizáciu so špeciálnym určovaním hyper-parametrov $\boldsymbol{\theta}$ a ξ, resp. β pre α_{UCB}. Nech $\boldsymbol{\theta} = (\theta_1,\ldots,\theta_k,\boldsymbol{\ell})$, kde k je počet parametrov z jadrovej funkcie netýkajúcich sa mierky a $\boldsymbol{\ell}$ sú parametre mierky. Najčastejšie bude $\boldsymbol{\ell} = \ell$ jednorozmerné, t.j. rovnaká mierka pre všetky dimenzie. Ostatné označenia ostávajú rovnaké ako v predchádzajúcich častiach a predpokladáme, že $\mathcal{X} \subset \mathbb{R}^d$ je kompaktná spojitá.

V algoritme 2 zvolíme dva počiatočné body náhodným výberom z rovnomerného rozdelenia na kompaktnej spojitej množine $\mathcal{X}$ (prípadu nespojitej $\mathcal{X}$ sa budeme venovať v ďalšej časti). Minimálne dva body sú vhodným vstupom pre regresiu podľa GP. Pri aktualizácii hyper-parametrov sme sa inšpirovali prácou od Wang a de Freitas (2014).

Algoritmus 2 vlastný algoritmus BO (viď príloha 1 – kód v Python™)

input pevne zvoľ σ_y^2 (malé), $t_\sigma \leq \sigma_y^2$ a tie parametre z $\boldsymbol{\theta}$, ktoré sa netýkajú mierky $\boldsymbol{\ell}$
input vhodne zvoľ interval pre parameter $\boldsymbol{\ell}$: $\boldsymbol{\ell}^L < \boldsymbol{\ell}^U$, $\boldsymbol{\ell}^L \approx \mathbf{0}$
(náhodne) zvoľ prvé 2 body $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathcal{X}$
vypočítaj $y_1 = f(\boldsymbol{x}_1)$, $y_2 = f(\boldsymbol{x}_2)$ a definuj $\mathcal{D}_2 = \{(\boldsymbol{x}_1,y_1),(\boldsymbol{x}_2,y_2)\}$
zvoľ $\xi_2 = \frac{1}{2}|y_1 - y_2|$, prípadne $\beta_2 = 2$
generuj $\boldsymbol{\ell}_2$ z viacrozmerného $\mathcal{U}(\boldsymbol{\ell}^L,\boldsymbol{\ell}^U)$
inicializuj $a = 2$, $C = 0$
for $n = 2,3,\ldots,N$ **do**
    zostroj regresiu podľa GP použitím $\mathcal{D}_n$ a $\boldsymbol{\theta}_n = (\theta_1,\ldots,\theta_k,\boldsymbol{\ell}_n)$
    $\boldsymbol{x}_{n+1} = \arg\max_{\boldsymbol{x}\in\mathcal{X}} \alpha(\boldsymbol{x};\boldsymbol{\theta}_n,\xi_n,\mathcal{D}_n)$ [L-BFGS-B s reštartmi v náhodných bodoch alebo *basin-hopping* s L-BFGS-B s viac reštartmi]
    if $\sigma(\boldsymbol{x}_{n+1}) < t_\sigma$ **then**
        $C \leftarrow C + 1$
    else
        $C \leftarrow 0$
    vypočítaj hodnotu $y_{n+1} = f(\boldsymbol{x}_{n+1})$
    rozšír dáta $\mathcal{D}_{n+1} = \mathcal{D}_n \cup (\boldsymbol{x}_{n+1},y_{n+1})$
    if C=5 **then**
        $\boldsymbol{\ell}^U \leftarrow \max(\boldsymbol{\ell}^L, 0.8\boldsymbol{\ell}^U)$ po zložkách
        $C \leftarrow 0$
    generuj $\boldsymbol{\ell}_{n+1} \sim \mathcal{U}(\boldsymbol{\ell}^L,\boldsymbol{\ell}^U)$
    if $n \bmod 10 = 0$ **then**
        $a \leftarrow a + 1$
    generuj $\xi_{n+1} \sim \mathcal{U}\left(0,\frac{1}{a}\left[\max_{i\in 1:(n+1)} y_i - \min_{i\in 1:(n+1)} y_i\right]\right)$, resp. $\beta_{n+1} \sim \mathcal{U}\left(0,\frac{6}{a}\right)$
return $(\boldsymbol{x}^*,\ y^*)$: $y^* = \max_{i\in 1:N} y_i$, $\boldsymbol{x}^* = \arg\max_{i\in i:N} y_i$

Parametre mierky jadrovej funkcie nebudeme aktualizovať pomocou MCMC metódy ale priamo z apriórneho rozdelenia. Za apriórne rozdelenie zvolíme viacrozmerné rovnomerné rozdelenie $\mathcal{U}(\boldsymbol{\ell}^L,\boldsymbol{\ell}^U)$, ktoré bude vo väčšine prípadov jednorozmerné. Na začiatku volíme širší interval (ℓ^L,ℓ^U), napr. (0,100). Interval zúžime ak 5 iterácii za sebou nastane situácia, že nový bod $\boldsymbol{x}_{n+1}$ má v podmienenom GP n hodnotami menší rozptyl ako nejaká zvolená hranica $t_\sigma \leq \sigma_y^2$. Táto situácia môže nastať, ak máme veľa bodov blízko seba a teda máme dostatok informácií na vhodnú zmenu apriórneho rozdelenia. Dolné ohraničenie je blízke $\mathbf{0}$ a meniť ho nepotrebujeme.

Pre konvergenciu algoritmu je najdôležitejšia správna voľba parametru ξ. Ako sme videli na obrázku 1.2, čím je ξ väčšie, tým „rýchlejšie sa vieme pohybovať“ po celom priestore $\mathcal{X}$. Pri spustení algoritmu chceme v krátkom čase prejsť čo najširšiu oblasť aby sme neostali v okolí lokálneho maxima. Po dostatočnom počte iterácií nás už nezaujíma rýchle prehľadanie celého priestoru ale čo najpresnejšie globálne maximum. Túto podmienku vieme splniť ak budeme opäť generovať z rovnomerného apriórneho rozdelenia s tým, že hornú hranicu budeme po každých 10-tich iteráciách znižovať. ξ však nemôže byť príliš veľké, pretože potom sú

pravdepodobnosti veľmi malé a akvizičná funkcia je väčšinou plochá s nulovými gradientami.

Optimalizácia akvizičnej funkcie je základom pre funkčnosť BO. V algoritme sme implementovali 2 metódy. Optimalizácia cez L-BFGS-B so štartmi v náhodne vygenerovaných bodoch z $\mathcal{X}$, použijeme 15 reštartov (možné zmeniť). Alebo optimalizácia preskakovacou metódou s využitím L-BFGS-B – táto metóda je už výpočtovo náročnejšia a tiež je potrebné použiť viac reštartov. Za predvolenú metódu určíme minimalizáciu cez L-BFGS-B.

Aktualizáciu daných hyper-parametrov sme nevoli tak sofistikovane ako Wang a de Freitas (2014). Ide však o ukážku toho, že algoritmus pre BO sa dá implementovať rôznymi spôsobmi. V balíčku RoBo je implementovaných niekoľko typov BO s rôznymi apriórnymi rozdeleniami pre f, akvizičnými funkciami, metódami maximalizácie akvizičnej funkcie a voľbami hyper-parametrov (napr. cez MCMC).

1.4.1 Ukážkové príklady

Algoritmus 2 najskôr implementujeme na jednorozmernom ukážkovom príklade a pre niekoľko prvých krokov zobrazíme regresiu podľa GP a akvizičné funkcie. Ďalej algoritmus otestujeme na jednoduchej 3 rozmernej funkcii a aspoň graficky ukážeme konvergenciu.

Príklad 1

Prvým príkladom bude maximalizácia funkcie $f\colon [0{,}10] \to \mathbb{R}$, ktorá má 2 body nespojitosti. Funkciu sme zvolili tak aby bolo jasné, že BO môžeme použiť aj pre funkcie s konečným počtom bodov nespojitosti.

$$f(x) = \begin{cases} 4\sin\left(x + \frac{\pi}{3}\right) - x + 4, & 0 \leq x < 4, \\ 4\sin\left(x + \frac{\pi}{3}\right) - x + 7, & 4 \leq x < 7, \\ 10 - \exp(x - 7.5) + 3, & 7 \leq x \leq 10. \end{cases}$$

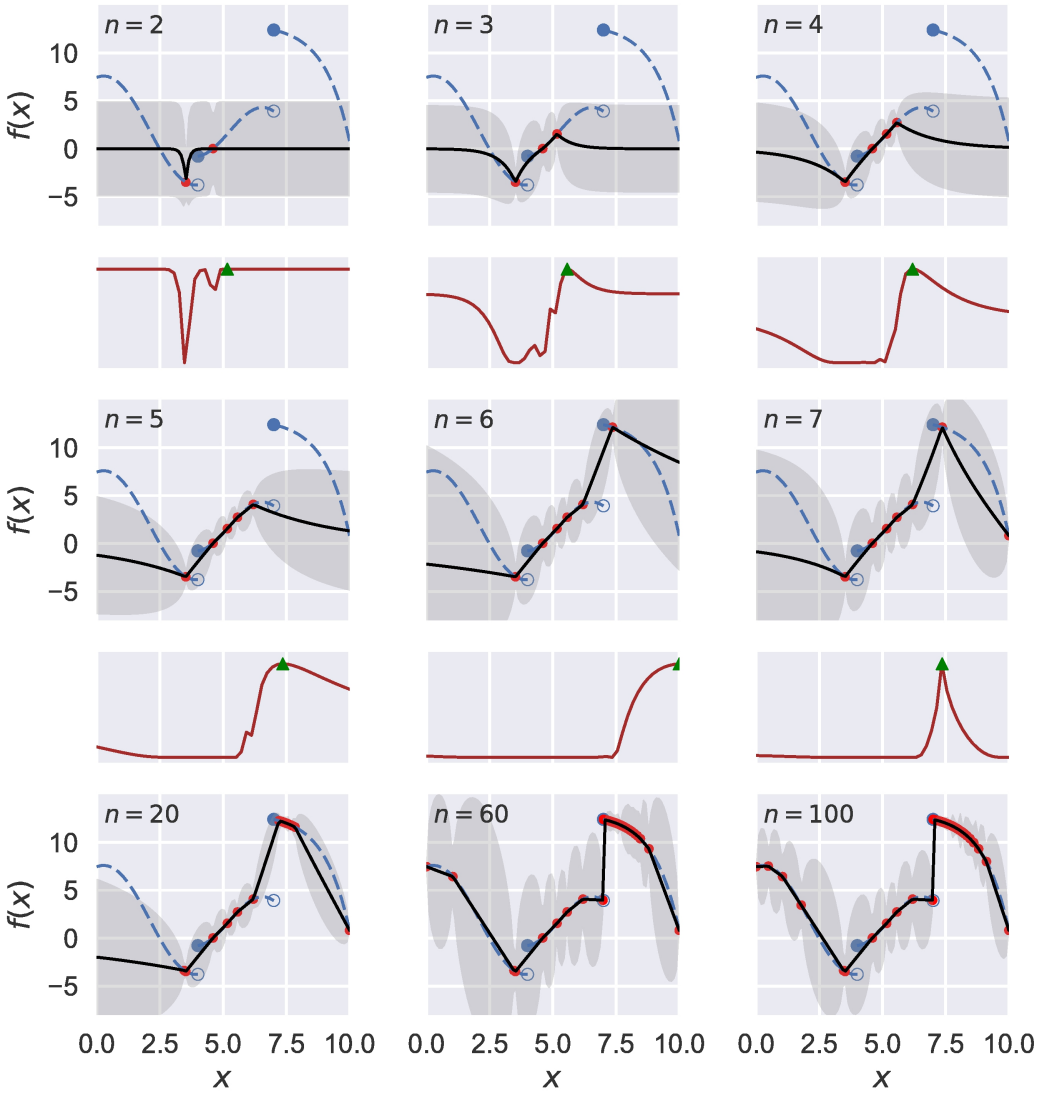
Postupné riešenie daného príkladu máme na obrázku 1.3. Použili sme algoritmus 2 a pre prvých 7 krokov sme vykreslili podmienené GP (regresiu) a akvizičné funkcie. Ďalej sme vykreslili ešte podmienené GP v krokoch 20, 60 a 100. Použili sme SE jadro s hyper-parametrom $\theta_0 = 1$ a pre mierku ℓ sme použili interval [0,100]. Rozptyl šumu sme určili $\sigma_y^2 = 0.001$.

Z definície funkcie f ľahko určiť, že maximum sa nachádza v bode $x = 7$, čo je aj viditeľné na obrázku. Maximum funkcie f sa nachádza v bode, ktorý nie je zľava spojitý. Po $n = 100$ krokoch BO sme dostali výsledok $x = 7.0027$.

Na obrázku 1.3 vidíme postupne ako regresia podľa GP reaguje na nové dáta. Pri GP volíme apriórnu strednú hodnotu nulovú, čo je pri menšom počte dát viditeľné – ďalej od dát sa stredná hodnota vracia k apriórne zvolenej nulovej úrovni. Pri vyššom počte dát vidíme, že apriórna nulová stredná hodnota už nemá veľký vplyv na aposteriórny GP.

Pre kroky $n = 2$ až $n = 7$ máme zakreslené hodnoty akvizičných funkcií aj s maximami – vidíme, že bod predchádzajúceho maxima akvizičnej funkcie sa už vyskytuje v dátach. V kroku $n = 2$ nemáme jednoznačné maximum, tak bola zvolená náhodná hodnota (jeden z náhodne zvolených štartovacích bodov pre L-BFGS-B optimalizáciu akvizičnej funkcie).

Po vyššom počte krokov ($n = 60$ a $n = 100$) sme získali presnejšiu hodnotu globálneho maxima a z obrázka vidíme aj dobrý odhad celej funkcie f.



Obrázok 1.3: Grafy BO pre jednorozmerný ukážkový príklad pre rôzne kroky. Modrá prerušovaná čiara sú funkčné hodnoty funkcie f a červené body reprezentujú dáta $\mathcal{D}_n$. Ďalej je zobrazená stredná hodnota a 95% predikčný pás z podmieneného GP (z regresie podľa GP) a pre prvých 7 krokov máme aj akvizičné funkcie v osobitných grafoch s vyznačeným maximom. Podrobný opis obrázku je v texte práce.

Príklad 2

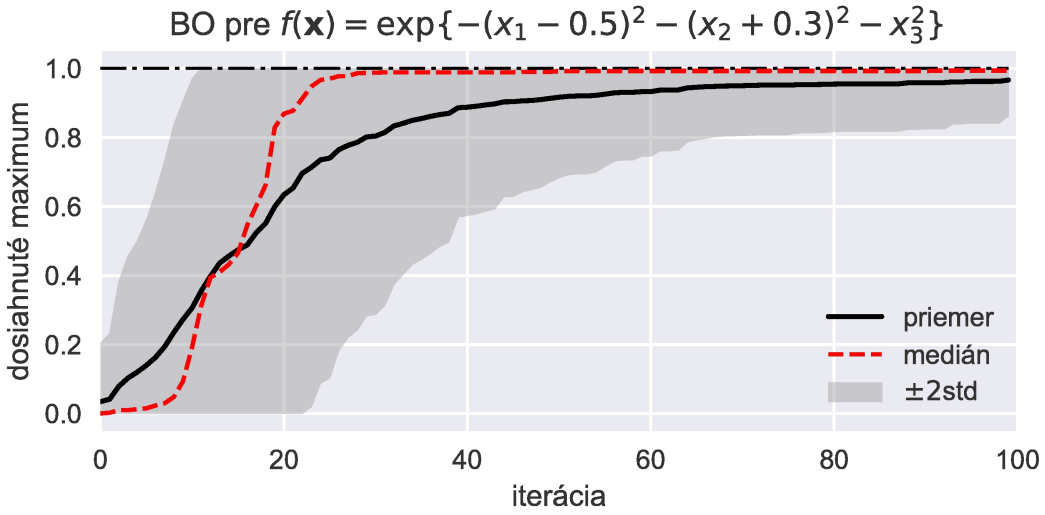
Na test funkčnosti algoritmu 2 pre vyššie dimenzie a celkovo ukážku konvergencie sme použili jednoduchú funkciu $f\colon [-3{,}3]^3 \to \mathbb{R}$,

$$f(\boldsymbol{x}) = \exp\left(-(x_1 - 0.5)^2 - (x_2 + 0.3)^2 - x_3^2\right),$$

ktorej maximu dokážeme nájsť priamo z pohľadu na definíciu funkcie. Teda $\boldsymbol{x}_{\text{opti}} = (0.5, -0.3{,}0)^\top$ a $f(\boldsymbol{x}_{\text{opti}}) = 1$. Pevné hyper-parametre sme zvolili rov-

nako ako v príklade 1 a opäť sme použili SE jadro. Na ukážku konvergencie sme urobili 50 nezávislých BO s $n = 100$ krokmi a postupne určovali nájdené maximum pre každý krok. Nakoniec sme priemerné maximum, mediánové maximum a 95% interval zakreslili do obrázku 1.4.

Keďže odľahlé pozorovania dokážu silno ovplyvniť priemer, tak sme použili aj medián. Medián bol blízky reálnemu maximu už od 25. kroku. Priemer je ovplyvnený odľahlými pozorovaniami – z 50 aplikácii sme v 3 prípadoch po 100 krokoch dosiahli hodnotu okolo 0.8, vo všetkých ostatných prípadoch sme dosiahli hodnotu vyššiu 0.9 a väčšinou vyššiu ako 0.99, čo je viditeľné na mediáne.



Obrázok 1.4: Ukážka konvergencie algoritmu BO pre jednoduchú 3-rozmernú funkciu. Hodnoty sú získané z 50 nezávislých aplikácii BO.

1.5 Rozšírenia

1.5.1 Paralelná optimalizácia viacerých úloh

V praxi sa môžme stretnúť aj s *paralelnou optimalizáciou viacerých úloh* [multi-task optimization], t.j. pre funkciu $f\colon \mathcal{X} \to \mathbb{R}^T$. Pre $\boldsymbol{x} \in \mathcal{X}$, každý prvok vektoru $f(\boldsymbol{x})$ reprezentuje úlohu t. Funkcia f by sa dala opísať aj ako $f(\boldsymbol{x}) = (f_1(\boldsymbol{x}), \ldots, f_T(\boldsymbol{x}))^\top$, kde f_i sú nejakým spôsobom navzájom korelované, nejde však o rôzne funkcie ako vo viac-kritériálnej optimalizácii. S daným problémom sa môžeme stretnúť ak sa snažíme optimalizovať parametre nejakého modelu na viac úlohách naraz. Každá úloha má rôzne dáta na ktoré aplikujeme ten istý model a naším cieľom je nájsť optimálne parametre naprieč všetkými úlohami (napr. minimalizácia chyby pri modeloch strojového učenia).

Ako apriórne rozdelenie použijeme GP pre paralelné úlohy. Jeho základom je určenie vhodnej kovariančnej funkcie $\kappa_{\mathsf{multi}}((\boldsymbol{x},t),(\boldsymbol{x}',t'))$ medzi pármi bodov a úloh. Swersky a kol. (2013) navrhujú riešenie s názvom *vnútorný model koregionalizácie* [intrinsic model of coregionalization]:

$$\kappa_{\mathsf{multi}}((\boldsymbol{x},t),(\boldsymbol{x}',t')) = \kappa_{\mathsf{t}}(t,t') \otimes \kappa_{\mathsf{x}}(\boldsymbol{x},\boldsymbol{x}'),$$

kde $\otimes$ je Kroneckerov súčin, κ_{x} meria vzťah medzi bodmi a κ_{t} vzťah medzi úlohami. Pre dané kovariančné funkcie už môžeme použiť pozitívne definítne jadro. S kovariančnou funkciou κ_{multi} už máme klasický GP.

Ak v odvodenej aposteriórnej strednej hodnote a v rozptyle z (1.9) použijeme κ_{multi}, tak získame $\mu(\boldsymbol{x},t)$ a $\sigma^2(\boldsymbol{x},t,t')$ (podobne ako v (1.13)). $\mu(\boldsymbol{x},t)$ je aposteriórna stredná hodnota v bode $\boldsymbol{x}$ pre úlohu t a $\sigma^2(\boldsymbol{x},t,t')$ je aposteriórna kovariancia medzi bodmi z úloh t a t', t.j. medzi $(\boldsymbol{x},t)$ a $(\boldsymbol{x},t')$. Najjednoduchšou možnosťou je v každej iterácií vypočítať hodnoty pre všetky úlohy a optimalizovať akvizičnú funkciu z priemerného GP, t.j.

$$\bar{\mu}(\boldsymbol{x}) = \frac{1}{T}\sum_{t=1}^{T}\mu(\boldsymbol{x};t), \quad \bar{\sigma}^2(\boldsymbol{x}) = \frac{1}{T^2}\sum_{t=1}^{T}\sum_{t'=1}^{T}\sigma^2(\boldsymbol{x},t,t').$$

Tento spôsob však v každej iterácií potrebuje vypočítať hodnotu funkcie pre všetky úlohy, čo môže byť veľmi výpočtovo náročné.

Swersky a kol. (2013) predstavili spôsob na základe prehľadávania entropie a vytvorili novú akvizičnú funkciu. $\alpha_{ES}(\boldsymbol{x};\boldsymbol{\theta},\mathcal{D}_n)$ je definovaná pre jednu účelovú funkciu, teda v tomto prípade pre jednu úlohu. Ak zavedieme funkciu $c_t\colon \mathcal{X} \to \mathbb{R}^+$ ako náklady na výpočet f pre úlohu t, tak môžme definovať *informačný prírastok na jednotku nákladu* [information gain per unit cost]:

$$\alpha_{\mathrm{IG}}(\boldsymbol{x}^t;\boldsymbol{\theta},\mathcal{D}_n) = \mathsf{E}_{\,y|\boldsymbol{x},\boldsymbol{\theta}}\,\frac{h(\boldsymbol{x}^*|\mathcal{D}_n) - [h(\boldsymbol{x}^*|\mathcal{D}_n \cup \{\boldsymbol{x},y\})]}{c_t(\boldsymbol{x})},$$

kde $\boldsymbol{x}^t$ označuje, že sa zaoberáme bodom $\boldsymbol{x}$ a úlohou t. Daný vzťah platí ak je $\sigma_y^2 = 0$, inak platí $y \sim \mathcal{N}(f(\boldsymbol{x}),\sigma_y^2)$ a potrebujeme aplikovať ďalšiu strednú hodnotu (podobne aj pri prehľadávaní entropie). Funkcia c_t nám však nie je známa a preto Swersky a kol. (2013) ju odporúčajú odhadnúť podobne ako funkciu f, teda viac-úlohovým GP pre $\ln c_t$.

1.5.2 Kategoriálne premenné funkcie f

Doteraz sme sa venovali len spojitým premenným funkcie f. V praxi sa však môžme stretnúť s funkciami, ktorých vstupy môžu byť aj kategoriálne (diskrétne alebo kvalitatívne). Konkrétne sme predpokladali, že prvky vektora $\boldsymbol{x}$ sú z uzavretých intervalov. Pri spojitých premenných nenastávajú žiadne vážnejšie problémy. Napríklad ak by nejaký vstup x_i, $i \in 1{:}d$ bol zo zjednotenia disjunktných intervalov, tak by sme zobrali uzavretý interval, ktorý prekrýva všetky menšie intervaly a pridali by sme obmedzenia.

Problém nastáva ak máme kategoriálne vstupy. Jednou z možností je vytvoriť jadrovú funkciu pre kategoriálne vstupy. Vezmime v úvahu priestor $\mathcal{X} = \mathcal{X}_{\mathrm{spoj}} \times \mathcal{X}_{\mathrm{kat}}$ a definujme D_{spoj} ako množinu indexov pre spojité vstupy a D_{kat} pre kategoriálne. Použijeme modifikované SE jadro (resp. modifikované RBF jadro) pre $\boldsymbol{x},\boldsymbol{x}' \in \mathcal{X}$ s hyper-parametrom mierky $\lambda_i > 0, i \in 1{:}d$:

$$k_{\mathrm{spoj}}(\boldsymbol{x},\boldsymbol{x}') = \exp\left[-\sum_{i\in D_{\mathrm{spoj}}}\lambda_i(x_i - x_i')^2\right],$$

$$k_{\mathrm{kat}}(\boldsymbol{x},\boldsymbol{x}') = \exp\left[-\sum_{i\in D_{\mathrm{kat}}}\lambda_i \mathbb{1}_{[x_i\neq x_i']}\right],$$

Hutter a kol. (2011) dokázali, že pre GP môžeme použiť nasledujúce zmiešané jadro:

$$k_{\text{mix}}(\boldsymbol{x},\boldsymbol{x}') = \exp\left[\sum_{i\in D_{\text{spoj}}} -\lambda_i(x_i - x_i')^2 + \sum_{j\in D_{\text{kat}}} -\lambda_j \mathbb{1}_{[x_j\neq x_j']}\right].$$

S daným jadrom už vieme aplikovať regresiu podľa GP a vytvoriť akvizičnú funkciu. Daná akvizičná funkcia však bude mať aj spojité aj kategoriálne vstupy. Maximalizácia akvizičnej funkcie je tentokrát problematická – ide o odlišnú multimodálnu funkciu pre každý kategoriálny parameter. Keďže výpočet akvizičnej funkcie nie je výpočtovo náročný, Hutter a kol. (2011) doporučujú vybrať 10 bodov s najvyššími hodnotami akvizičnej funkcie z $\boldsymbol{x}_{1:n}$ (autori používajú modifikované očakávané zlepšenie)[2]. Pri hľadaní lokálneho maxima sa pre kategoriálne vstupy použije náhodné zvolený najbližší „sused“, t.j. všetky konfigurácie sa od počiatočnej hodnoty líšia len o 1 diskrétny parameter a pre numerické vstupy sa náhodne určia 4 „susedia“. Ak znormalizujeme rozsah spojitých vstupov tak, aby sme mali intervaly [0,1], potom náhodných susedov k bodu x_i vyberieme z $\mathcal{N}(x_i,0.04)$ tak, aby patrili do intervalu [0,1]. Ide o modifikované náhodné prehľadávanie. Ďalej autori odporúčajú ešte náhodne zvoliť 10 000 bodov a v nich vypočítať hodnotu akvizičnej funkcie. Maximum z daných hodnôt by však malo byť v jednom z nájdených lokálnych maxím.

Aj keď je výpočet akvizičnej funkcie nenáročný, výpočet takého počtu bodov už nemusí byť veľmi efektívny. Ďalšiu metódu, ktorá dokáže pracovať aj s kategoriálnymi vstupmi, predstavíme v nasledujúcej časti. V danej metóde sa za apriórne rozdelenie nepoužíva GP ale *náhodný les* [random forest] a teda regresia cez náhodné lesy.

1.5.3 Iné prístupy k apriórnemu rozdeleniu pre f

Doteraz sme sa venovali gaussovským procesom ako apriórnemu rozdeleniu pre funkciu f a regresii podľa GP na získanie aposteriórneho rozdelenia. Existuje viac prístupov k tejto problematika, často využívané sú rôzne modifikácie GP. V tejto časti predstavíme ďalšie dva prístupy, ktoré nie sú založené na GP.

Náhodné lesy

Jednou z možností, ktorú sme už naznačili v predchádzajúcej časti, sú náhodné lesy. Náhodné lesy boli prvý krát predstavené L. Breimanom v roku 2001. Ide o metódu založenú na rozhodovacích stromoch – klasifikačné a regresné stromy (CART). CART postupne rozdeľuje, „vetví“ dizajnový priestor do pod-priestorov na základe pozorovaných dát. Regresný strom vetví spojitý priestor a klasifikácia sa používa pri kategoriálnych dátach. Náhodný les vzniká pomocou viac rozhodovacích stromov. V krátkosti, náhodný les tvojí skupina rozhodovacích stromov, kde každý strom je aplikovaný na náhodný výber z množiny pozorovaných dát. Následným spriemerovaním predikcií cez jednotlivé stromy získame presnejší výstup (Murphy, 2012).

[2]Autori hľadajú minimum a ich úžitkovou funkciou je $I_{\text{exp}}(\boldsymbol{x},y,\boldsymbol{\theta}) = \max(0, y^- - e^y)$, kde $y = f(\boldsymbol{x})$ a y^- je zatiaľ vypočítané minimum. Nech $v = \frac{\ln(y^-)-\mu(\boldsymbol{x})}{\sigma(\boldsymbol{x})}$ a $\mathsf{E}_{y|\boldsymbol{x},\boldsymbol{\theta}}[I_{\text{exp}}(\boldsymbol{x},y,\boldsymbol{\theta})] = y^-\Phi(v) - e^{\frac{1}{2}\sigma^2(\boldsymbol{x})+\mu(\boldsymbol{x})}\Phi(v-\sigma(\boldsymbol{x}))$.

Náhodné lesy sú často využívanou metódou s dobrými výsledkami. Ich hlavnou výhodou je, že aj pri použití väčšieho počtu stromov nedosiahneme „preučenie“ [overfitting]. Keďže téma rozhodovacích stromov a náhodných lesov je veľmi rozsiahla nebudeme sa jej ďalej venovať.

Hutter a kol. (2011) predstavili algoritmus SMAC [sequntial model-based algorithm configuration], ktorý využíva náhodné lesy na odvodenie aposteriórneho rozdelenia funkcie f. Hlavnou výhodou je, že náhodné lesy dokážu spoľahlivo pracovať s kategoriálnymi vstupmi alebo s podmienenými premennými. Algoritmus v každom kroku najskôr zostrojí náhodný les – m stromov na m náhodných podmnožinách rovnakej veľkosti $n_* < n$ z $\mathcal{D}_n$. Podobne ako pri regresii podľa GP, aby sme zostrojili akvizičnú funkciu potrebujeme určiť aposteriórny priemer $\mu(\boldsymbol{x})$ a rozptyl $\sigma^2(\boldsymbol{x})$ pre každý bod $\boldsymbol{x} \in \mathcal{X}$ (v tomto prípade ide o predikciu a predikčný interval v bode $\boldsymbol{x}$). Potrebné veličiny získame aritmetickým priemerom predikcie a predikčného intervalu v bode $\boldsymbol{x}$ cez všetky stromy. Ako akvizičnú funkciu využijeme modifikované EI, predstavené v predchádzajúcej časti. Hutter a kol. (2011) odporúčajú akvizičnú funkciu maximalizovať náhodným prehľadávaním, prípadne spôsobom, ktorý sme predstavili pri kategoriálnych premenných.

Ako sme spomínali, SMAC si dokáže poradiť zo všetkými typmi vstupov a bezproblémovo zvládne aj nespojitú účelovú funkciu f. Na aplikáciu náhodných lesov však potrebujeme väčšie množstvo dát, čo znamená viac výpočtov funkcie f. Zostrojenie m stromov a veľké množstvo predikcií pre každý strom v každom kroku algoritmu môže byť časovo náročné.

Neurónové siete

Springenberg a kol. (2016) predstavili bayesovskú optimalizáciu s umelými neurónovými sieťami a Hamiltonovským Monte Carlo (BOHAMIANN). Táto metóda je implementovaná v balíčku RoBO a autori tejto metódy sú zároveň spolu-tvorcami tohto balíčka, skupina ML4AAD (do danej skupiny patria aj Hutter a kol. (2011) – autori SMAC). Namiesto už uvedených regresných prístupov pre odhad funkcie f sa použije umelá neurónová sieť.

S využitím už zaužívaného označenia v tejto práci definujme pravdepodobnostný model:

$$p(f_t(\boldsymbol{x})|\boldsymbol{x},\boldsymbol{\theta}) = \phi(\hat{f}(\boldsymbol{x},t;\theta_\mu),\theta_{\sigma^2}),$$

kde t označuje t-tu úlohu (viď 1.5.1), $\boldsymbol{\theta} = (\theta_\mu,\theta_{\sigma^2})^\top$ sú hyper-parametre modelu, $\hat{f}(\boldsymbol{x},t;\theta_\mu)$ je nejaký parametrický model pre funkciu f s parametrom θ_μ, θ_{σ^2} je parameter rozptylu (v prípade heteroskedasticity môže byť funkciou $\boldsymbol{x}$) a ϕ je hustota normálneho rozdelenia. V tomto prípade $\hat{f}$ reprezentuje výstup z neurónovej siete. Predpokladajme T úloh, prípadne $T = 1$ pre jednoduchú optimalizáciu. Ak označíme vstupy a funkčné hodnoty ako $\mathcal{D}$ a $\mathcal{D}_{(t)}$ ako dáta len z t-tej úlohy, tak združená hustota pravdepodobnosti je

$$p(\mathcal{D},\boldsymbol{\theta}) = p(\theta_\mu)p(\theta_{\sigma^2}) \prod_{t=1}^{T} \prod_{i=1}^{|\mathcal{D}_{(t)}|} \phi(y_i^t|\hat{f}(\boldsymbol{x}_i^t,t;\theta_\mu),\, \theta_{\sigma^2}),$$

kde $p(\theta_\mu)$ a $p(\theta_{\sigma^2})$ sú apriórne rozdelenia hyper-parametrov a $y_i^t = f_t(\boldsymbol{x}_i)$.

K výpočtu akvizičnej funkcie potrebujeme najskôr určiť aposteriórne rozdelenie $p(f_t(\boldsymbol{x})|\boldsymbol{x},\mathcal{D})$. Keďže model pre f_t je neurónová sieť, tak analytický výpočet je

nemožný. Najskôr vzorkujeme hodnoty hyper-parametrov $\boldsymbol{\theta}^{(i)} \sim p(\boldsymbol{\theta}|\mathcal{D})$, $i \in 1{:}M$ pomocou Monte Carlo. Springenberg a kol. (2016) ukázali ako vzorkovať pomocou metódy Hamiltonovský Monte Carlo stochastický gradient. Použitím týchto vzoriek môžme aproximovať

$$p(f_t(\boldsymbol{x})|\boldsymbol{x},\mathcal{D}) = \int_{\boldsymbol{\Theta}} p(f_t(\boldsymbol{x})|\boldsymbol{x},\boldsymbol{\theta}) p(\boldsymbol{\theta}|\mathcal{D}) \mathrm{d}\boldsymbol{\theta} \approx \frac{1}{M} \sum_{i=1}^{M} p(f_t(\boldsymbol{x})|\boldsymbol{x},\boldsymbol{\theta}^{(i)}).$$

Nezabudnime, že predpokladáme normálne rozdelenie pre $f_t(\boldsymbol{x})$, teda strednú hodnotu a rozptyl môžeme aproximovať

$$\mu(\boldsymbol{x};t) = \frac{1}{M} \sum_{i=1}^{M} \hat{f}(\boldsymbol{x},t;\theta_{\mu}^{(i)}), \quad \sigma^2(\boldsymbol{x};t) = \frac{1}{M} \sum_{i=1}^{M} \left[\left(\hat{f}(\boldsymbol{x},t;\theta_{\mu}^{(i)}) - \mu(\boldsymbol{x};t) \right)^2 + \theta_{\sigma^2}^{(i)} \right].$$

Ďalej sa môže použiť diferencovateľná α_{EI}. Ak používame viac procesorov, tak úlohu môžme rozšíriť tak aby jeden procesor počítal hodnotu funkcie v jednom bode a ďalší v inom. Tejto téme a zostrojeniu akvizičnej funkcie, ktorá „čaká na nový bod“ a zároveň určuje ďalší, sa venovali Snoek a kol. (2012).

BOHAMIANN prirodzene podporuje viac-úlohovú optimalizáciu, ako aj paralelné výpočty funkcie a zvláda aj optimalizáciu vo vyšších dimenziách. Mierne odlišný prístup, tiež na základe neurónových sietí, používa algoritmus DNGO – hlboké siete pre globálnu optimalizáciu.

2. Nadmerná dimenzionalita dát a redukcia dimenzie

S problémom nadmernej dimenzie dát sa v dnešnej dobe stretávame pomerne často. Či už ide o rozsiahle databázy, kde počet parametrov značne prevyšuje počet pozorovaní alebo o funkcie, ktoré majú priveľa vstupov, tento problém sa stal súčasťou každodennej reality. Richard E. Bellman výstižne pomenoval tento problém ako **Curse of dimensionality**, teda „prekliatie rozmernosti," keď sa s ním stretol v dynamickom programovaní.

Jednoduchým príkladom nadmernej (extrémnej) dimenzionality je štandardná 1080p obrazovka ($1920 \cdot 1080 = 2073600$ pixlov). Ak by mohol každý pixel zobraziť len bielu alebo čiernu farbu, máme $2^{2073600}$ potencionálnych obrázkov. Ide o nepredstaviteľne veľké číslo. Obrazovka zobrazuje pixly v rôznych farbách (najčastejšie 24-bitové farby – kombinácia 8 bitov pre červenú, zelenú a modrú), čo predstavuje 2^{24} možností pre 1 pixel. Vráťme sa však k optimalizácii.

Veľmi obľúbeným nástrojom na zníženie dimenzie je metóda hlavný komponent [principal component analysis], ktorá dokáže efektívne znížiť dimenziu pozorovaných dát. My sa však venujeme optimalizácii výpočtovo náročnej, prípadne neznámej funkcie $f\colon \mathcal{X} \to \mathbb{R}$, kde $\mathcal{X} \subset \mathbb{R}^D$. Nadmerná dimenzionalita pre nás znamená nadmerný počet vstupov pre účelovú funkciu. V prvej kapitole sme predstavili bayesovskú optimalizáciu, ktorá dokáže nájsť extrém (priblížiť sa k extrému) pre neznáme a výpočtovo náročné funkcie. Základom je optimalizácia multimodálnej akvizičnej funkcie, čo môže byť pre vyššie dimenzie problém. Ďalej predstavíme metódu, ktorá dokáže znížiť dimenziu a tým znížiť výpočtovú náročnosť BO.

2.1 Bayesovská optimalizácia s náhodným preložením

2.1.1 Efektívne dimenzie a náhodné preloženie

V tejto časti budeme vychádzať hlavne z práce od autorov Wang a kol. (2016), ktorí predstavili v praxi dobre fungujúci spôsob zníženia dimenzie. Vychádzame hlavne z predpokladu, že nie všetky vstupy funkcie f sú pre hodnotu funkcie dôležité. Dimenzie ktoré sú dôležité nazveme efektívne a predpokladáme, že efektívne dimenzie už tvoria priestor s nízkou dimenziou (v inej literatúre sa môžeme stretnúť aj s názvom aktívne dimenzie).

Definícia 2.1. *Hovoríme, že funkcia $f\colon \mathbb{R}^D \to \mathbb{R}$ má efektívnu dimenziu d_e, kde $d_e \leq D$, ak:*

- *existuje lineárny podpriestor $\mathcal{T}$ s dimenziou d_e taký, že $\forall \boldsymbol{x}_\top \in \mathcal{T} \subset \mathbb{R}^D$ a $\forall \boldsymbol{x}_\perp \in \mathcal{T}^\perp \subset \mathbb{R}^D$ platí $f(\boldsymbol{x}_\top + \boldsymbol{x}_\perp) = f(\boldsymbol{x}_\top)$, kde $\mathcal{T}^\perp$ je ortogonálny doplnok k $\mathcal{T}$ a*
- *d_e je najmenšie prirodzené číslo spĺňajúce túto vlastnosť.*

$\mathcal{T}$ *nazývame* ***efektívny podpriestor*** *a* $\mathcal{T}^\perp$ ***konštantný podpriestor***.

Definícia 2.1 nám hovorí, že funkčná hodnota sa nezmení pre body z $\mathcal{T}^\perp$, preto to nazývame konštantný podpriestor. Pri vhodnom usporiadaní dimenzií máme

$$\boldsymbol{x}_\top = (x_1, \ldots, x_{d_e}, \underbrace{0, \ldots, 0}_{D-d_e})^\top \text{ a } \boldsymbol{x}_\perp = (\underbrace{0, \ldots, 0}_{d_e}, x_{d_e+1}, \ldots, x_D)^\top,\ x_i \in \mathbb{R}\ \forall i \in 1{:}D.$$

Daná podmienka je veľmi silná a v praxi ťažko dosiahnuteľná. Je však potrebná na teoretické odvodenia a dokázania konvergencie algoritmu. V praxi sa ukázalo, že stačí ak $f(\boldsymbol{x}_\top + \boldsymbol{x}_\perp) \approx f(\boldsymbol{x}_\top)$, k čomu sa dostaneme neskôr. Presné určenie efektívnej dimenzie však nemusí byť možné. Ukázalo sa, že nepotrebujeme poznať presnú efektívnu dimenziu a problém môžeme riešiť tzv. *náhodným preložením* [random embedding].

Veta 2.1. *Majme danú funkciu* $f\colon \mathbb{R}^D \to \mathbb{R}$ *s efektívnou dimenziou* d_e *a náhodnú maticu* $\boldsymbol{A} \in \mathbb{R}^{D\times d}$*, ktorej prvky sú nezávisle generované z* $\mathcal{N}(0{,}1)$ *a* $D \geq d \geq d_e$*. Potom, s pravdepodobnosťou 1, pre každé* $\boldsymbol{x} \in \mathbb{R}^D$ *existuje* $\boldsymbol{z} \in \mathbb{R}^d$ *také, že* $f(\boldsymbol{x}) = f(\boldsymbol{A}\boldsymbol{z})$.

Dôkaz. Viď Wang a kol. (2016).

□

Veta 2.1 hovorí o tom, že pre ľubovoľne dané $\boldsymbol{x} \in \mathbb{R}^D$ a pre danú náhodnú maticu $\boldsymbol{A} \in \mathbb{R}^{D\times d}$ existuje s pravdepodobnosťou 1 bod $\boldsymbol{z} \in \mathbb{R}^d$ taký, že $f(\boldsymbol{x}) = f(\boldsymbol{A}\boldsymbol{z})$. To samozrejme platí aj pre optimum $\boldsymbol{x}^* \in \mathbb{R}^D$. Za daných predpokladov potom stačí optimalizovať funkciu $g(\boldsymbol{z}) = f(\boldsymbol{A}\boldsymbol{z})$ na priestore s nižšou dimenziou d. Bayesovská optimalizácia na takto zúženom priestore sa nazýva *bayesovská optimalizácia s náhodným preložením*, **REMBO** [Random EMbedding Bayesian Optimization].

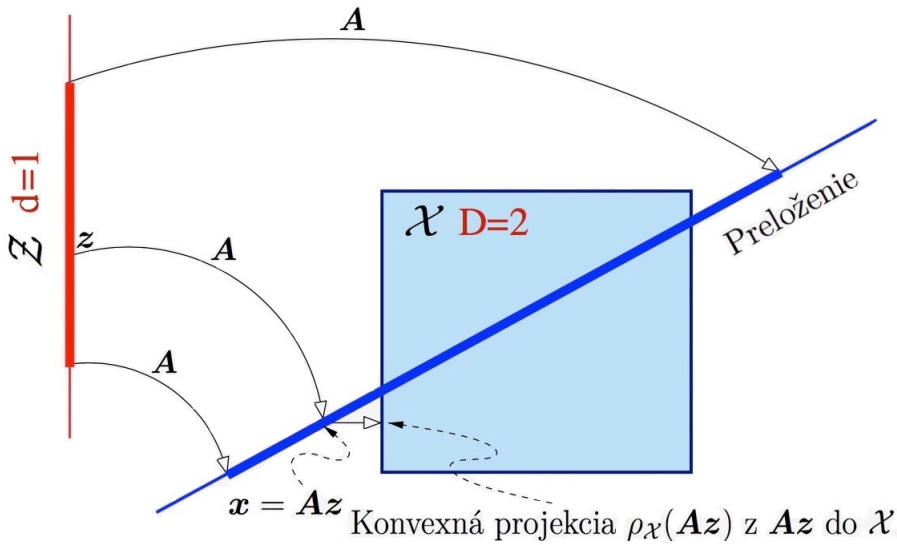
Dôvod pomenovania „preloženie“, prípadne „prekrytie“ [z angl. slova embedding] je viditeľný na obrázku 2.1. Pri optimalizačných úlohách nás väčšinou zaujíma optimalizácia funkcie f na nejakej kompaktnej množine $\mathcal{X} \subset \mathbb{R}^D$, najčastejšie ide o uzatvorené intervaly v dimenzii D, tzv. „krabicové“ ohraničenia. Modrá priamka $\boldsymbol{A}\boldsymbol{z}$ na obrázku 2.1, prípade podpriestor vo vyšších dimenziách, „prekladá“/„prekrýva“ časť množiny $\mathcal{X}$. Funkcia f je väčšinou mimo množiny $\mathcal{X}$ nedefinovaná. Pri náhodnom preložení môže nastať situácia, že vyberieme bod $\boldsymbol{z} \in \mathcal{Z}$ tak, že $\boldsymbol{A}\boldsymbol{z} \notin \mathcal{X}$ (viď časť modrej priamky na obrázku 2.1, ktorá neprekladá množinu $\mathcal{X}$). Keďže f je mimo $\mathcal{X}$ nedefinovaná, náhodné preloženie musí obsahovať aj projekciu $\rho_\mathcal{X}\colon \mathbb{R}^D \to \mathbb{R}^D$. Ide o štandardnú projekciu do nášho kompaktného priestoru $\mathcal{X}$:

$$\rho_\mathcal{X}(\boldsymbol{A}\boldsymbol{z}) = \operatorname*{arg\,min}_{\boldsymbol{x}\in\mathcal{X}} ||\boldsymbol{x} - \boldsymbol{A}\boldsymbol{z}||_2, \tag{2.1}$$

ktorá nám vráti $\boldsymbol{A}\boldsymbol{z}$ ak $\boldsymbol{A}\boldsymbol{z} \in \mathcal{X}$ alebo najbližší bod z $\mathcal{X}$ (hraničný bod), ak $\boldsymbol{A}\boldsymbol{z} \notin \mathcal{X}$. Účelová funkcia v priestore $\mathcal{Z}$ má tvar

$$g(\boldsymbol{z}) = f\Big(\rho_\mathcal{X}(\boldsymbol{A}\boldsymbol{z})\Big). \tag{2.2}$$

Pre úplne definovanie náhodného preloženia ešte potrebujeme opísať spôsob výberu ohraničenej množiny $\mathcal{Z} \subset \mathbb{R}^d$. Táto časť je obzvlášť dôležitá, pretože efektivita REMBO závisí od veľkosti množiny $\mathcal{Z}$. Nájsť optimum pre menšiu $\mathcal{Z}$ je jednoduchšie, ale ak bude $\mathcal{Z}$ príliš malá, tak nemusí obsahovať globálny extrém.



Obrázok 2.1: Ilustrácia náhodného preloženia jednorozmernej množiny $\mathcal{Z}$ do dvojrozmernej krabicovej množiny $\mathcal{X}$. Ak sa zobrazenie $\boldsymbol{Az}$ nachádza mimo $\mathcal{X}$, tak je pomocou projekcie $\rho_{\mathcal{X}}$ zobrazené do $\mathcal{X}$. Množina $\mathcal{Z}$ musí byť dostatočne veľká, aby $\boldsymbol{A}\mathcal{Z}$ preložilo celú množinu $\mathcal{X}$. Obrázok prevzatý od Wang a kol. (2016).

Veta 2.2. *Predpokladajme, že chceme optimalizovať funkciu* $f\colon \mathbb{R}^D \to \mathbb{R}$ *s efektívnou dimenziou* $d_e \leq d$ *pri „krabicových" ohraničeniach* $\mathcal{X} \subset \mathbb{R}^D$, *kde* $\mathcal{X}$ *je centrovaný okolo* $\mathbf{0}$. *Ďalej predpokladajme, že efektívny podpriestor* $\mathcal{T}$ *funkcie* f *je obalom* d_e *bázových vektorov a nech* $\boldsymbol{x}^*_\top \in \mathcal{T} \cap \mathcal{X}$ *je optimum funkcie* f *vnútri* $\mathcal{T}$. *Ak* $\boldsymbol{A}$ *je* $D \times d$ *náhodná matica s nezávislými prvkami z* $\mathcal{N}(0,1)$, *tak existuje optimum* $\boldsymbol{z}^* \in \mathbb{R}^d$ *také, že* $f(\boldsymbol{Az}^*) = f(\boldsymbol{x}^*_\top)$ *a* $||\boldsymbol{z}^*||_2 \leq \frac{\sqrt{d_e}}{\epsilon}||\boldsymbol{x}^*_\top||_2$ *s pravdepodobnosťou aspoň* $1-\epsilon$.

Dôkaz. Viď Wang a kol. (2016). □

Veta 2.2 nám zadáva ohraničenie pre optimum v množine $\mathcal{Z}$ s pravdepodobnosťou aspoň $1-\epsilon$. Keďže hľadáme optimum, tak dané ohraničenie môžeme použiť pre celý priestor $\mathcal{Z}$. Ak $\mathcal{X} = [-1,1]^D$, čo vieme vždy dosiahnuť preškálovaním priestoru $\mathcal{X}$, tak s pravdepodobnosťou aspoň $1-\epsilon$ máme

$$||\boldsymbol{z}^*||_2 \leq \frac{\sqrt{d_e}}{\epsilon}||\boldsymbol{x}^*_\top||_2 \leq \frac{\sqrt{d_e}}{\epsilon}\sqrt{d_e}.$$

Musíme teda zabezpečiť aby $\mathcal{Z}$ obsahovala guľu s polomerom d_e/ϵ centrovanú okolo $\mathbf{0}$. V praxi sa však používajú tak komplexné účelové funkcie, že je veľmi nepravdepodobné aby optimum bolo v rohu krabicového ohraničenia, t.j. $||\boldsymbol{x}^*_\top||_2 < \sqrt{d_e}$.

Wang a kol. (2016) ďalej dokázali vlastnosti REMBO potrebné pre vyslovenie a dokázanie vety o konvergencii algoritmu REMBO. Pomocou simulačnej štúdie pre dimenzie $d \in \{1,\ldots,50\}$ a pre 10000 vygenerovaných matíc $\boldsymbol{A}$ zistili, že s empirickou pravdepodobnosťou $1-\epsilon$, pre klesajúce hodnoty ϵ, môžeme zostaviť $\mathcal{Z}$ ako

$$\mathcal{Z} = \left[-\frac{1}{\epsilon}\max\{\ln(d),1\}, \frac{1}{\epsilon}\max\{\ln(d),1\}\right]^d. \tag{2.3}$$

V ďalších experimentoch zvolili $\epsilon = \ln(d)/\sqrt{d}$, teda $\mathcal{Z} = [-\sqrt{d},\sqrt{d}]^d$ pre $\mathcal{X} = [-1,1]^D$. Danú voľbu použijeme aj my pri aplikácii REMBO.

2.1.2 Algoritmus REMBO

V časti 2.1.1 sme predstavili metódu náhodného preloženia pre zníženie dimenzie priestoru $\mathcal{X}$. Na priestore s nižšou dimenziou $\mathcal{Z}$ už môžeme vykonať bayesovskú optimalizáciu z 1. kapitoly (viď algoritmus 3). Pri BO už postupujeme rovnako ako v nižších dimenziách – určíme apriórnu kovariančnú funkciu (SE alebo Matérn jadro), hyper-parametre a akvizičnú funkciu.

Algoritmus 3 REMBO

1: Generuj náhodnú maticu $\boldsymbol{A} \in \mathbb{R}^{D\times d}$ z $\mathcal{N}(0,1)$
2: Urč ohraničenú množinu $\mathcal{Z} \subset \mathbb{R}^d$
3: Použi algoritmus BO pre optimalizáciu funkcie $g(\boldsymbol{z}) = f\Big(\rho_{\mathcal{X}}(\boldsymbol{A}\boldsymbol{z})\Big)$, $\boldsymbol{z} \in \mathcal{Z}$

Vráťme sa ešte k výsledkom vety 2.2, ktoré hovoria, že ak množinu $\mathcal{Z}$ zostavíme uvedeným spôsobom, t.j. $\mathcal{Z} = [-\sqrt{d},\sqrt{d}]^d$, tak obsahuje optimum s pravdepodobnosťou aspoň $1-\epsilon$. Existuje teda nenulová pravdepodobnosť $\delta \leq \epsilon$, že optimum leží mimo množinu $\mathcal{Z}$. Následná BO na takejto množine $\mathcal{Z}$ potom nenájde globálne optimum. Veľmi jednoduchou možnosťou je vykonať k navzájom nezávislých aplikácii REMBO. Keďže pravdepodobnosť neúspechu jedného REMBO je δ, tak pre k nezávislých spustení máme δ^k. Keďže volíme malé ϵ, tak po použití niekoľkých REMBO už máme dostatočne malú pravdepodobnosť neúspechu.

Wang a kol. (2016) vykonali experimenty na porovnanie niekoľkých prístupov k optimalizácii neznámych funkcií vo vysokých dimenziách a empiricky ukázali výborné výsledky metódy REMBO. Optimalizovali rôzne výpočtovo náročné funkcie, ktorých optimalizácia trvala od niekoľkých minút až po 4-5 dní. Celkovo využili viac ako pol roka čistého času procesora. Časovo najnáročnejšia bola optimalizácia parametrov klasifikácie častí ľudského tela cez náhodné lesy (používane v Kinect – hardware od Microsoftu, ktorý sníma ľudské telo a používa jeho pohyby pri rôznych hrách pre Xbox a pod.). Ukázalo sa, že aj keď bolo všetkých 14 parametrov dôležitých, tak REMBO pre $d = 3$ dosiahol výborné výsledky – lepšie ako náhodné prehľadávanie, avšak klasická BO pre 14 dimenzií dosiahla lepšie výsledky.

Dané výsledky sú veľmi optimistické a ukazuje sa, že aj pri nevhodnom odhade efektívnej dimenzie REMBO nezlyháva a stále dosahuje výsledky lepšie alebo porovnateľné s inými dostupnými metódami. V praxi pri optimalizácii výpočtovo náročných funkcií sa väčšinou uspokojíme s hodnotou blízkou k globálnemu optimu – základom je dostať výsledok po rozumnom počte iterácií. Vhodný spôsob ako sa dostať k dobrým výsledkom je použiť 4 nezávislé spustenia REMBO s počtom iterácií 125, teda celkovo funkciu vypočítame 500-krát (autori často používali tuto konfiguráciu a v každom príklade to vykonali 50 krát aby získali vhodný odhad priemeru a intervalu spoľahlivosti pre porovnanie s inými metódami).

Príklad

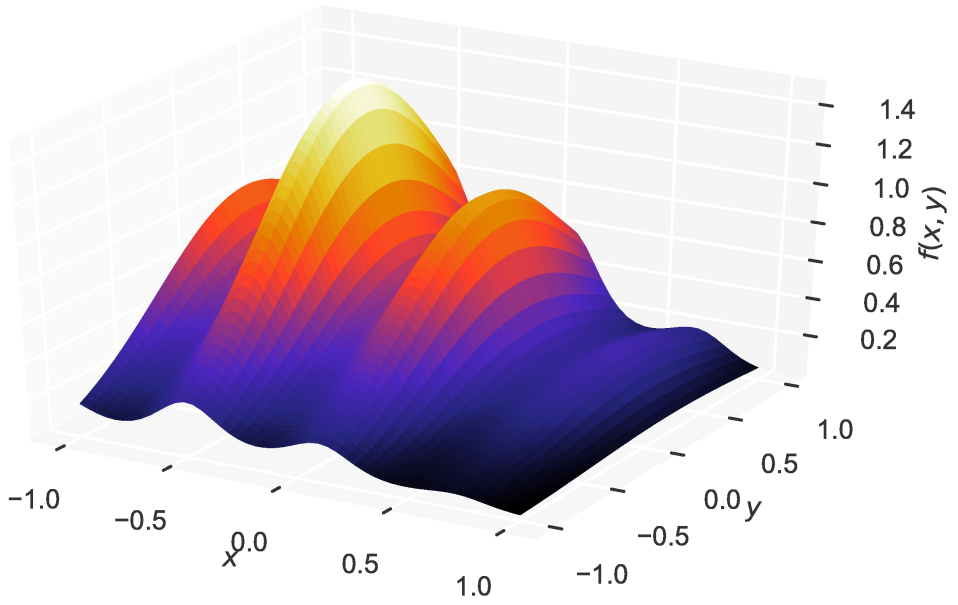
Dvojrozmernú funkciu $h\colon [-1,1]^2 \to \mathbb{R}$, definovanú ako

$$h(x,y) = \exp\left(-(x+0.5)^2 - (y-0.3)^2 + 0.4\cos(10x+5)\right)$$

vložíme do funkcie $f\colon [-1,1]^{30} \to \mathbb{R}$ s efektívnou dimenziou $d_e = 2$ tak, že nebudeme vedieť ktoré dimenzie sú efektívne. Pre dané $i_1, i_2 \in \{1,\dots,30\}$ máme

$$f(\boldsymbol{x}) = h(x_{i_1}, x_{i_2}).$$

Z definície h vidíme, že maximum je v bode $(x,y)_{\text{opti}} = (-0.5, 0.3)^\top$ a jeho hodnota je $h((x,y)_{\text{opti}}) \doteq 1.4918$. Funkcia h má na $[-1,1]^2$ viac lokálnych maxím (viď obrázok 2.2).



Obrázok 2.2: 3-rozmerný graf funkcie h z príkladu pre REMBO.

Použili sme algoritmus 3 spolu s algoritmom 2 pre BO v nižšej dimenzii. Podobne ako v prvej kapitole, použijeme SE jadro s hyper-parametrom $\theta_0 = 1$, $\ell \in [0,100]$ a $\sigma_y^2 = 0.001$. Aby sme znížili pravdepodobnosť neúspechu z vety 2.2, tak aplikujeme 3 nezávisle REMBO s $n = 120$ pre rôzne dimenzie, $d \in \{2,3\}$. Použijeme $\epsilon = \ln(d)/\sqrt{d}$ a teda $\mathcal{Z} = [-\sqrt{d}, \sqrt{d}]^d$. Výsledky máme v tabuľke 2.1.

	1	2	3	max	ϵ	ϵ^3
$d = 2$, $n = 120$	1.4659	1.4915	1.4680	1.4915	0.4901	0.1177
$d = 3$, $n = 120$	1.4805	0.8724	1.4727	1.4805	0.6343	0.2552

Tabuľka 2.1: Výsledky 3-och navzájom nezávislých aplikácii REMBO pre $d = 2$ a $d = 3$.

Ako vidíme z tabuľky 2.1, pre $d = 2$ máme veľmi blízku hodnotu k reálnemu maximu a pravdepodobnosť neúspechu je ≤ 0.1177 (pri 3-och nezávislých

REMBO). Pre $d = 3$ se dosiahli mierne horšie výsledky a v jednej aplikácii sme dokonca našli iba lokálne optimum s hodnotou približne 0.87 – v danej aplikácii REMBO sa globálne maximum pravdepodobne nenachádzalo v zúženom priestore. Keď sa však pozrieme na pravdepodobnosť neúspechu, tak ta je pre 1 aplikáciu REMBO až 0.63 a pre 3 nezávislé aplikácie 0.25.

Poznámka. Hodnota $\epsilon = \ln(d)/\sqrt{d}$ pre nižšie hodnoty stúpa ($d < 8$), až potom konverguje k 0.

2.2 Ďalšie metódy BO pre vysoké dimenzie

Efektívna globálna optimalizácia výpočtovo náročných/neznámych funkcií je stále otvorený problém. V krátkosti predstavíme ďalšie 3 prístupy k bayesovskej optimalizácii vo vysokých dimenziách. Prvá metóda bude mať podobný predpoklad ako REMBO, t.j. efektívne dimenzie, ďalšia bude vychádzať z predpokladu separovateľnosti účelovej funkcie a posledná predstavená metóda nemá žiadne špeciálne predpoklady a snaží sa zefektívniť optimalizáciu akvizičnej funkcie vo vysokých dimenziách.

2.2.1 Vysoko-dimenzionálna bayesovská optimalizácia HD BO

Bayesovskej optimalizácii vo vysokých dimenziách sa venovali aj Chen a kol. (2012). Podobne ako REMBO vychádzali z predpokladu efektívnych dimenzií (v tomto článku ich nazývali aktívne dimenzie). Predpoklady sú teda rovnaké ako pri REMBO, označíme efektívne dimenzie $\mathcal{A} \subset \{1,\dots,D\}$ (aktívne). Autori predstavili *hierarchické diagonálne vzorkovanie* [**HDS** – Hierarchical Diagonal Sampling], ktorého úlohou je nájsť konkrétne efektívne dimenzie. REMBO využíval náhodné preloženie a nepotrebovali sme vedieť ktoré dimenzie sú efektívne. V krátkosti opíšeme HD BO metódu pre $f\colon [-1,1]^D \to \mathbb{R}$.

HDS bude deliť dimenzie $\{1,\dots,D\}$ pomocou stromu – dimenzie sa budú vetviť podľa toho či sú aktívne alebo neaktívne. Na konci stromu keď sa pozrieme na jednotlivé listy tak zistíme ktoré dimenzie sú aktívne a ktoré nie. Každý uzol v strome reprezentuje nejakú množinu dimenzií $I \subseteq \{1,\dots,D\}$. Na začiatku náhodné vygenerujeme bod $\boldsymbol{x}^{(0)} \in [-1,1]^D$. Definujme $\boldsymbol{x}_I\colon [-1,1] \to [-1,1]^D$ nasledovne:

$$x_{I,i}(z) = \begin{cases} z & \text{ak } i \in I, \\ x_i^{(0)} & \text{inak,} \end{cases}$$

pre $i \in \{1,\dots,D\}$, kde $z \in [-1,1]$. Pre všetky dimenzie z I definujeme rovnaké z, preto názov diagonálne. Zostrojíme 1-dimenzionálnu funkciu $f_I\colon [-1,1] \to \mathbb{R}$

$$f_I(z) = f(\boldsymbol{x}_I(z)),$$

ktorá je apriórne 1-rozmerného GP s kovariančnou funkciou SE jadro, ktorého hyper-parameter $\boldsymbol{\Sigma}^{-1} = \frac{2a_I}{b^2}$, kde $a_I = |\mathcal{A} \cap I|$ a b je hyper-parameter mierky, teda

$$k(\boldsymbol{x},\boldsymbol{x}') = \theta_0 \exp\left(-\frac{a_I}{b^2}||\boldsymbol{x} - \boldsymbol{x}'||_2\right), \quad \theta_0 > 0.$$

Na identifikáciu či I obsahuje aktívne premenné je postačujúce vhodne testovať, či daný GP dobre charakterizuje funkciu f. Tento test sa robí pomocou pomeru vierohodnosti. Chen a kol. (2012) predstavili 2 rôzne metódy – sekvenčný test pomeru vierohodnosti pre konečné diferencie (FDT) a sekvenčný test pomeru vierohodností podľa GP (GPT). Každá metóda má svoje výhody. Dané prístupy predstavíme len veľmi stručne.

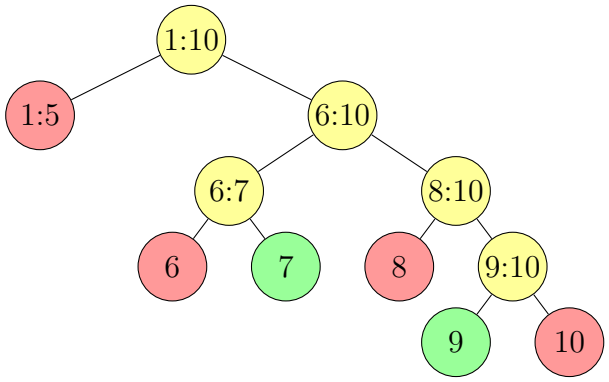
Podstatou metódy FDT je určenie rozdelenia pre diferenciu zašumených pozorovaní funkcie f, t.j. $y_I(z_j) = f_I(z_j) + \varepsilon_j$, kde $\varepsilon_j \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, \sigma_y^2)$ pre $j \in 1{:}n$. Diferenciu označíme $dy_I = y_I(z) - y_I(z+\delta)$. FDT využíva predpoklad $dy_I \sim \mathcal{N}(0, \sigma_a^2)$, kde $\sigma_a^2 = 2[1 - \exp(-a\delta^2/b^2)\theta_0 + \sigma_y^2]$, $\theta_0 > 0$ je z SE jadra a a je počet aktívnych dimenzií. δ vyberieme náhodne a pomerom vierohodnosti testujeme hypotézu

$$\mathrm{H}_0\colon \operatorname{var} dy_I = 2\sigma_y^2 \text{ (žiadne aktívne dimenzie) proti}$$
$$\mathrm{H}_1\colon \operatorname{var} dy_I = \sigma_a^2 \text{ (a=1, jedna aktívna dimenzia).}$$

Pomer vierohodnosti vypočítame ako sumu pomerov vierohodností pre n vzoriek $\{dy_I(z_j)\}_{j=1}^n$. Novú vzorku z_{n+1} určíme náhodne z $[-1{,}1]$, vzorkovanie pre daný uzol skončíme ak pomer vierohodnosti prekročí prah Θ_1 (H_1 prijímame) alebo klesne pod Θ_0 (H_0 nezamietame).

GPT využíva GP, teda $y_{I,n+1}|z_{n+1}, z_{1:n}, y_{1:n} \sim \mathcal{N}(\mu_a^n(z_{n+1}), (\sigma_a^n(z_{n+1}))^2)$, kde aposteriórna stredná hodnota a rozptyl sa vypočítajú analogicky k (1.9). Z definície $\boldsymbol{x}_I(z)$ vidíme, že už vypočítane funkčné hodnoty môžme používať v ďalších iteráciách. Novú vzorku vypočítame ako maximum z UCB (horný konfidenčný interval) pre pomery vierohodností z predchádzajúcich iterácií. Pre novú vzorku z_{n+1} vypočítame pomer vierohodnosti (rovnaké hypotézy ako pre FDT). Test pre daný uzol (súčet všetkých vierohodností) skončí prekročením prahu ako v FDT.

Pri HDS zvolíme maximálny počet vzoriek n a postupne zmenšujeme množinu I tak, že kontrolujeme prijatie hypotézy H_1 (prvý uzol je $I = \{1, \ldots, D\}$). Ak je H_1 prijatá tak buď rozdelíme I na dve rovnako veľké podmnožiny a ďalej vzorkujeme na nich, alebo ak $|I| = 1$ tak danú dimenziu pridáme medzi aktívne a ďalej sa ňou nezaoberáme. Ak nezamietneme hypotézu H_0 pre nejaké I, tak daný uzol už ďalej nerozdeľujeme, stáva sa konečným listom stromu. Pre lepšiu predstavu viď obrázok 2.3.



Obrázok 2.3: Ukážka stromu hierarchického diagonálneho vzorkovania pre 10 dimenzií, kde {7,9} sú aktívne. Uzly v strome reprezentujú množiny I a pre dané uzly sa vykonáva opísaný test. Farby uzlov reprezentujú výsledok testu: zelená – prijatá H_1, červená – nezamietnutá H_0 a žltá – prijatá H_1 pre I, avšak $|I| \neq 1$.

Týmto sme po n výpočtoch funkcie f získali množinu efektívnych dimenzií a ďalej môžeme použiť klasickú bayesovskú optimalizáciu pre zistené dimenzie $\mathcal{A}$ a za neefektívne dimenzie dosadíme hodnoty z vygenerovaného bodu $\boldsymbol{x}^{(0)}$. Opäť z definície $\boldsymbol{x}_I(z)$ vidíme, že môžeme použiť body $\boldsymbol{x}_{1:n}$ a $y_{1:n}$ ako dáta, lebo

$$x_{j,i}(z) = \begin{cases} z_j & \text{ak } i \in \mathcal{A}, \\ x_i^{(0)} & \text{inak}, \end{cases} \quad \forall j \in 1{:}n.$$

V prípade niektorých bodov vypočítaných v HDS sú neefektívne dimenzie rovné z_j a nie $x_i^{(0)}$, čo však nevadí, lebo neefektívne dimenzie nemajú vplyv na hodnotu funkcie (prípadne majú veľmi malý vplyv). Autori použili akvizičnú funkciu UCB so špeciálnym určením parametra β. Ak sa cez HDS nezistia efektívne dimenzie tak sa používa klasická BO pre vysoké dimenzie, čo nemusí dosiahnuť vhodné výsledky.

2.2.2 Aditívna BO

Kandasamy a kol. (2015) predstavili BO vo vysokých dimenziách pre separovateľnú účelovú funkciu. Predpokladajme, že $\mathcal{X} = [0{,}1]^D$ a funkcia f je separovateľná:

$$f(\boldsymbol{x}) = f^{(1)}(\boldsymbol{x}^{(1)}) + f^{(2)}(\boldsymbol{x}^{(2)}) + \cdots + f^{(M)}(\boldsymbol{x}^{(M)}), \tag{2.4}$$

kde $\boldsymbol{x}^{(j)} \in \mathcal{X}^{(j)} = [0{,}1]^{d_j}$ sú komponenty z nižšej dimenzie a $\mathcal{X}^{(j)}$ sú navzájom disjunktné nízko-dimenzionálne priestory. $\boldsymbol{x} = \cup_j \boldsymbol{x}^{(j)}$ a $\mathcal{X} = \cup_j \mathcal{X}^{(j)}$. Skupinové dimenzie sú ohraničené $d_j \leq d \ll D$. Dôležitým predpokladom pre zostrojenie aditívneho GP je nezávislosť funkcií $f^{(j)}$ $\forall j$. Analogicky ako v prvej kapitole určíme apriórne rozdelenie

$$f^{(j)} \sim \mathcal{GP}(\mu_0^{(j)}, \kappa^{(j)}),$$

kde $\mu_0^{(j)} \colon \mathcal{X}^{(j)} \to \mathbb{R}$ je apriórne zvolená funkcia strednej hodnoty (BUNV $\mu_0^{(j)} = 0$) a $\kappa^{(j)} \colon \mathcal{X}^{(j)} \times \mathcal{X}^{(j)} \to \mathbb{R}$ je apriórne zvolená kovariančná funkcia (napr. pozitívne definítne jadro). Z predpokladu nezávislosti potom máme $f \sim \mathcal{GP}(\mu_0, \kappa)$, kde $\mu_0 = \sum_j \mu_0^{(j)}$ a $\kappa(\boldsymbol{x},\boldsymbol{x}') = \sum_j \kappa^{(j)}(\boldsymbol{x}^{(j)}, \boldsymbol{x}^{(j)\prime})$.

Ďalej pokračujeme ako v časti 1.2.1 o regresii podľa GP. Na základe vypočítaných funkčných hodnôt (dáta $\mathcal{D}_n$) určíme podmienené aposteriórne rozdelenie pre každú funkciu $f^{(j)}$. Keďže f je aditívna s nezávislými zložkami, tak akviziční funkcia je tiež aditívna:

$$\tilde{\alpha}(\boldsymbol{x}; \boldsymbol{\theta}, \mathcal{D}_n) = \sum_{j=1}^{M} \alpha^{(j)}(\boldsymbol{x}^{(j)}; \boldsymbol{\theta}^{(j)}, \mathcal{D}_n), \tag{2.5}$$

kde $\alpha^{(j)}$ je akviziční funkcia pre $f^{(j)}$ a $\boldsymbol{\theta}^{(j)}$ sú hyper-parametre pre apriórnu kovariančnú funkciu $\kappa^{(j)}$. Maximalizácia akvizičnej funkcie potom môže byť vykonaná samostatne pre každú $\alpha^{(j)}$ na nízko-dimenzionálnom priestore $\mathcal{X}^{(j)}$. Autori používali akviziční funkciu UCB so špeciálnou voľbou parametra β a za apriórnu kovariančnú funkciu zvolili pozitívne definítne jadro SE alebo Matérn.

Predpoklad separovateľnosti je opäť veľmi striktný a pri porušení tohoto predpokladu už metóda nedosahuje prijateľné výsledky. Ak však vieme, že výpočtovo náročná funkcia je separovateľná a vieme rozložiť dizajnový priestor $\mathcal{X}$, tak tento

prístup dosahuje najlepšie výsledky. Ak funkcia nie je separovateľná, tak viacerými aplikáciami tejto metódy pre rôzne rozloženia priestoru $\mathcal{X}$ môžeme dosiahnuť aj prijateľný výsledok (je však otázne či je to efektívne). Autori sa v ďalšom výskume chcú venovať voľbe d a M pre neseparovateľné funkcie.

2.2.3 Elastický GP

REMBO aj HD BO predpokladali existenciu efektívnych dimenzií. Aditívna BO zas predpokladala separovateľnosť účelovej funkcie. Tieto predpoklady sú veľmi obmedzujúce a v praxi často nesplnené. Funkčnosť a modifikácia týchto prístupov pri nesplnených predpokladoch sú stále predmetom štúdie.

Li a kol. (2016) sa neobmedzovali na rôzne striktné predpoklady, ale sústredili sa na zlepšenie optimalizácie akvizičnej funkcie. Problém vysokých dimenzií je hlavne pri globálnej optimalizácii multimodálnej akvizičnej funkcie (s GP sa dobre pracuje aj vo vyšších dimenziách). Akvizičná funkcia je ľahko spočítateľná a diferencovateľná. Problém pri vysokých dimenziách je, že je väčšinou plochá s niekoľkými vrcholmi. Pri plochej funkcii zlyhávajú metódy založené na gradientoch.

Autori predstavili elastický GP, pri ktorom aktívne menia hyper-parameter rozptylu (dostatočne veľký rozptyl pre apriórne rozdelenie ovplyvní aposteriórne rozdelenie, teda akvizičná funkcia nebude až taká plochá). Pri globálnej maximalizácii akvizičnej funkcie najskôr postupne zvyšujú hyper-parameter rozptylu a pre každý hyper-parameter hľadajú maximum a overujú, či sa líši od štartovacieho bodu. Ak nájdu nové maximum, rôzne od štart bodu, tak začnú postupne zmenšovať hyper-parameter a hľadať maximum pre nižší rozptyl (štartovací bod sa vždy mení na nové maximum). Teda zvýšením hyper-parametra rozptylu nájdu maximum pre neplochú akvizičnú funkciu a postupným zmenšovaním nájdu maximum aj pre plochú akvizičnú funkciu.

Metóda dosahuje dobré výsledky pre vysoké dimenzie (D okolo 50), avšak zvyšovaním dimenzií sa celkovo zvyšuje časová náročnosť maximalizácie vysokodimenzionálnej akvizičnej funkcie. Hlavnou výhodou je, že nepotrebujeme žiadne dodatočné predpoklady pre účelovú funkciu f.

3. Aplikácie

V predchádzajúcich kapitolách sme predstavili jednu z metód tzv. efektívnej globálnej optimalizácie, bayesovskú optimalizáciu. Prešli sme teoretické základy pre BO v nízkych aj vysokých dimenziách, ukázali sme optimalizáciu jednoduchých funkcií a aspoň stručne spomenuli ďalšie rozšírenia. Ešte pred tým ako si ukážeme aplikácie na praktických úlohách je vhodné pripomenúť, že aj keď sme videli veľmi dobré výsledky, pre mnohé zložité funkcie/algoritmy existujú efektívnejšie metódy optimalizácie. Bayesovskú optimalizáciu je vhodné použiť ak sú iné metódy nepoužiteľné, prípadne sa nám ponúka iba náhodné prehľadávanie. V prípadne výpočtovo náročných funkcií, pre ktoré nie sú vytvorené konkrétne optimalizačné metódy, nám BO dáva veľmi dobré výsledky. Najvhodnejšie je použiť BO na neznáme [black-box] funkcie.

Táto kapitola je venovaná optimalizácii funkcií na praktických príkladoch v rôznych oblastiach. Daným oblastiam sa nebudeme podrobne venovať, pretože to nie je náplňou tejto práce. Funkcie aspoň stručne opíšeme, no môžeme ich vnímať ako neznáme funkcie.

3.1 Optimalizácia parametrov SVM pre klasifikáciu

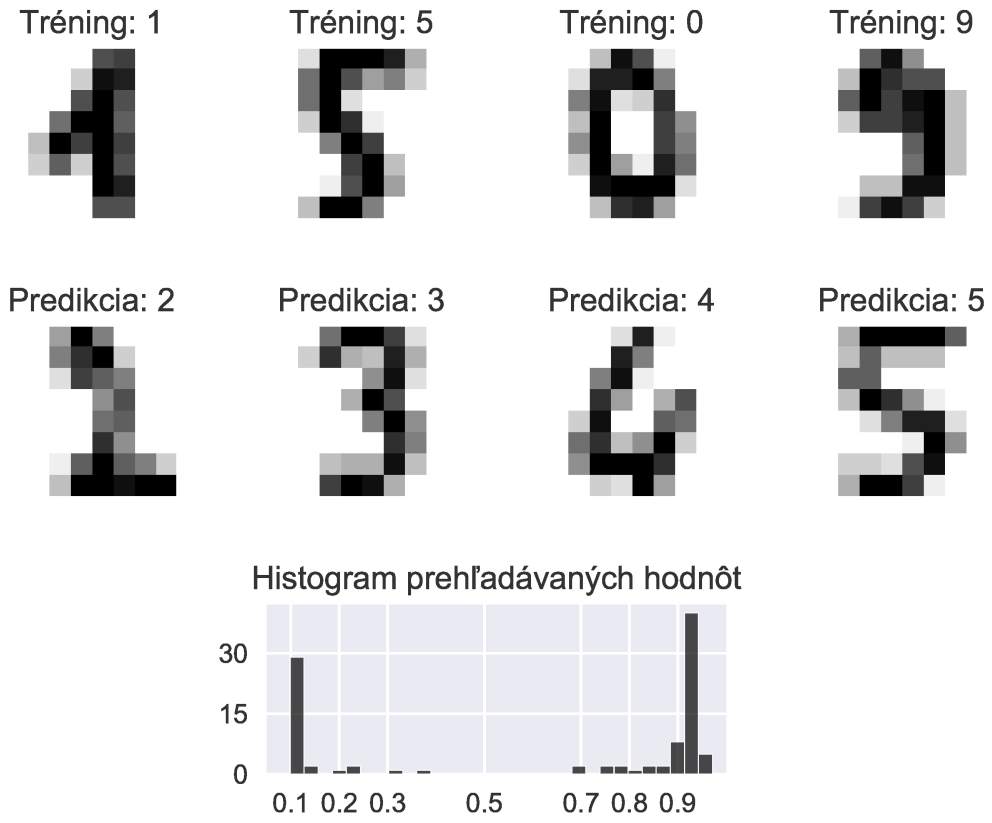
Našou úlohou je nájdenie vhodných parametrov pre klasifikačný problém pomocou metódy podporných vektorov, SVM [support vector machines]. Daná metóda je implementovaná v knižnici scikit-learn v Python™. Hľadáme 2 parametre (penalizačný parameter pre chybu C a parameter mierky pre RBF jadro γ) tak, aby sme maximalizovali úspešnosť predikcie.

Použijeme klasifikačný problém na rozoznanie rukou písaných číslic počítačom – ide o ukážkovú úlohu v dokumentácii k SVM klasifikácii (Jones a kol., 2001–). Vstupom je databáza obrázkov o veľkosti 8×8 pixlov, kde každý pixel môže mať bielu, čiernu alebo rôzny typ sivej farby (vyjadrené číselne). Pre lepšiu vizualizáciu viď obr. 3.1. Databázu rozdelíme na tréningovú podmnožinu [in-sample] a testovaciu podmnožinu [out-of-sample], pričom na in-sample klasifikáciu natrénujeme a z out-of-sample získame pravdepodobnosť úspechu predikcie. Databázu s 1797 číselnými vektormi dĺžky 64 rozdelíme pomerom 70:30.

V dokumentácii použili predvolené hodnoty $C = 1$ a $\gamma = 0.001$. Podľa rôznych odporúčaní (závisí však od typu úlohy na klasifikáciu) a po niekoľkých spusteniach daného algoritmu sa dajú nájsť „rozumné“ intervaly pre hľadané parametre – použijeme $C \in (0.1,2)$ a $\gamma \in (0.0001,0.1)$. Spustíme 100 krokov klasickej BO s akvizičnou funkciou $\alpha_{EI}(\boldsymbol{x};\boldsymbol{\theta},\mathcal{D}_n)$ a so SE jadrom s $\theta_0 = 1$.

Autori dosiahli pravdepodobnosť úspechu predikcie s predvolenými parametrami rovnú 0.970. Po aplikácii BO sme zistili, že ide pravdepodobne o skokovitú lineárnu funkciu. Čo je prirodzené keďže máme testovaciu množinu iba o veľkosti 540 a maximalizujeme podiel počtu správnych predikcií a veľkosti množiny. Našli sme maximálnu hodnotu 0.972 pre $C \doteq 1.94$ a $\gamma \doteq 0.0008$. Čo je iba o niečo viac ako pri použití predvolených parametrov. Malá zmena parametrov nemá na hodnotu funkcie vplyv.

Na obrázku 3.1 máme okrem ukážky rukou písaných číslic aj histogram hodnôt (pravdepodobnosti úspechu) pre parametre, ktoré sme použili v BO. Vidíme, že môžeme dosiahnuť buď veľmi zlú alebo naopak veľmi dobrú predikciu.



Obrázok 3.1: Ukážka obrázkov rukou písaných číslic. V prvom riadku máme náhodne vybrané obrázky z in-sample aj so skutočnou hodnotou. V druhom riadku máme obrázky z out-of-sample aj s predikciou. Pre zaujímavosť sme pridali aj histogram vypočítaných funkčných hodnôt z BO.

3.2 Bayesovská optimalizácia obchodnej stratégie

V tejto časti budeme hľadať najvhodnejšie parametre pre obchodný algoritmus maximalizáciou výstupu z *back-testu*. Nejde o optimalizáciu váh portfólia v štýle Markowitza, ale o obchodnú stratégiu, ktorá robí nepravidelné nákupy a predaje. Halls-Moore (2016) v svojej e-knihe opisuje spôsob použitia rôznych metód strojového učenia, prípadne časových radov, v algoritmickom obchodovaní na burze. Autor vytvoril knižnicu `qstrader`, v ktorej sa dá jednoducho orientovať a slúži hlavne na back-test rôznych stratégií.

V práci využijeme algoritmus párového obchodovania typu *mean-reversion*[1]

[1] „Návrat k strednej hodnote“ [mean-reversion] znamená, že ceny sa časom približujú/vrátia k dlhodobej strednej hodnote. Párové obchodovanie je veľmi rozšírená stratégia a v tomto prípade sa aplikuje mean-reversion na dvojicu aktív – „ceny aktív sa po vzájomnom vychýlení vrátia späť k strednej hodnote.“

pre ETF ([exchange traded funds] – fondy obchodované na burze) na US dlhopisy. Pri párovom obchodovaní sa snažíme vhodne odhadnúť závislosť medzi dvoma aktívami, konkrétne TLT (ETF na 20+ ročné dlhopisy) a IEI (ETF na 3-7 ročné dlhopisy). Na sledovanie závislosti použijeme lineárnu regresiu cez Kalmanove filtre (budeme ju aplikovať postupne v čase). Ak je daná závislosť nejakým spôsobom porušená (pozorujeme veľké odchýlenie od odhadnutej strednej hodnoty), tak jedno aktívum nakupujeme a druhé predávame. Aktíva sú zvolené tak aby existencia závislosti ich cien nebola spochybniteľná. Môžu sa použiť aj iné aktíva, napr. akcie, avšak nájsť vhodne korelované akcie nie je jednoduchá úloha.

Stručný prehľad stratégie

Stavová rovnica a rovnica pozorovaní pre lineárnu regresiu cez Kalmanov filter:

$$\begin{aligned} \boldsymbol{\beta}_{t+1} &= \boldsymbol{I}\boldsymbol{\beta}_t + \boldsymbol{v}_t, \quad t = 1,2,\dots, \\ P_{TLT,t} &= (1,P_{EIE,t})\boldsymbol{\beta}_t + w_t, \quad t = 1,2,\dots, \end{aligned}$$

kde $\boldsymbol{\beta}_t = (\beta_{0,t},\beta_{1,t})^\top$ je stavový vektor (parameter pri regresii), $P_{TLT,t}$ je pozorovanie – cena aktíva TLT, $\boldsymbol{I}$ je jednotková matica parametrov stavovej rovnice, $(1,P_{EIE,t})$ je vektor parametrov pozorovaní a $\boldsymbol{v}_t, w_t$ sú náhodné zložky. $\boldsymbol{v}_t \overset{\text{iid}}{\sim} \mathcal{N}(\boldsymbol{0},\boldsymbol{V}(\delta))$, $w_t \overset{\text{iid}}{\sim} \mathcal{N}(0,\sigma_w^2)$ a $\text{cov}(\boldsymbol{v}_s,w_t) = 0\ \forall s,t = 1,2,\dots$. $\boldsymbol{V}(\delta) = \frac{\delta}{1-\delta}\boldsymbol{I}$, δ a σ_w^2 sú parametre, ktoré potrebujeme určiť.

Filtrovanie, vyrovnávanie a predikcia sú už vykonávané známym spôsobom (viď Cipra (2013)). Označme $e_t = P_{TLT,t} - \hat{P}_{TLT,t|t-1}$, kde $\hat{P}_{TLT,t|t-1} = \mathsf{E}_{t-1}(P_{TLT,t})$ je predikcia z času $t-1$ do t a $Q_t = \text{var}_{t-1}(P_{TLT,t})$. Pri vzniku daných situácii vykonáme nasledujúce transakcie:

- $e_t < -\sqrt{Q_t}$: kúp M jednotiek TLT a predaj $\lfloor \beta_{1,t} M \rfloor$ jednotiek IEI (otvorenie dlhej pozície);
- $e_t \geq -\sqrt{Q_t}$: uzavri všetky pozície v TLT a IEI (uzavretie dlhej pozície);
- $e_t > \sqrt{Q_t}$: predaj M jednotiek TLT a kúp $\lfloor \beta_{1,t} M \rfloor$ jednotiek IEI (otvorenie krátkej pozície);
- $e_t \leq \sqrt{Q_t}$: uzavri všetky pozície v TLT a IEI (uzavretie krátkej pozície).

Porovnaním vypočítanej chyby predikcie vieme určiť, či sme sa od strednej hodnoty vychýlili dostatočne ďaleko a očakávame zmenu cien smerom späť k strednej hodnote. Ak je chyba záporná a dostatočne vzdialená od strednej hodnoty tak očakávame buď nárast ceny TLT alebo pokles ceny IEI. Podobne v opačnej situácii. Ak sa chyba dostane naspäť do intervalu $[-\sqrt{Q_t},\sqrt{Q_t}]$, tak otvorené pozície uzavrieme. Zvolili sme $M = 2000$ aby nákup/predaj aktív bol možný s počiatočnou investíciou 100000 USD.

Aplikácia BO pre danú stratégiu

Danú stratégiu aplikujeme a back-testujeme cez qstrader v Python™ a môžeme sa na ňu pozerať ako na neznámu funkciu. Denné ceny aktív získame z voľne prístupnej *Yahoo!* databázy. Použijeme obdobie 2010-2016 (vrátane) a pre každý obchod započítame bežné transakčné náklady platné pre americký akciový trh

od maklérskej spoločnosti Interactive Brokers LLC. Vhodné parametre nájdeme maximalizáciou výnosu a minimalizáciou rizika. Vhodným meradlom rizika pri obchodných stratégiách je maximálny pokles portfólia [max drawdown]. Za výnos budeme považovať celkový výnos portfólia na konci obdobia. Budeme maximalizovať podiel celkového výnosu a maximálneho poklesu.

Problémom pri optimalizácii parametrov takýchto stratégii je možná nespojitosť funkcie a tiež nebezpečenstvo pre-učenia [overfitting] na historických dátach. Pre parameter δ môžeme určiť ohraničenie (10^{-7},10^{-4}) (pre hodnoty vyššie 10^{-4} nastáva príliš málo obchodných príležitosti a kumulatívny výnos je väčšinou konštantný), autor zvolil $\delta = 10^{-4}$. Parameter σ_w^2 musíme zvoliť veľmi veľký, pretože back-testovací algoritmus v qstrader používa vysoký multiplikátor pre ceny aktív a hodnotu portfólia (multiplikátor spresňuje výpočty na vyšší počet desatinných miest). Vyskúšaním niekoľkých parametrov sme objavili vhodný interval (10^{25},10^{28}), pre $\sigma_w^2 < 10^{25}$ nie je žiaden vplyv na výsledok back-testu. Autor v tomto prípade zvolil hodnotu 0.001, čo je však hodnota s nulovým vplyvom a back-test dáva rovnaký výsledok aj pre 10^{25}. Autor však použil iné časové obdobie.

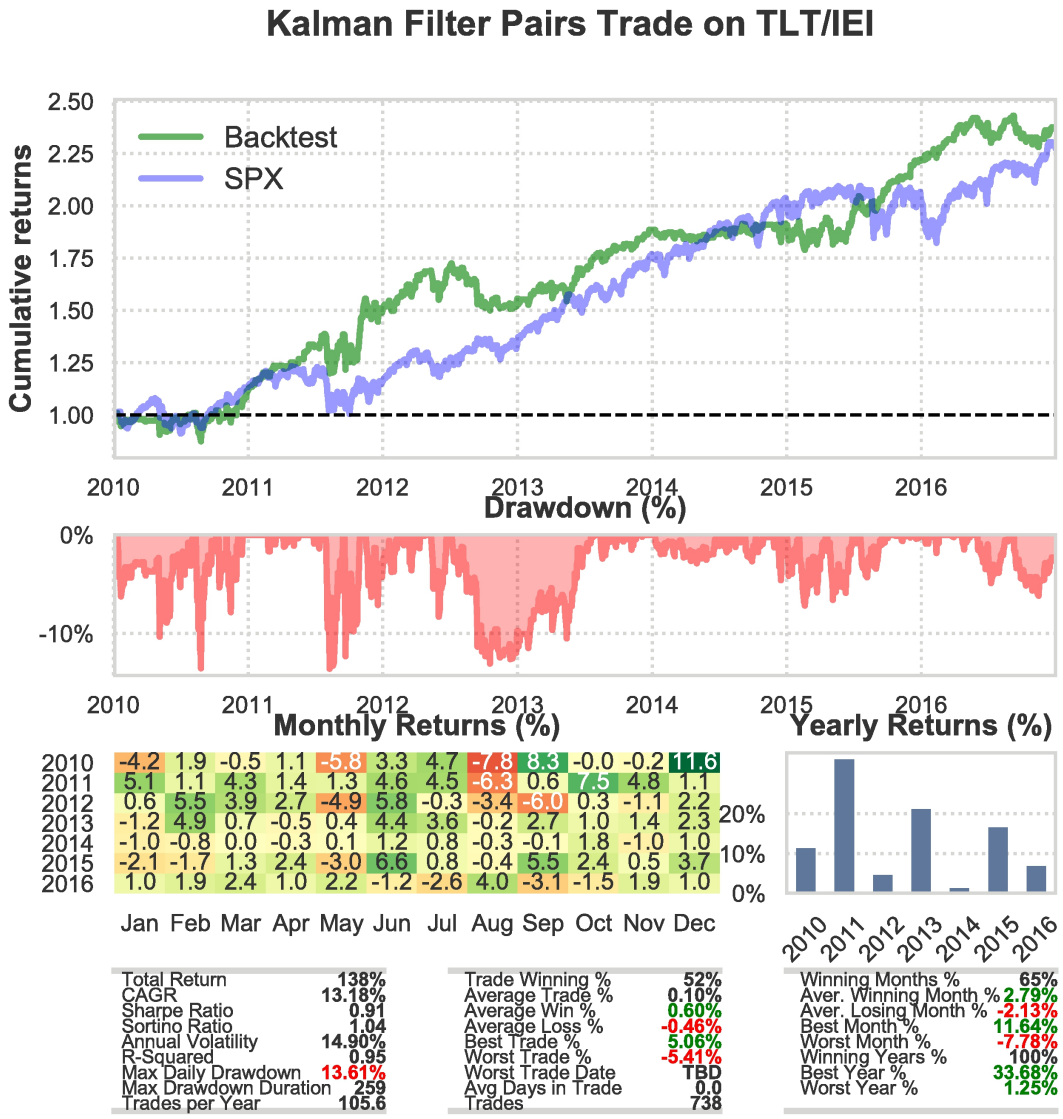
Výsledky BO

Použitím 120 krokovej BO s predvolenými parametrami (podobne ako v predchádzajúcich príkladoch) sme získali maximum podielu výnosu a maximálneho prepadu 10.11, čo je výrazne lepší výsledok ako dosiahlo nastavenie parametrov autorom (0.23). Pre naše parametre sme dosiahli výnos 138% v porovnaní so 6% s pôvodnými parametrami. Všetky výstupy a pre investora zaujímavé informácie sa nachádzajú na obrázku 3.2 na nasledujúcej strane.

Hodnota optimálnych parametrov je $\delta_{\text{opti}} \doteq 2.79 \cdot 10^{-6}$ a $\sigma^2_{w,\text{opti}} \doteq 2.47 \cdot 10^{27}$. Pri menšej zmene parametrov sa funkčné hodnoty menia avšak stále sa pohybujú v intervale (8,11), čo je výrazne vyššia hodnota ako pre parametre vzdialené od optimálnych.

Ak sa pozrieme na obrázok 3.2 pri porovnaní kumulatívnych výnosov vidíme, že sa nám podarilo „poraziť“ trh. Výnos investície do indexu S&P 500 bol za dané obdobie 127%. Naša stratégia dosahuje najlepšie výsledky keď index mierne stagnuje, pri prudkom raste indexu sa jej prestáva dariť a kumulatívny výnos stagnuje.

Ak zúžime ohraničenie pre δ na (10^{-7},10^{-5}), môžeme dosiahnuť výsledok až 11.19 (tu sme však použili $\alpha_{UCB}(\boldsymbol{x};\boldsymbol{\theta},\mathcal{D}_n)$). Ak ďalej zúžime pôvodne veľmi široký interval pre σ_w^2, tak už nedosiahneme až tak vysoké hodnoty. Takáto optimalizácia však môže viesť k pre-učeniu. Aby sme zabránili pre-učeniu na historických dátach, v praxi môžeme aplikovať aj tzv. *walk forward* optimalizáciu (prípadný preklad môže byť aj „kĺzavá optimalizácia“ so stanovenou dĺžkou okna). Napríklad by sme mohli optimalizovať back-test na dátach za posledný 1 rok a parametre použiť pri obchodovaní nasledujúci mesiac. Po mesiaci obchodovania by sme znovu určili parametre optimalizáciou 1 ročného obdobia obsahujúceho aj posledný obchodovaný mesiac. Týmto spôsobom vieme pokračovať a mať stále „aktuálne“ parametre.



Obrázok 3.2: Výstup z back-testu z balíčka qstrader. Na prvom obrázku máme kumulatívne výnosy z back-testu danej stratégie a výnosy indexu S&P 500 pre porovnanie. Na ďalšom grafe máme zobrazené prepady portfólia (rozdiel od dosiahnutého maximálneho výnosu), ktoré sú dobrým meradlom rizikovosti danej stratégie. Nakoniec máme mesačné a ročné výnosy v % a tabuľky s ďalšími zaujímavými štatistikami.

Záver

Hlavnou úlohou tejto práce bolo predstavenie pomerne novej metódy na optimalizáciu neznámych a/alebo výpočtovo náročných funkcií – bayesovskej optimalizácie (BO). V teoretickej časti sme vychádzali hlavne z niekoľkých článkov o bayesovskej optimalizácii a vytvorili sme ucelený prehľad BO s rôznymi rozšíreniami. V praktickej časti sme vytvorili vlastný algoritmus BO, pričom sme sa inšpirovali už existujúcimi prístupmi a aplikovali ho na reálne problémy.

V prvej kapitole sme odvodili regresiu podľa gaussovského procesu (GP) a pomocou nej vytvorili pravdepodobnostný model pre účelovú funkciu. Ďalej sme zostrojili výpočtovo nenáročnú diferencovateľnú akvizičnú funkciu, ktorej maximalizáciou sme získali nový bod pre ďalší výpočet účelovej funkcie. Predstavili sme niekoľko typov akvizičnej funkcie, ktoré sme zostrojili tak aby zohľadňovali informáciu z predchádzajúcich výpočtov účelovej funkcie. Ukázali sme spôsoby určenia hyper-parametrov, ktoré sú potrebné pre kovariančné funkcie pri GP a tiež pre akvizičné funkcie.

Získané vedomosti sme aplikovali pri tvorbe vlastného algoritmu bayesovskej optimalizácie, ktorý sme následne naprogramovali v jazyku Python™ (zdrojový kód je textovou súčasťou práce). Pri programovaní sme sa stretli s problémom optimalizácie multimodálnych a prevažne plochých viacrozmerných funkcií. Zvolili sme jednu z kvázi-Newtonových metód s viac reštartmi v náhodne zvolených bodoch. Ďalším vylepšením numerickej optimalizácie akvizičnej funkcie by sme mohli dosiahnuť rýchlejšiu konvergenciu algoritmu k optimu. Taktiež sme zvolili vlastný prístup výberu hyper-parametrov a to konkrétne generovaním z rovnomerného rozdelenia s postupnou zmenou intervalu. Aplikáciou algoritmu na jednoduchých príkladoch sme aspoň prakticky ukázali konvergenciu k optimu.

Poslednú časť prvej kapitoly sme venovali ďalším rozšíreniam BO – konkrétne paralelnej optimalizácii viacerých úloh, prípadu kategoriálnych parametrov účelovej funkcie a iným prístupom k apriórnemu rozdeleniu účelovej funkcie.

Druhá kapitola obsahuje nadstavbu BO pre vysoké dimenzie. Pri vyšších dimenziách sme museli zvoliť špeciálny prístup pretože numerická optimalizácia aj výpočtovo nenáročných funkcií nie je jednoduchá. Predstavili sme metódu REMBO (bayesovská optimalizácia s náhodným preložením), pri ktorej využívame predpoklad existencie efektívnych dimenzií, t.j. nie všetky dimenzie majú vplyv na hodnotu účelovej funkcie. Na zníženie dimenzie sme použili jednoduchú lineárnu projekciu. Opísali sme výsledky práce autorov tejto metódy a metódu aplikovali na jednoduchom viacrozmernom príklade. V krátkosti sme predstavili aj iný prístup k zníženiu dimenzionality (HD BO) a tiež aditívnu BO, ktorá využíva predpoklad separovateľnosti účelovej funkcie.

Metódy opísané v druhej kapitole majú veľmi striktné predpoklady. Ukázalo sa, že REMBO dosahuje veľmi dobré výsledky aj pri porušení základného predpokladu. Pre mnohé praktické problémy už existujú vhodné optimalizačné metódy, ktoré vznikli priamo pre potrebu riešenia daného problému. Takýmto problémom nedokážeme konkurovať, prípadne nedokážeme dosiahnuť výrazne lepšie výsledky. Ak však potrebujeme optimalizovať neznámu a/alebo výpočtovo náročnú funkciu, pre ktorú neexistuje špeciálna optimalizačná metóda, tak bayesovskou optimalizáciou môžeme dosiahnuť výrazne lepšie výsledky ako pri iných prístupoch.

Tretia kapitola obsahuje aplikácie vlastného algoritmu BO na praktické príklady. Prvý príklad je optimalizácia parametrov klasifikačnej metódy strojového učenia (metódy podporných vektorov – SVM) pri klasifikácii obrázkov rukou písaných číslic. Po aplikácii BO sme získali hodnoty blízke autormi odporúčaným hodnotám. Druhou aplikáciou bola obchodná stratégia na burze, konkrétne párové obchodovanie využívajúce Kalmanov filter. Úlohou bolo nájsť najvhodnejšie parametre rozptylu pre Kalmanov filter pri maximalizácii výnosu a minimalizácii rizika na back-teste. Podarilo sa nám nájsť výrazne lepšie výsledky ako autorovi danej stratégie – na období 2010-2016 sme aj po započítaní transakčných nákladov dosiahli vyšší výnos ako index S&P 500. Z optimalizačného hľadiska sme sa na stratégiu dívali ako na neznámu funkciu, ktorá bola pravdepodobne po častiach konštantná s vysokým počtom bodov nespojitosti.

Po teoretickej aj praktickej stránke sa BO javí ako veľmi dobrá optimalizačná metóda. Pri použití BO v praxi však treba prihliadať na účelovú funkciu. Ak je účelová funkcia známa a existujú vhodnejšie optimalizačné metódy tak najlepším riešením je ich použiť. Napríklad optimalizácia váh v neurónových sieťach sa môže javiť ako vhodný príklad na aplikáciu REMBO, keďže nie všetky váhy sú dôležité. Použitie klasických metód ako stochastické klesajúce gradienty však rýchlejšie dosiahne lepšie výsledky. BO by sa v tomto prípade mohla použiť na optimalizáciu ďalších parametrov neurónovej siete (namiesto mriežkového vyhľadávania [Grid search]). BO je vhodná pre neznámu účelovú funkciu, prípadne ak neexistuje žiadna špeciálna optimalizačná metóda a vhodným riešením sa zdá jedine použitie náhodného prehľadávania.

Zoznam použitej literatúry

Brochu, E., Cora, V. M. a de Freitas, N. (2010). A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *arXiv:1012.2599v1*.

Chen, B., Castro, R. M. a Krase, A. (2012). Joint optimization and variable selection of high-dimensional gaussian processes. In *Proceedings of the 29th International Conference on Machine Learning*.

Cipra, T. (2013). *Finančnú ekonometrie*. Ekopres s.r.o., 2. upravené vydanie edition. ISBN 978-80-86929-93-4.

GPy (2012–). GPy: A gaussian process framework in python. `http://github.com/SheffieldML/GPy`.

Halls-Moore, M. L. (2016). *Advanced Algoritmic Trading*. URL `https://www.quantstart.com`.

Hutter, F., Hoos, H. H. a Leyton-Brown, K. (2011). Sequential model-based optimization for general algorithm configuration. *Learning and Intelligent Optimization*, pages 507–523.

Jones, E., Oliphant, T., Peterson, P. a kol. (2001–). SciPy: Open source scientific tools for Python. URL `http://www.scipy.org/`.

Kandasamy, K., Schneider, J. a Poczos, B. (2015). High dimensional bayesian optimisation and bandits via additive models. In *Proceedings of The 32nd International Conference on Machine Learning 2015*, volume 37, Lille, France, 2015.

Li, C., Rana, S., Gupta, S., Nguyen, V. a Venkatesh, S. (2016). High dimensional bayesian optimization with elastic gaussian process. In *30th Conference on Neural Information Processing Systems (NIPS)*.

ML4AAD. Machine learning for automated algorithm design; RoBO and SMAC. URL `http://www.ml4aad.org`.

Murphy, K. P. (2012). *Machine Learning: A Probabilistic Perspective*. The MIT Press, Cambridge, Massachusetts. ISBN 978-0-262-01802-9.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. a Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, **12**, 2825–2830.

Shahriari, B., Swersky, K., Wang, Z., Adams, R. a de Freitas, N. (2016). Taking human out of the loop: A review of bayesian oprimization. *Proceedings of IEEE*, **104**, 148–175.

Snoek, J., Larochelle, H. a Adams, R. P. (2012). Practical bayesian optimization of machine learning algorithms. *arXiv:1206.2944*.

Springenberg, J. T., Klein, A., Falkner, S. a Hutter, F. (2016). Bayesian optimization with robust bayesian neural networks. In *29th Conference on Neural Information Processing Systems (NIPS)*.

Swersky, K., Snoek, J. a Adams, R. P. (2013). Multi-task bayesian optimization. In *26th Conference on Neural Information Processing Systems (NIPS)*.

Villemonteix, J., Vazquez, E. a Walter, E. (2009). An informational approach to the global optimization of expensive-to-evaluate functions. *J Glob Optim*, **43**, 373–389.

Wales, D. J. a Doye, J. P. K. (1997). Global optimization by basin-hopping and the lowest energy structures of lennard-jones clusters containing up to 110 atoms. *Journal of Physical Chemistry A*, **101**, 5111–5116.

Wang, Z. a de Freitas, N. (2014). Theoretical analysis of bayesian optimisation with unknown gaussian process hyper-parameters. *NIPS Workshop on Bayesian Optimization*.

Wang, Z., Shakibi, B., Jin, L. a de Freitas, N. (2014). Bayesian multi-scale optimistic optimization. *AI and Statistics*, pages 1005–1014.

Wang, Z., Hutter, F., Zoghi, M., Matheson, D. a de Freitas, N. (2016). Bayesian optimization in a billion dimensions via random embeddings. *arXiv:1301.1942v2*.

Prílohy

Zdrojový kód k algoritmu BO a REMBO v **Python**™

```
import numpy as np
from scipy.stats import norm
from GPy.kern import Exponential
from GPy.models import GPRegression
from scipy.optimize import minimize, basinhopping
from time import time

def expected_improvement(x, xi, fmax, GPreg, dim):
    pred = GPreg.predict(x.reshape(-1,dim))
    mean, std = pred[0].T[0], np.sqrt(pred[1].T[0])
    z = (mean - fmax - xi)/std
    ei = -std*(z*norm.cdf(z) + norm.pdf(z))
    ei[std == 0.0] = 0.0
    return ei

def upper_confidence_bound(x, beta, GPreg, dim):
    pred = GPreg.predict(x.reshape(-1,dim))
    mean, std = pred[0].T[0], np.sqrt(pred[1].T[0])
    return - mean - beta*std

class BO(object):
    def __init__(self, objective, bounds, n_iterations, n_init = 2,
                 rand = None, acquisition_func = "EI", hyper_par =
                 1, scale = [1e-5,100], noise_var = 0.001, t_sigma
                 = 0.001, log_info = True, n_iters_aqui = 15,
                 use_bashinhopping = False):
        """
        Peter Kostovcik: Bayesovska Optimalizacia; diplomova praca
            2017
        Karlova Univerzita, Matematicko-fyzikalni fakulta
        e-mail: p.kostovcik@gmail.com
        ======== INPUT ========
        objective:          ucelova funkcia
        bounds:             krabicove ohranicenia (list of tuples)
        n_iterations:       pocet iteracii/vypoctov ucelovej funkcie
        n_init:             pocet starovacich bodov (default = 2)
        acquisition_func:   typ akvizicnej funkcie (UCB, default =
            EI)
        rand:               np.RandomState(cislo) (default = None,
            random choice)
        hyper_par:          hyper_par pre GP (okrem mierky
            lengthscale)
                            (default var = 1 pre SE kernel)
        scale:              interval pre mierku (default = [1e
            -5,100])
        noise_var:          rozptyl pre sum sigma_y (default =
            0.001)
        t_sigma:            hranica pre zmenu scale (default = 0.001
             == noise_var)
        log_info:           info o vypocte ucelovej funkcie (default
             = True)
```

```
        n_iters_aqui:       pocet start bodov pre L-BFGS-B opti
            akvizicnej funkcie
        use_bashinhopping   minimalizacia akvizicnej funkcie cez
            bashinhopping (default = False)
        ===============
        daju sa doplnit dalsie typy kernel funkcii a pod.
        """
        self.objective, self.bounds = objective, bounds
        self.n_iterations, self.n_init = n_iterations, n_init
        self.acquisition_func = acquisition_func == "EI"
        self.hyper_par, self.scale, self.noise_var = hyper_par,
            scale, noise_var
        self.bashop = use_bashinhopping
        if rand == None:
            self.rand = np.random.RandomState(np.random.randint(0,
                10000))
        else:
            self.rand = rand
        self.t_sigma, self.log_info = t_sigma, log_info
        self.dim, self.n_iters_aqui = len(self.bounds), n_iters_aqui
        self.run()

    def run(self):
        self.start()
        self.maximize(self.n_iterations-self.n_init)
        self.result()

    def start(self):
        self.start_time = time()
        x_init = self.rand.uniform(low = np.array(self.bounds).T[0],
                                   high = np.array(self.bounds).T
                                       [1], size=(2,self.dim))
        y_init = np.array([self.objective(x_init[0]),self.objective(
            x_init[1])])
        self.x, self.y = [x_init[0], x_init[1]], [y_init[0],y_init
            [1]]
        if self.log_info:
            print("%s step in BO; objective value: %s at %s; time: %
                s s." % (1, self.y[0] ,self.x[0], time()-self.
                start_time))
            print("%s step in BO; objective value: %s at %s; time: %
                s s." % (2, self.y[1] ,self.x[1], time()-self.
                start_time))
        self.aquis_par = [np.abs(y_init[0]-y_init[1])/2 if self.
            acquisition_func else 3]
        self.GP_lengthscale = [self.rand.uniform(self.scale[0], self
            .scale[1])]
        self.a, self.C = 2, 0
        self.lowering_scale = 0

    def sample_xi(self,a):
        y = np.sort(self.y)
        return self.rand.uniform(0, 1/a * (y[-1] - y[0]))
    def sample_beta(self,a):
        return self.rand.uniform(0, 6/a)

    def maximize(self, iterations):
        scales = self.scale.copy()
```

```
        for n in np.arange(iterations):
            kernel = Exponential(input_dim = self.dim, variance =
                self.hyper_par, lengthscale = self.GP_lengthscale
                [-1])
            X = np.array(self.x).reshape(-1,1) if self.dim == 1 else
                np.array(self.x)
            Y = np.array(self.y).reshape(-1,1)
            GPreg = GPRegression(X, Y, kernel = kernel, noise_var =
                self.noise_var)
            GPreg.optimize()
            # minimalizacia zapornej akvizicnej funkcie
            points = np.array([np.array(self.bounds).T[0]] + self.x
                + [np.array(self.bounds).T[1]])
            points.sort(axis=0)
            vals, par = [], []
            x0 = self.rand.uniform(np.array(self.bounds).T[0],np.
                array(self.bounds).T[1], size=(self.n_iters_aqui,self
                .dim))
            for x in x0:
                if self.acquisition_func:
                    if self.bashop:
                        opti = basinhopping(expected_improvement, x0
                            = x, minimizer_kwargs={ "method" : "L-
                            BFGS-B", "bounds":self.bounds, "args" : (
                            self.aquis_par[-1], np.max(self.y), GPreg
                            , self.dim,)})
                    else:
                        opti = minimize(expected_improvement, x0 = x
                            , method = "L-BFGS-B", args = (self.
                            aquis_par[-1], np.max(self.y), GPreg,
                            self.dim,), bounds = self.bounds)
                else:
                    if self.bashop:
                        opti = basinhopping(upper_confidence_bound,
                            x0 = x, minimizer_kwargs={ "method" : "L-
                            BFGS-B", "bounds":self.bounds, "args" : (
                            self.aquis_par[-1], GPreg, self.dim,)})
                    else:
                        opti = minimize(upper_confidence_bound, x0=x
                            , method = "L-BFGS-B", args = (self.
                            aquis_par[-1], GPreg, self.dim,),bounds =
                             self.bounds)
                par.append(opti.x)
                vals.append(opti.fun)
            vals, par = np.array(vals), np.array(par)
            self.x.append(par[vals.argmin()])
            self.y.append(self.objective(self.x[-1]))
            if self.log_info:
                print("%s step in BO; objective value: %s at %s;
                    time: %s s." % (len(self.x), self.y[-1] ,self.x
                    [-1], time()-self.start_time))
            # volba novych hyper-parametrov
            if n%10 == 0:
                self.a += 1
            if GPreg.predict(self.x[-1].reshape(-1,self.dim))[1][0]
                < self.t_sigma:
                self.C += 1
            if self.C == 5:
```

```
                scales[1] = np.max([scales[0], 0.8*scales[1]])
                self.C = 0
                self.lowering_scale += 1
            self.GP_lengthscale.append(self.rand.uniform(scales[0],
                scales[1]))
            self.aquis_par.append(self.sample_xi(self.a) if self.
                acquisition_func else self.sample_beta(self.a))
        self.end_time = time()
    def result(self):
        self.optimum = np.max(self.y)
        self.optimal_x = self.x[np.argmax(self.y)]
        self.computation_time = self.end_time-self.start_time
        print("Found max value %s at point %s. Computation time %s"
            % (self.optimum, self.optimal_x, self.computation_time))

def REMBO(f, D, d, n_iterations, n_init = 2, rand = None,
    acquisition_func = "EI", hyper_par = 1, scale = [1e-5,100],
    noise_var = 0.001, t_sigma = 0.001, log_info = True, n_iters_aqui
    = 15):
    """
    f           ucelova funkcia
    D           dimenzia f
    d > 1       (d >= d_e) znizena dimenzia
    ostatne vstupy vid BO
    """
    if rand == None:
        rand = np.random.RandomState(np.random.randint(0, 10000))
    A = rand.normal(size=(D,d))

    def rho(z):
        x = A.dot(z)
        x[x<-1] = -1
        x[x>1] = 1
        return x

    def objective_reduced(z):
        return f(rho(z))

    bounds = d*[(-np.sqrt(d),np.sqrt(d))] if d > 2 else d*[(-np.sqrt
        (d)/np.log(d), np.sqrt(d)/np.log(d))]

    return [BO(objective_reduced, bounds, n_iterations, n_init, rand
        , acquisition_func, hyper_par, scale, noise_var, t_sigma,
        log_info, n_iters_aqui), A]
```