RYCHLOST ADAPTIVNÍCH ALGORITMŮ PRO DETEKCE NOVOSTI

Matouš Cejnek, Adam Peichl

Ústav přístrojové a řídicí techniky, Fakulta strojní, ČVUT v Praze, adam.peichl@fs.cvut.cz

Abstrakt: Tento článek se zabývá analýzou a porovnáním rychlosti několika adaptivní algoritmů (ELBND, LE, MD, FD). Rychlost je klíčová vlastnost algoritmů pro detekci novosti, které jsou používány pro zpracování dat v reálném čase. Adaptivní detekce novosti je pro procesy měřené v reálném čase zajímavá speciálně díky své robustnosti proti vysoké ne-stacionaritě měřených dat. Zkoumané algoritmy jsou v tomto článku analyzovány teoreticky pomocí prostředků asymptotické složitosti a získané závěry jsou validovány experimentálně na testovacích datech. Získané výsledky ukazují, že rozdíly v časové náročnosti jednotlivých algoritmů nejsou zanedbatelné. Zatímco ELBDN a LE mají lineární časovou složitost s malými multiplikativními a aditivními konstantami, MD vykazuje vlastnosti časové složitosti kvadratické a FD má linární časovou složitost s podstatně vyššími multiplikativními a aditivními konstantami než ELBDN a LE.
Klíčová slova: detekce novosti, asymptotická složitost, adaptivní algoritmy

Abstract: This paper deals with the analysis and comparison of speeds of several adaptive algorithms (ELBND, LE, MD, FD). Novelty detection algorithms are used for real-time data processing, thus speed is a key atribute. Adaptive novelty detection is especially interesting for real-time measured processes due to its robustness against high non-stationarity of measured data. The studied algorithms are analyzed theoretically through time complexity and results are validated experimentaly on testing data. Obtained results shows, that distinctions in time complexity between algorithms are not negligible. While ELBDN and LE have linear time complexity with small multiplicative and additive constants, MD has quadratic time complexity and FD has linear time complexity with substantially higher multiplicative and additive constants than ELBDN and LE.
Keywords: Novelty detection, time complexity, adaptive algorithms

1 Úvod

Detekce novosti (*novelty detection*) je téma z oblasti strojového učení (*machine learning*), které se zabývá detekci neočekávaných stavů. Tyto stavy můžou představovat různé anomálie, které představují nestandardní chování sledovaného procesu. Takové chování je potřeba často detekovat, aby mohl být spuštěn krizový scénář reagující vhodným způsobem na daný neočekávaný stav. Toto téma je dnes obzvláště zajímavé, díky vysokému nárůstu procesů, jenž je potřeba monitorovat v reálném čase. Zdrojem těchto procesů jsou často moderní paradigmata jako Internet of Things (IoT) a Industry 4.0. Tyto nové zdroje monitorovaných procesů jsou příležitosti pro uplatnění strojového učení a přinášejí řadu nových výzev, jenž ještě nebyly spolehlivě vyřešeny. Jedna z těchto výzev, jenž online procesy přináší, je nutnost vysoké rychlosti při udržení dostatečné robustnosti proti různým variantám šumu v datech. Nejčasnější oblasti kde jsou tyto algoritmy nasazeny jsou: sledování průmyslových procesů, monitoring lékařských signálů, detekce událostí pro automatické obchodování a detekce webových útočníků. S ohledem na charakteristiku těchto oblastí, je evidentní že ve spoustě případů je potřeba, aby algoritmy pracovaly dostatečně rychle - stovky až tisíce vzorků za sekundu.

Velké množství doposud publikovaných algoritmů určených pro detekci novosti nesplňuje alespoň jednu ze základních podmínek zpracování dat v reálném čase - nejsou dostatečně rychlé, nebo nejsou dostatečně robustní. Algoritmy jenž jsou studovány v této práci mají zajištěnou jistou míru robustnosti díky své adaptivní podstatě. Adaptace je totiž intuitivní způsob jak kompenzovat offset nebo jinou postupně vznikající nerovnováhu v datech [1, 2, 3]. Jejich rychlost ale zatím nebyla studována detailněji. Z těchto důvodů jsou výsledky této práce přínosné pro budoucí uživatele těchto algoritmů. Výpočetní rychlost algoritmu strojového učení (detekce novosti v tomto

případě) záleží na množství faktorů. Některé faktory je možné ovlivnit a jiné ne. Obvykle ovlivnitelné faktory rychlosti procesu jsou hardware a software. I když i zde jsou často limitace dány možnostmi daného pracoviště a nebo místa nasazení daného procesu. Vcelku neovlivnitelný faktor daného procesu je složitost použitého algoritmu ve své podstatě, respektive jeho optimalizovaná podoba.

V tomto článku se pokoušíme testované algoritmy porovnat jak analyticky - za použití teoretických konceptů z oblasti asymptotické složitosti [4], tak experimentálně pomocí testovacího frameworku, který jsme zhotovili pro daný účel. Získané výsledky o implementační složitosti a chování adaptivní algoritmů jsou cenné poznatky užitečné pro každého, kdo by chtěl zkoumané algoritmy použít v libovolné reálné aplikaci.

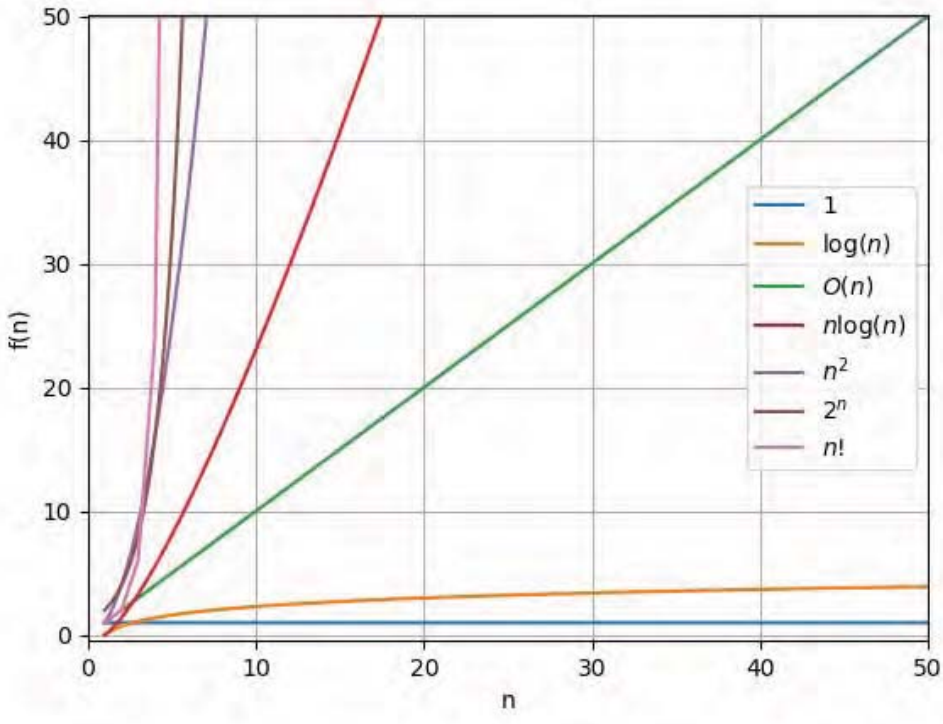
2 Metody

V této sekci jsou představeny postupy a metody, jenž jsou použity v tomto článku. V podsekci 2.1 je představen koncept, který je použit k analytickému porovnání testovaných algoritmů. V podsekci 2.3 jsou představy zkoumané metody adaptivní detekce novosti, jmenovitě jejich algoritmy včetně jejich složitostí. Následně v podsekci 2.2 jazyk Python, v kterém jsou implementovány všechny experimenty provedené v tomto článku.

2.1 Asymptotická složitost

Asymptotická složitost [4] je způsob dělení algoritmů podle operační náročnost algoritmu. Asymptotická složitost algoritmu vypovídá o tom, jakým způsobem se bude chovat algoritmus v závislosti na změně rozsahu vstupních dat. Zapisuje se pomocí Landauovy notace (Omikron notace) jako $O(f(n))$, kde n je počet vstupních dat. Asymptotická složitost je funkce $f : \mathbb{N} \longrightarrow \mathbb{N}$, která vyjadřuje vztah mezi velikostní vstupních dat a množstvím spotřebovaného výpočetního času. V našem případě předpokládáme, že každá elementární aritmetická operace (násobení, sčítání, . . .) spotřebuje předem dané (konstantní) množství času.

V našem případě budeme používat horní odhad složitosti $O(f(n))$, což je nejhorší možná složitost[1]. Bude tedy platit, že náš algoritmus proběhne asymptoticky rychleji nebo při nejhorším stejně rychle, jako náš odhad.



Obr. 1: Porovnání vybraných tříd asymptotických složitostí. Vysvětlení k legendě je možné najít v Tab. 1

[1]Složitost pro nejhorší možný vstup.

Tab. 1: Vybrané třídy složitosti a jejich typické příklady

O-notace složitosti	název	příklad
$O(1)$	konstantní	výběr prvku z pole
$O(\log n)$	logaritmická	hledání v seřazeném poli
$O(n)$	lineární	hledání minima, maxima
$O(n \log n)$	lineárně logaritmická	quicksort
$O(n^2)$	kvadratická	bubblesort
$O(2^n)$	exponenciální	řešení problému obchodního cestujícího
$O(n!)$	faktoriální	brute force řešení problému obchodního cestujícího

V této sekci jsou vysvětleny metody použité v této studii, včetně postupu křížové validace, která byla použita pro vyhodnocení výsledků této studie.

2.2 Jazyk Python

Python je vysokoúrovňový skriptovací programovací jazyk, který v roce 1991 navrhl Guido van Rossum [5]. Nejrozšířenější implementace jazyka Python je v jazyce C (často označováno jako CPython). Z tohoto důvodu je výkon Pythonu silně spjatý s výkonem jazyka C. Pro vyšší optimalizaci výpočetně náročných úkolů (operace s maticemi atp.) jsou pro Python dostupné knihovny, které umožňují tyto operace provádět přímo v jazyce C tak, že Python figuruje pouze jako klient. Pro implementaci algoritmů v tomto článku byla použita knihovna Numpy [6], která je považována za standard pro výpočetní operace v Pythonu. Několik příkladů operací s polem v Pythonu (Python list) včetně jejich asymptotické složitosti je v Tab. 2.

Tab. 2: Asymptotické složitosti pro datový typ list v jazyce Python

Operace	Příklad	Třída asymptotické složitosti
index	pole[i]	$O(1)$
přiřazení	pole[i] = None	$O(1)$
délka	len(pole)	$O(1)$
append	pole.append(False)	$O(1)$
pop	pole.pop()	$O(1)$
clear	pole.clear()	$O(1)$
řez	pole[a:b]	$O(b-a)$
extend	pole.extend(l)	$O(len(l))$
construction	list(l)	$O(len(l))$

2.3 Metody detekce novosti

2.3.1 Error and learning based novelty detection (ELBND)

Metoda ELBND detekuje novost základě okamžitého přírůstku adaptivních vah a chyby podle následujícího pravidla

$$\mathrm{ELBND}(k) = \Delta \mathbf{w}(k) e(k). \tag{1}$$

Všimněte si, že v rovnici 1 není k výpočtu použity žádné předchozí hodnoty, ale pouze okamžité hodnoty. Výstup 1 je vektor, který popisuje míru novosti pro každou adaptivní váhu zvlášť v daném diskrétním čase k. Tento vektor lze dále redukovat na jednu hodnotu, pro jednodušší vyhodnocení novosti v daném vzorku. Běžný způsob jak získat z vektoru $\mathrm{ELBND}(k)$ skalár $\mathrm{elbnd}(k)$ je funkce maxima z absolutních hodnot

$$\mathrm{elbnd}(k) = \max |\mathrm{ELBND}(k)|. \tag{2}$$

Všimněte si, že tento způsob výpočtu novosti nepotřebuje žádné další parametry, takže problém s optimalizací metody je minimalizován. Na druhou stranu je nutné podotknout, že výstup metody je silně závislý na adaptivním algoritmu. Touto problematikou se více zabývala například práce [7]. Tento algoritmus je zde posuzován na základě jeho implementace v knihovně Padasip [8]

Asymptotická složitost tohoto algoritmu je představena v tabulce 3. Jak je z tabulky patrné, žádný krok nemá vyšší asymptotickou složitost než $O(n)$, můžeme tedy prohlásit, že algoritmus má *lineární* asymptotickou složitost.

Tab. 3: Časová náročnost a počet operací pro jeden krok algoritmu ELBND (n je počet adaptivních parametrů).

pořadí	operace	složitost	sčítání	násobení	poznámka
1.	$o_1 = \mathbf{\Delta w(k)}e$	$O(n)$	0	n	-
2.	$o_2 = \lvert o_1 \rvert$	$O(n)$	0	0	abs()
3.	$\max(o_2)$	$O(n)$	0	0	max()

2.3.2 Approximate Individual Sample Learning Entropy (AISLE)

Approximate Individual Sample Learning Entropy [9] je často označována jen jako Learning Entropy. Výpočet AISLE pro každý vzorek je dán následujícím pravidlem:

$$\text{AISLE}(k) = \frac{1}{n \cdot n_\alpha} \sum f(\Delta w_i(k), \alpha) \; ; \; \forall \alpha \in \boldsymbol{\alpha} \, , \tag{3}$$

kde n je počet adaptivních vah a n_α je počet citlivostí detekce, které si volí uživatel

$$\boldsymbol{\alpha} = [\alpha_1, \alpha_2, \ldots, \alpha_{n_\alpha}] \; ; \; \alpha_1 < \alpha_2 < \ldots < \alpha_{n_\alpha} . \tag{4}$$

Funkce $f(\Delta w_i(k), \alpha)$ je definována jako

$$f(\Delta w_i(k), \alpha) = \begin{cases} 1, & \text{if } |\Delta w_i(k)| > \alpha \cdot \overline{|\Delta w_{Mi}(k)|} \\ 0, & \text{v ostatních případech} \end{cases} \tag{5}$$

kde $\overline{|\Delta w_{Mi}(k)|}$ je střední hodnota okna použitého pro výpočet AISLE. Tato velikost okna m by měla být zvolena s ohledem na možnou periodicitu v datech [9]. Další parametr volený uživatelem je počet citlivostí pro detekci α a jejich hodnoty. Tento algoritmus je zde posuzován na základě jeho implementace v knihovně Padasip [8].

Asymptotická složitost tohoto algoritmu je představena v tabulce 4. Jak je z tabulky patrné, žádný krok nemá vyšší asymptotickou složitost než $O(n, n_\alpha)$, můžeme tedy prohlásit, že algoritmus má *lineární* asymptotickou složitost.

Tab. 4: Časová náročnost a počet operací pro jeden krok algoritmu ELBND (n je počet adaptivních parametrů a n_α je počet citlivostí detekce α).

pořadí	operace	složitost	sčítání	násobení	poznámka
1.	$o_1 = \lvert \mathbf{\Delta W_M(k)} \rvert$	$O(n)$	0	0	abs()
2.	$o_2 = \bar{o_1}$	$O(n, m)$	$m \cdot n$	n	-
3.	$o_3 = \lvert \mathbf{\Delta w_i(k)} \rvert$	$O(n)$	0	0	abs()
4.	$o_4 = \alpha \cdot o_2$	$O(n, n_\alpha)$	0	$n \cdot n_\alpha$	-
5.	$o_5 = (o_3 > o_4)$	$O(n, n_\alpha)$	0	0	-
6.	$o_6 = \sum o_5$	$O(n, n_\alpha)$	n_α	$(n_\alpha \cdot n) - 1$	-
7.	$o_6 / (n \cdot n_\alpha)$	$O(2)$	0	2	-

2.3.3 Mahanobilis Distance (MD)

Tato metoda využívá k dektekci novosti Mahanobilisovu vzdálenost[10], což je vzdálenost, která bere v úvahu závislosti mezi jednotlivými veličinami[2].
Nechť je naše pozorování adaptivních parametrů ve vzorku k:

$$\mathbf{\Delta w(k)} = [\Delta w_0(k), \Delta w_1(k), \ldots, \Delta w_n(k)] \tag{6}$$

A množina předchozích pozorování $\mathbf{\Omega(k)}$:

$$\mathbf{\Omega(k)} = \begin{bmatrix} \mathbf{\Delta w(k-1)} \\ \mathbf{\Delta w(k-2)} \\ \mathbf{\Delta w(k-3)} \\ \vdots \\ \mathbf{\Delta w(k-M)} \end{bmatrix} \tag{7}$$

[2]Kovarianční matici.

Dále nechť $\mathbf{S}$ je kovarianční matice $\mathbf{\Omega(k)}$:

$$\mathbf{S} = \mathbf{\Omega^T(k)\Omega(k)} \tag{8}$$

A $\overline{\mathbf{\Delta w_M(k)}}$ je těžiště $\mathbf{\Omega(k)}$, tedy vektor průměrných hodnot každého sloupce $\mathbf{\Omega(k)}$:

$$\overline{\mathbf{\Delta w_M(k)}} = \frac{\sum_{i=0}^{M-1} \mathbf{\Omega_i(k)}}{M} = \frac{\sum_{j=1}^{M} \mathbf{\Delta w(k-j)}}{M} \tag{9}$$

Potom D_M je mahanobilisova vzdálenost $\mathbf{w(k)}$ a $\mathbf{\Omega(k)}$:

$$D_M\left(\mathbf{\Delta w(k)}, \mathbf{\Omega(k)}\right) = \sqrt{\left(\mathbf{\Delta w(k)} - \overline{\mathbf{\Delta w(k)}}_\mathbf{M}\right) \cdot \mathbf{S^{-1}} \cdot \left(\mathbf{\Delta w(k)^T} - \overline{\mathbf{\Delta w(k)}}_\mathbf{M}^\mathbf{T}\right)} \tag{10}$$

Zjednodušený kód v jazyce python[3]:

```
import numpy as np

def mahanobilis_distance(omega,dw):
    S = np.cov(omega,rowvar=false) # ekvivalentni np.dot(omega.T,omega)
    S_1 = np.linalg.inv(S) # inversni matice, v realnem kodu se obcas pouziva Moore-Penrose
                                                    inverse
    mu = np.mean(omega,axis=0) # teziste omega
    mu[:np.newaxis] # doplneni rozmeru vektoru teziste z (n,) na (n,1)
    mahanobilis = np.sqrt(np.abs(np.matmul(np.matmul(dw.T-mu.T,S_1),(dw-mu))))
    return mahanobilis
```

Asymptotická složitost tohoto algoritmu je představena v tabulce 5. Z pohledu asymptotické složitosti jsou zajímavé řádky 2 a 3, algoritmus má v každém případě *polynominální* složitost a vzhledem k tomu, že s nejvyšší pravděpodobností bude platit $M > n$, bude výsledná asymptotická složitost $O(Mn^2)$.

Tab. 5: Časová náročnost a počet operací pro jeden krok algoritmu MD(n je počet adaptivních parametrů, M je počet řádků matice Ω)

pořadí	operace	složitost	sčítání	násobení	poznámka
1.	$o_1 = \overline{\mathbf{\Delta w_M(k)}}$	$O(Mn)$	$(M-1)n$	n	těžiště
2.	$o_2 = \mathbf{\Omega^T\Omega}$	$O(Mn^2)$	$(M-1)n$	Mn^2	kovarianční matice
3.	$o_3 = (o_2)^{-1}$	$O(n^3)$	-	-	inverzní matice
4.	$o_4 = \mathbf{\Delta w(k)} - o_1$	$O(n)$	n	-	-
5.	$o_5 = o_4 o_3$	$O(n^2)$	$(n-1)n$	n^2	-
6.	$o_6 = (o_4)^T$	$O(1)$	-	-	transpozice vektoru
7.	$o_7 = o_5 o_6$	$O(n^2)$	$(n-1)n$	n^2	-
8.	$D_M = \sqrt{o_7}$	$O(\sqrt{n})$	-	-	odmocnina

2.3.4 Fuzzy Density (FD)

Tato metoda detekuje novost na základě odhadu funkce hustoty pravděpodobnosti pomocí fuzzy množin[11] podle následujícího předpisu:

Nechť je naše pozorování adaptivních parametrů ve vzorku k:

$$\mathbf{\Delta w(k)} = [\Delta w_0(k), \Delta w_1(k), \ldots, \Delta w_n(k)] \tag{11}$$

A množina předchozích pozorování $\mathbf{\Omega(k)}$:

$$\mathbf{\Omega(k)} = \begin{bmatrix} \mathbf{\Delta w(k-1)} \\ \mathbf{\Delta w(k-2)} \\ \mathbf{\Delta w(k-3)} \\ \vdots \\ \mathbf{\Delta w(k-M)} \end{bmatrix} \tag{12}$$

Potom fuzzy hustota $\Phi_\mathbf{\Omega}(\mathbf{X})$:

$$\Phi_\mathbf{\Omega}(\mathbf{X}) = \frac{1}{M} \sum_{\forall \Omega_i} \prod_{\forall x_j \in \mathbf{X}} \mu_G(x_j, \Omega_{i,j}, \sigma_j) \tag{13}$$

[3]Byly odstraněny některé operace, které nejsou uvedeny v popisu algoritmu výšše: např. transformace 1D vektoru o délce n ja matici o rozměrech (n,1)

Kde funkce $\mu_G(x_j, \Omega_{i,j}, \sigma_j)$ je gaussovská funkce příslušnosti:

$$\mu_G(x_j, \Omega_{i,j}, \sigma_j) = e^{-\left(\frac{x_j - \Omega_{i,j}}{\sigma_j}\right)^2} \tag{14}$$

Přičemž platí, že σ_j je *average distance* j-tého sloupce Ω:

$$\sigma_j = f_{AD} = \frac{1}{M} \sum_{k=0}^{M-1} \sum_{l=l+1}^{M-1} |\Omega_{k,j} - \Omega_{l,j}| \tag{15}$$

Zjednodušený kód v jazyce python:

```
import numpy as np

def average_norm(omega):
    n = np.shape(omega)[1]
    sigma = []
    for i in range(n):
        X, Y = np.meshgrid(omega[:,i],omega[:,i])
        norm = np.sum(np.abs(X-Y))/(2*n*(n-1))
        sigma.append(norm)
    return sigma


def gaussian(x,center,sigma):
    return np.exp(-((x-center)/sigma)**2)

def fuzzy_density(omega,dw,M):
    sigma = average_norm(omega)
    value = 0
    for row in omega:
        p = 1
        for k in range(len(row)):
            p = p*gaussian(dw[k],row[k],sigma[k])
        value += p
    return value/M
```

Tab. 6: Časová náročnost a počet operací pro jeden krok algoritmu FD(n je počet adaptivních parametrů, M je počet řádků matice Ω)

pořadí	operace	složitost	sčítání	násobení	poznámka
1.	$\sigma = f_{AD}(\mathbf{\Omega})$	$O(nM^2)$	$\frac{1}{2}M^2n$	-	average distance
2.	$\Phi_{\mathbf{\Omega}}(\mathbf{X})$	$O(Mn)$	$M(n+1)$	$5Mn+1$	fuzzy density

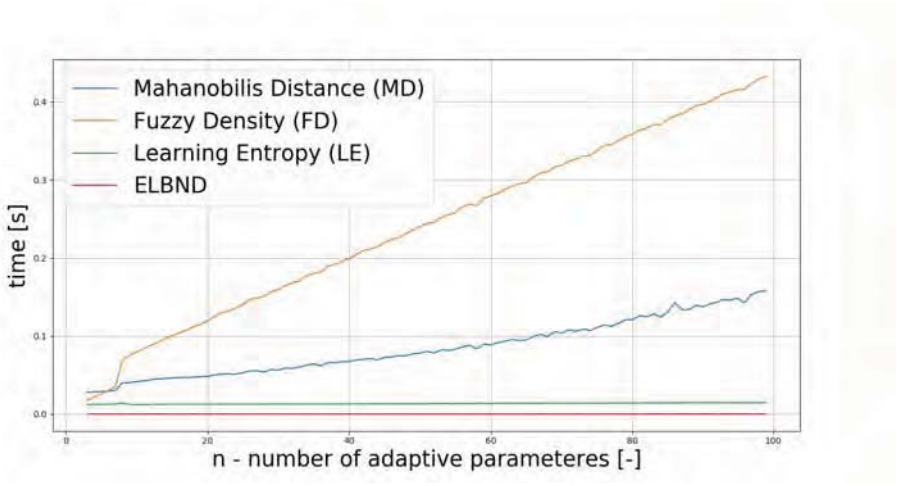
3 Experimentální analýza

Během experimentální analýzy, byl měřen čas na detekci novosti při použiti všech testovaných algoritmů. Adaptivní algoritmy pro detekci byly testovány v rozsahu 3-99 vstupů (parametr n), jelikož tento interval popisuje nejčastější použití daných algoritmů. Testovací data byla rozdělena do segmentů o délce 1000 vzorků. Každý segment byl změřen 1000x. Výsledky čas na segment se získal zprůměrováním těchto 1000 nezávislých testů. Každý jednotlivý test se provedl pro každý algoritmus předtím, než se postoupilo na další test, aby se rovnoměrně rozložilo případné kolísání výkonu testovacího hardware. Algoritmy, které používají paměť předchozích vektorů adaptivních vah, měli všichni shodně nastaveny délku paměti na 500 posledních vektorů.

Získané výsledky jsou zobrazeny v grafu na Obr. 2. Výsledky pro vybrané hodnoty parametru n jsou číselně zaznamenány v Tab. 7.

4 Závěr

V toto článku byla porovnána časová náročnost čtyř různých algoritmů pro detekci novosti (ELBND, LE, FD, MD). Časová náročnost byla posuzována pomocí asymptotické složitosti algoritmů, počtu operací (násobení, adice) v jejich implementaci a také pomocí změření rychlosti v jazyce Python. Metody ELBND, LE a FD mají lineární asymptotickou složitost a metoda MD má kvadratickou asymptotickou složitost(Vzhledem k počtu adaptivních parametrů), což potvrdily výpočetní testy. Do budoucna by bylo zajímavé validovat výsledky i pomocí jiných implementačních jazyků a nástrojů.



Obr. 2: Výsledky - v tomto grafu je znázorněna závislost časové náročnosti jednotlivých algoritmů pro detekci novost a počtu adaptivních parametrů.

Tab. 7: Časová náročnost v ms v závislosti na parametru n pro jednotlivé algoritmy.

n	ELBND	LE	FD	MD
3	0.040565	12.106009	17.114869	27.900899
13	0.057102	12.484739	91.514165	45.028556
23	0.067401	12.566594	132.167673	51.513525
33	0.078859	12.785707	170.275357	60.336528
43	0.091092	13.055424	211.303988	71.177681
53	0.102716	13.517995	251.945065	82.149545
63	0.112905	13.724406	291.961026	95.538772
73	0.128387	14.128523	330.522018	108.705011
83	0.140568	14.402364	370.774985	128.212707
93	0.152008	14.572935	409.528680	146.476234

Poděkování

Tento projekt byl podpořen grantem *SGS18/177/OHK2/3T/12*. Všechny simulace byly provedeny v jazyce *Python*. Zdrojové kódy je možné dostat na požádání od autora.

Literatura

[1] Terran Lane and Carla E Brodley. Approaches to online learning and concept drift for user identification in computer security. In *KDD*, pages 259–263, 1998.

[2] Jeffrey C Schlimmer and Richard H Granger. Beyond incremental processing: Tracking concept drift. In *AAAI*, pages 502–507, 1986.

[3] Leandro L Minku, Allan P White, and Xin Yao. The impact of diversity on online ensemble learning in the presence of concept drift. *IEEE Transactions on Knowledge and Data Engineering*, 22(5):730–742, 2010.

[4] Sanjeev Arora and Boaz Barak. *Computational complexity: a modern approach*. Cambridge University Press, 2009.

[5] Guido Van Rossum and Fred L Drake Jr. *Python tutorial*. Centrum voor Wiskunde en Informatica Amsterdam, The Netherlands, 1995.

[6] Stéfan van der Walt, S Chris Colbert, and Gael Varoquaux. The numpy array: a structure for efficient numerical computation. *Computing in Science & Engineering*, 13(2):22–30, 2011.

[7] Matous Cejnek and Ivo Bukovsky. Concept drift robust adaptive novelty detection for data streams. *Neurocomputing*, 2018.

[8] Matous Cejnek et al. Padasip: Python adaptive signal processing, 2016–. [Online; accessed 2018-06-25].

[9] Ivo Bukovsky. Learning entropy: Multiscale measure for incremental learning. *Entropy*, 15(10):4159–4187, 2013.

[10] Prasanta Chandra Mahalanobis. On the generalized distance in statistics. National Institute of Science of India, 1936.

[11] Mohsen Arefi, Reinhard Viertl, and S Mahmoud Taheri. Fuzzy density estimation. *Metrika*, 75(1):5–22, 2012.

Selected article from
Tento dokument byl publikován ve sborníku

Nové metody a postupy v oblasti přístrojové techniky, automatického řízení a informatiky 2018
New Methods and Practices in the Instrumentation, Automatic Control and Informatics 2018
28. 5. – 30. 5. 2018, Příbram - Podlesí

ISBN 978-80-01-06477-1

Web page of the original document:
http://control.fs.cvut.cz/nmp
http://iat.fs.cvut.cz/nmp/2018.pdf

Obsah čísla/individual articles:
http://iat.fs.cvut.cz/nmp/2018/