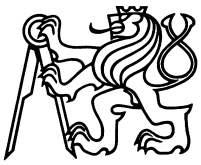
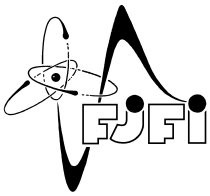


České vysoké učení technické v Praze
Fakulta jaderná a fyzikálně inženýrská



Hierarchické modely síťového provozu

Hierarchical models of network traffic

Bakalářská práce

Autor: **Marek Dědič**

Vedoucí práce: **Ing. Tomáš Pevný, Ph.D.**

Konzultant: **Mgr. Petr Somol, Ph.D.**

Akademický rok: 2016/2017

- Zadání práce -

Poděkování:
Chtěl bych zde poděkovat především svému školiteli, doktoru Tomáši Pevnému, za ochotu, vstřícnost, odborné i lidské zázemí a zdravě kritický pohled při vedení mé bakalářské práce.

Čestné prohlášení:
Prohlašuji, že jsem tuto práci vypracoval samostatně a uvedl jsem všechnu použitou literaturu.

V Praze dne 7. července 2017 Marek Dědič

Název práce:

Hierarchické modely síťového provozu

Autor: Marek Dědič

Obor: Matematická informatika

Druh práce: Bakalářská práce

Vedoucí práce: Ing. Tomáš Pevný, Ph.D., Cisco systems, Inc.

Konzultant: Mgr. Petr Somol, Ph.D., Cisco systems, Inc.

Abstrakt: Současné přístupy k detekci nežádoucího software sledováním síťového provozu klientů využívají ručně navržených příznaků jako části modelu. Tento přístup má několik nevýhod. Tato práce navrhuje plně automatický klasifikátor rozpoznávající aktivity malware na úrovni síťových spojení. K tomu byl využit přístup pomocí multi-instančního učení. Součástí této práce je teoretické zavedení multi-instančního učení pomocí dvou rozdílných formalismů a shrnutí základních myšlenek dosavadních prací v oboru multi-instančního učení. Dále je popsána hierarchická struktura adresy URL jako vstupního objektu klasifikátoru. Je navržen model reflektující tuto inherentní strukturu a vysvětleno, jak využívá multi-instanční učení, v čem se liší a jak byl implementován pomocí umělých neuronových sítí. Jsou zde popsány metody sloužící k vyhodnocení kvality klasifikátoru a k porovnání s předchozím dílem. Navržený klasifikátor je porovnán s nejlepším předchozím modelem a je srovnán vliv jednotlivých parametrů modelu na jeho kvalitu.

Klíčová slova: detekce malware, klasifikace síťového provozu, model adresy URL, multi-instanční učení, učení se reprezentace, strojové učení

Title:

Hierarchical models of network traffic

Author: Marek Dědič

Abstract: The current approach to the detection of unwanted software by monitoring client traffic uses handwritten features as part of the model. This approach has several disadvantages. This thesis proposes a fully automated classifier which recognises malware activity at network connection level. The multi-instance learning approach was used in order to achieve this. As a part of this thesis the multi-instance learning was theoretically defined using two different formalisms and current work in this field was summarised. Subsequently, there was described the hierarchical structure of an URL address which was used as an input for the classifier. A model reflecting this inherent hierarchical structure was proposed and an explanation of how multi-instance learning was utilised and modified was presented together with the description of the implementation of the model using neural networks. Methods of classifier quality assessment were outlined. The classifier presented here was compared with prior art and the influence of model parameters on its quality was assessed.

Keywords: machine learning, malware detection, multi-instance learning, network traffic classification, representation learning, URL address model

Obsah

Úvod

Podle statistik z roku 2014 se v 100% podnikových sítí vyskytuje mimo běžného provozu i provoz způsobený nežádoucím software (srov. *Cisco 2014 Annual Security Report*, 2014). Mnoho moderního nežádoucího software využívá ke své činnosti internetové sítě, ať už jako prostředku šíření, pro komunikaci s řidicím (C&C) serverem či například pro DDoS útoky. Klasický antivirový software spoléhá na systémy pravidel a databázi známého malware k detekci tohoto nežádoucího software. V nedávné době se začaly vyskytovat pokusy o automatizaci antivirového software a využití metod umělé inteligence a strojového učení. Tato práce má za cíl najít klasifikátor síťového provozu, který je schopen o daném klientu určit, zda je nakažen nějakým nežádoucím software, a to pouze z vnějšího pozorování síťového provozu takového klienta bez možnosti fyzického přístupu k danému zařízení. Takový přístup umožňuje snadné nasazení například v podnikových sítích v podobě klasifikátoru zabudovaného do síťových prvků dané organizace, efektivně chránícího všechny klienty na této síti bez nutnosti instalace speciálního software na cílová zařízení. Cílem této práce je najít pomocí metod strojového učení klasifikátor síťového provozu pracující na úrovni jednotlivých síťových spojení, určující, zda tato síťová spojení pocházejí z běžné aktivity klienta, či zda jsou důsledkem aktivity malware komunikujícího po síti. Machlica et al., 2017 navrhli systém pro automatickou detekci tohoto malware pomocí techniky náhodných lesů. I tento přístup však spoléhá na ručně navržené příznaky jako první fázi popisu cílové destinace probíhajícího síťového spojení. S tím se pojí několik nevýhod tohoto přístupu – ruční návrh příznaků je velmi časově náročný, často spoléhá na techniku pokus-omyl a není zaručeno, že se zvyšující se složitostí moderního malware bude stále možné navrhnout dobře fungující příznaky. Vzhledem k tomu, že aktivita a množství výskytů jednotlivých nežádoucích programů se v současné době mění velice rychle v řádu týdnů, není zaručeno, po jak dlouhou dobu bude sada ručně navržených příznaků efektivní, než bude zapotřebí najít novou. Cílem práce je navrhnout plně automatický klasifikátor bez ručně navržených částí, řešící výše popsané problémy. Tento klasifikátor bude vyhodnocen na souboru dat z reálného síťového provozu, čítajícím více jak jednu miliardu síťových spojení, a srovnán s nejlepším předchozím modelem řešícím stejný problém. Hlavním cílem práce je, aby navržený automatický klasifikátor měl alespoň stejnou kvalitu jako srovnávaný model používající ručně navržené příznaky.

Pro řešení této úlohy byl vybrán přístup pomocí tzv. multi-instančního učení. Multi-instanční učení umožňuje navrženému modelu reflektovat hierarchickou strukturu adresy URL, která je používána jako vstupní identifikátor cílové destinace síťového spojení. Pro modelování celých adres URL byl přístup pomocí multi-instančního učení aplikován dvakrát na sebe sama a následně pozměněn, aby odpovídal struktuře a funkci jednotlivých částí adresy URL. Pracuje se s předpokladem, že tato korespondence mezi vstupními daty a modelem tato data popisujícím pozitivně ovlivní kvalitu představeného klasifikátoru. Navržený model je topologicky složitější než srovnatelný model nevyužívající multi-instanční přístup, ale je méně výpočetně náročný.

Součástí této práce je teoretické zavedení multi-instančního učení pomocí množinového formalismu a pomocí stochastického formalismu. Oba tyto formalismy poskytují odlišný pohled na danou problematiku. Následuje popis paradigmat využívaných v multi-instančním učení a shrnutí hlavních myšlenek

předchozích prací věnujících se mutli-instančnímu učení. Dále je teoreticky popsán navržený model a jeho odlišnosti od standardního multi-instančního učení. V kapitole 4 jsou popsány metody použité k zhodnocení kvality navrženého klasifikátoru a k porovnání s předchozími pracemi na této úloze. Je srovnán vliv některých parametrů modelu na jeho kvalitu a vybráno nejlepší nastavení. V následující kapitole jsou poskytnuty výsledky praktického ověření představeného modelu a srovnání těchto výsledků s nejlepším předchozím modelem.

Kapitola 1

Řešená úloha

Řešená úloha je problémem binární klasifikace, tedy problémem, kdy je cílem zařadit objekty do jedné ze dvou tříd, jež nazýváme pozitivní a negativní, tedy navrhnout vhodnou klasifikační funkci, která pro daný vstupní objekt určí příslušnou třídu. Vstupními objekty jsou v tomto případě adresy URL (srov. Berners-Lee et al., 1994). Příklad takové adresy je na obrázku 1.1. **Negativní třídou** se rozumí veškerý síťový provoz, který je součástí běžného provozu uživatele-klienta a jím spuštěných programů. **Pozitivní třídou** se rozumí síťový provoz který pochází z aktivit malware a nežádoucího software.

http://some.domain.org/path/to/file?key=val&other=val2&fin=3141

Obrázek 1.1: Adresa URL

Adresa URL se skládá z několika částí, mezi běžné patří **protokol** (*protocol*), **doména** (*domain* nebo *host*), **port**, **cesta** (*path*), **dotaz** (*query* nebo *searchpart*). Vzhledem k tomu, že protokolů je relativně málo a port se ve většině případů neudává, lze tyto dvě části pominout a zabývat se pouze doménou, cestou a dotazem.

1.1 Struktura adresy URL

Berners-Lee et al., 1994 definuje abecedu (dále označovanou Σ) všech možných znaků v adrese URL jako malá a velká písmena anglické abecedy, čísla a znaky $ - _ . + ! * ' ( ) , % ; / ? : @ & =. Každá adresa URL je slovem abecedy Σ (ne však naopak). Vybrané tři části adresy URL jsou definovány následovně.

Definice 1.1. Doména je podslovem adresy URL konstruovaným následovně: Pokud adresa URL obsahuje podslovo ”://”, doména začíná za tímto podslovem. V opačném případě doména začíná na začátku adresy URL. Doména končí před prvním výskytem znaku ”/” po začátku domény, případně na konci adresy URL, pokud tato už žádný znak ”/” neobsahuje.

Definice 1.2. Pokud je doména sufixem adresy URL, je **cesta** definována jako prázdné slovo. V opačném případě je cesta definována jako podslovo adresy URL, začínající za znakem ”/” ukončujícím doménu. Cesta končí před prvním výskytem znaku ”?” po začátku cesty, případně na konci adresy URL, pokud tato už žádný znak ”?” neobsahuje.

Definice 1.3. Pokud je doména nebo cesta sufixem adresy URL, je **dotaz** definován jako prázdné slovo. V opačném případě je dotaz definován jako sufix adresy URL začínající za znakem ”?” ukončujícím cestu.

http://some.domain.org/path/to/file?key=val&other=val2&fin=3141

Obrázek 1.2: Části adresy URL

Na obrázku 1.2 je příklad dělení adresy URL. Doména ja zvýrazněná červenou barvou, cesta fialovou a dotaz modrou.

Každá z těchto tří částí adresy URL se sama skládá z několika částí, obecně zvaných **tokeny**. Doména se skládá z několika úrovní majících funkci tokenů, lišících se obecností a oddělených znakem ”.”. Cesta se skládá z tokenů, které jsou názvy složek a souborů, jež jsou dotazovány, a jsou odděleny znakem ”/”. Dotaz je tvořen tokeny tvaru klíč–hodnota, oddělenými znakem ”&”. Na obrázku 1.3 je příklad tohoto rozdělení, barvy korespondují s obrázkem 1.2, různé tokeny jedné části adresy URL se liší odstínem téže barvy.

http://some.domain.org/path/to/file?key=val&other=val2&fin=3141

Obrázek 1.3: Části a tokeny adresy URL

Je zřejmé, že záleží na pořadí tokenů domény, definují totiž ”cestu” k cílovému serveru a jsou řazeny od nejkonkrétnějšího k nejobecnějšímu (srov. Mockapetris, 1987). Stejně tak u cesty závisí na pořadí, které definuje adresářové umístění požadovaného souboru na serveru.

1.2 Výhody přístupu pomocí MIL

Klasifikační funkce byla realizována pomocí tzv. **multi-instančního učení** (MIL)[1], poprvé popsaného v Dietterich et al., 1997. Tento přístup, podrobněji popsaný v kapitole 2, vychází z předpokladu, že vzorek nelze reprezentovat vektorem o konstantní délce, ale sadou (dále zvanou **taška**) takových vektorů (dále zvaných **instance**). Dále je předpokládáno, že každou tašku, obsahující neznámý počet instancí, je možno zařadit do nějaké třídy, přestože ne všechny její instance náleží do této třídy. Adresu URL lze považovat za takovou tašku při využití jejího hierarchického modelu, nastíněného v sekci 1.1. Proto lze přístup pomocí multi-instančního učení aplikovat na tento problém.

Přístup pomocí multi-instančního učení produkuje topologicky složitější modely, než je srovnatelný model založený na ručně navržených příznacích. Výměnou za tuto nevýhodu v podobě zvětšené složitosti implementační je zmenšená složitost výpočetní. Multi-instanční model totiž modeluje každou instanci nejprve zvlášť, a teprve poté následuje model pro celý vzorek. To v případě použití neuronové sítě znamená výrazně méně propojení než u srovnatelné plně propojené síťě se stejnými počty neuronů ve stejných vrstvách. Důležitým rozdílem též je, že narozdíl od tohoto modelu je multi-instanční model rozdělen do dvou částí (model pro instance a model pro tašky), přičemž ten samý instanční model je v rámci jedné tašky použit několikrát.

Multi-instanční přístup tedy dovoluje vytvořit model, jehož topologie odpovídá hierarchické struktuře adresy URL, je výpočetně méně náročný než srovnatelný klasický model a umožňuje opětovné použití některých částí. To jsou důvody, proč byl tento přístup zvolen.

[1] resp. jeho speciální varianta, blíže popsaná v kapitole 3

1.3 Trénovací a testovací data

K trénování a testování modelu byla použita data získaná z Cisco Cloud Web Security. Tato data představují záznamy HTTP provozu více jak 100 společností v šesti různých časových úsecích mezi listopadem 2013 a březnem 2015. Datový soubor je rozdělen na data určená k trénování a data určená k ověření kvality navržených klasifikátorů. Oba tyto soubory jsou dále rozděleny na dvě části, z nichž jedna obsahuje pouze legitimní provoz a druhá obsahuje směs legitimního provozu a provozu pocházejícího z činnosti malware. Toto rozdělení datového souboru je shrnuto v tabulce 1.1.

Tabulka 1.1: Souhrn použitých souborů dat

	Počet adres	
Datový soubor	Legitimní	Malware
Trénovací legitimní	417 208 484	0
Trénovací směs	17 390 889	34 359 733
Testovací legitimní	1 522 255 052	0
Testovací směs	2 855 992	4 051 944

Kapitola 2

Multi-instanční učení

V současné době existují dva převládající přístupy k učení z příkladů. Těmi jsou **učení s učitelem** (*supervised learning*) a **učení bez učitele** (*unsupervised learning*). Učení s učitelem má za cíl naučit se správné zařazení příkladů do tříd z trénovacích příkladů, pro které jsou tyto třídy známé. Následně je toto zařazení rozšířeno na nové, neznámé příklady. Učení bez učitele má za cíl naučit se odvodit strukturu příkladů, kde třídy nejsou známé ani pro trénovací data. Oba tyto přístupy předpokádají, že příklady jsou reprezentovány vektory o konstantní délce. Dietterich et al., 1997 navrhli přístup pomocí **multi-instančního učení** (*multi-instance learning, MIL*), které rozšiřuje metodu učení s učitelem na problémy, ve kterých příklady nelze vyjádřit vektory o konstantní délce. Jednotlivé příklady v multi-instančním učení jsou reprezentovány **taškami** (*bags*), které mohou obsahovat libovolné množství objektů. Třídy nemusí být známé na úrovni těchto objektů, stačí pouze znalost tříd na úrovni tašek.

Aby bylo možné korektně matematicky popsat multi-instanční učení, je zapotřebí zavést několik pojmů z teorie množin.

2.1 Multimnožiny

Knuth, 1968 definuje pojem multimnožiny, tedy množiny, ve které se prvky mohou opakovat, následovně.

Definice 2.1. Nechť $\mathbb{A}$ je množina a $m : \mathbb{A} \to \mathbb{N}$. Uspořádanou dvojici $(\mathbb{A}, m)$ nazýváme **multimnožinou** nad množinou $\mathbb{A}$. Pro $a \in \mathbb{A}$ se $m(a)$ nazývá **multiplicitou** (počtem výskytů) prvku a. Pokud $a \in \mathbb{A}$, říkáme, že a je prvkem multimnožiny $(\mathbb{A}, m)$ a značíme

$$a \in (\mathbb{A}, m)$$

Z faktu, že multimnožina je rozšířením pojmu množina, vychází i značení, kdy multimnožina může být definována výčtem prvků, ve kterém je prvek opakován tolikrát, jaká je jeho multiplicita.

Příklad 2.2. Multimnožinu $(\{a, b, c\}, \{(a, 2), (b, 3), (c, 5)\})$ zapisujeme jako $\{a, a, b, b, b, c, c, c, c, c\}$.

Poznámka 2.3. Množina je speciálním případem multimnožiny, ve které má každý prvek multiplicitu 1.

Definice 2.4. Nechť $\mathbb{A}$ je množina. Multimnožina $(\mathbb{B}, m)$ je **podmultimnožinou** množiny $\mathbb{A}$, pokud platí $\mathbb{B} \subset \mathbb{A}$. Značíme $(\mathbb{B}, m) \subset \mathbb{A}$.

Je možné rozšířit i definici potenční množiny, běžně označované $2^{\mathbb{A}}$.

Definice 2.5. Nechť $\mathbb{A}$ je množina. Jako **množinu všech podmultimnožin** $\mathbb{A}$ označujeme množinu

$$\mathcal{P}^M(\mathbb{A}) = \{(\mathbb{B}, m) | (\mathbb{B}, m) \text{ je multimnožina } \wedge (\mathbb{B}, m) \subset \mathbb{A}\}$$

Definice 2.6. Nechť $\mathbb{B} = (\mathbb{A}, m)$ je multimnožina, kde $\mathbb{A}$ je konečná množina. Definujeme **velikost multimnožiny** $\mathbb{B}$ jako

$$|\mathbb{B}| = \sum_{a \in \mathbb{A}} m(a)$$

2.2 Multi-instanční učení

Multi-instanční učení (srov. Dietterich et al., 1997) předpokládá existenci vstupních objektů, náležících obecně do vstupního prostoru $\mathcal{X}$. Dále je předpokládána existence výstupních objektů (tříd) z nějakého výstupního prostoru $\mathcal{Y}$. Za účelem klasifikace jsou jednotlivé vstupní objekty seskupeny do tašek, definovaných následovně.

Definice 2.7. Nechť $\mathcal{X}$ je prostorem vstupních objektů. Potom **prostorem tašek** je multimnožina $\mathcal{B}$ splňující podmínky

1. $\mathcal{B} \subset \mathcal{P}^M(\mathcal{X})$
2. $(\forall x \in \mathcal{X})(\exists b \in \mathcal{B})(x \in b)$

Prvky prostoru tašek se nazývají **taškami**.

Cílem mutli-instančního učení je řešit problémy, ve kterých není známa správná třída pro vstupní objety, avšak je známá třída právě pro tašky vstupních objektů. Dietterich et al., 1997 dávají za příklad multi-instančního problému následující úlohu:

Příklad 2.8. Dveře od skladu v kanceláři mají zámek, od nějž mají klíče všichni zaměstnanci v kanceláři. Každý zaměstananec má svazek klíčů, ze kterých právě jeden je od sdíleného skladu. V kanceláři je ale používán tzv. systém generálního klíče, tedy zatímco některým zaměstancům slouží klíč od skladu pouze k otevření skladu, jiní používají stejný klíč nejenom k přístupu ke skladu, ale i k odemknutí jedněch, či dokonce více dalších dveří. Naší úlohou (například v roli zámečníka) je najít nejobecnější tvar, jaký klíč musí mít, aby odemkl dveře od skladu. To by nám umožnilo inspekcí libovolného klíče určit, zda tento klíč odemkne sklad, či ne. Naše práce je ale ztížena tím, že zaměstnanci nejsou ochotni nám ukázat, který klíč z jejich svazku odemyká sklad, pouze nám předají celý svůj svazek klíčů. Přístup ke skladu je omezen, proto nemůžeme žádný z klíčů vyzkoušet. Obecný tvar klíče můžeme odvodit pouze studiem získaných svazků klíčů.

V tomto příkladu mají jednotlivé klíče (resp. jejich popis nějakým vektorem příznaků, popsaným v kapitole 3.1) funkci vstupních objektů, svazky klíčů jsou taškami. Přestože má smysl se u každého klíče ptát, zda odemyká sklad, není tato informace k dispozici pro žádný z klíčů, pouze pro celé svazky. Má také smysl tvrdit o celých svazcích, že odemykají (respektive neodemykají) dveře od skladu, pokud obsahují alespoň jeden (respektive neobsahují žádný) klíč, kterým lze sklad odemknout. Tím je zaveden pojem **třídy tašky**, který Dietterich et al., 1997 definují následovně:

Definice 2.9. Nechť $\mathcal{X}$ je prostorem vstupních objektů, $\mathcal{Y}$ je prostorem tříd, kde platí

$$(\forall x \in \mathcal{X})(\exists! y_x \in \mathcal{Y})$$

Nechť $\mathcal{B}$ je prostorem tašek. Nechť je v prostoru $\mathcal{Y}$ definována funkce maximum. Potom třídou tašky $b \in \mathcal{B}$ je

$$y_b = \max_{x \in b}(y_x) \in \mathcal{Y}$$

Tato definice se jeví, jak bude dále ukázáno, ve většině případů přinejmenším problematickou, neboť velké množství úloh (včetně úlohy popsané v kapitole 1) předpokládá, že třídy vstupních objektů nejenom nejsou známé, ale nejsou ani dobře definované.

2.3 Metody řešení multi-instančních úloh

V následující podkapitole jsou popsány tři převládající přístupy k řešení multi-instančních problémů. Celá tato podkapitola vychází z Pevny et al., 2016b, Pevny et al., 2016a a Amores, 2013.

2.3.1 Paradigma prostoru instancí

Paradigma prostoru instancí (*Instance-space paradigm*) je původním přístupem k multi-instančnímu učení tak, jak jej popsali Dietterich et al., 1997. Předpokládá existenci (neznámých) tříd pro vstupní objekty a využívá definice 2.9. Je předpokládána existence výstupních objektů pro všechny vstupní objekty (odpovídající instancím), přestože tyto hodnoty nejsou známé. Pro každou tašku je pak předpokládána existence výstupního objektu y_b. Cílem metody je najít klasifikační funkci $f : \mathcal{X} \to \mathcal{Y}$ a posléze určit

$$y_b = \max_{x \in b} \left(f\left(x\right) \right)$$

Existuje mnoho přístupů využívajících paradigmatu prostoru instancí (srov. Andrews et al., 2003 a C. Zhang et al., 2006). Následuje přehled nejdůležitějších z těchto přístupů.

BP-MIP (srov. Zhou; M.-L. Zhang, 2002) řeší problém binární klasifikace (tedy $\mathcal{Y} = \{-1, +1\}$) a předpokládá výstup nějaké vrstevnaté neuronové sítě (*feedforward neural network*) o_{ij} pro j-tý vstupní objekt v i-té tašce. Symbolem y_{b_i} je označována třída tašky b_i. Dále je definována chybová funkce pro vstupní objekty

$$E_{ij} = \begin{cases} 0 & \text{pro} \quad y_{b_i} = +1 \wedge o_{ij} \geq 0.5 \\ 0 & \text{pro} \quad y_{b_i} = -1 \wedge o_{ij} < 0.5 \\ \frac{1}{2}\left(o_{ij} - 0.5\right)^2 & \text{jinak} \end{cases}$$

Pomocí této definice je definována chybová funkce pro celé tašky jako [1]

$$E_i = \begin{cases} \min_{1 \leq j \leq |b_i|} E_{ij} & \text{pro} \quad y_{b_i} = +1 \\ \max_{1 \leq j \leq |b_i|} E_{ij} & \text{pro} \quad y_{b_i} = -1 \end{cases}$$

A dále globální chybová funkce jako

$$E = \sum_{i=1}^{|\mathcal{B}|} E_i$$

Díky takto dobře definované chybové funkci lze využít k trénování neuronové sítě **algoritmus zpětné propagace** (*backpropagation*), modifikovaný pro multi-instanční učení (modifikace popsána v Zhou; M.-L. Zhang, 2002).

EM-DD (srov. Q. Zhang et al., 2002) kombinuje EM (*Expectation-maximization*) algoritmus (srov. Dempster et al., 1977) a DD (*Diverse density*) algoritmus (srov. Maron et al., 1998). Algoritmus EM-DD vychází z nějakého odhadu cílového bodu h v prostoru vstupních objektů, tuto hypotézu h poté vylepšuje EM algoritmem. V E-kroku je z každé tašky vybrán jeden vstupní objekt, o němž se předpokládá, že má největší vliv na třídu tašky. V M-kroku je využito algoritmu gradientního vzestupu (*gradient ascent*) k nalezení nové hypotézy h', která maximalizuje $DD\left(h\right)$ (odpovídající pravděpodobnosti, že h je skutečným cílovým bodem). Opakováním těchto kroků je hypotéza stále vylepšována.

[1] Původní článek chybně v mezích uvádí $|b_j|$.

2.3.2 Paradigma prostoru tašek

Paradigma prostoru tašek (*Bag-space paradigm*) omezuje předpoklady úlohy pouze na existenci tříd na úrovni tašek a zcela opouští představu existence tříd vstupních objektů a s ní i definici 2.9. Přístup pomocí paradigmatu prostoru tašek často definuje nějakou vzdálenostní funkci (*distance function*) nebo jádrovou funkci (*kernel function*), mající podobu

$$k : \mathcal{B} \times \mathcal{B} \to \mathbb{R}_0^+$$

Vzhledem k tomu, že není zaveden pojem třídy vstupního objektu, má hledaná klasifikační funkce tvar $f : \mathcal{B} \to \mathcal{Y}$ a je hledána přímo v této podobě.

Citation-kNN (srov. J. Wang et al., 2000) definuje pro tašky $b_1, b_2 \in \mathcal{B}$ pojem **modifikované Hausdorffovy vzdálenosti** jako

$$H(b_1, b_2) = \min_{x \in b_1} \min_{y \in b_2} \|x - y\|$$

Je využit algoritmus k-nejbližších sousedů (srov. Dasarathy, 1991), modifikovaný o systém tzv. **citací**, kde kromě k-nejbližších sousedů (zde nazývaných **reference**) je definováno i c-nejbližších citujících, kde za c-nejbližších citujících instance $x \in b$ je považováno

$$\text{Citers}(x, c) = \{x_i | \, \text{Rank}(x_i, x) \leq c \wedge x_i \in b\}$$

Pro každou tašku je následovně pomocí Haudorffovy vzdálenosti nalezeno r-nejbližších referencí a c-nejbližších citujících. Pokud je mezi těmito referencemi a citujícími dohromady více pozitivních tašek než negativních, je taška označena za pozitivní. V opačném případě je taška označena za negativní.

Citation-kNN je pouze jedním z přístupů pomocí paradigmatu prostoru tašek. Mezi další patří J. Wang et al., 2000, Kwok et al., 2007, Gärtner et al., 2002, Haussler, 1999, Zhou; Sun et al., 2009 a Muandet et al., 2012.

2.3.3 Paradigma vloženého prostoru

Při použití **paradigmatu vloženého prostoru** (*Embedded-space paradigm*) jsou, stejně jako v případě paradigmatu prostoru tašek, předpoklady omezeny pouze na existenci výstupních objektů na úrovni tašek. Je zde zavedena **vkládající funkce** (*embedding function*) $\phi : \mathcal{B} \to \bar{\mathcal{X}}$, kde $\bar{\mathcal{X}}$ je nějaký vstupní prostor, který může, ale nemusí být totožný s prostorem $\mathcal{X}$. Díky této funkci lze reprezentovat každou tašku vstupním objektem $\phi(b) \in \bar{\mathcal{X}}$ a následně lze použít jakýkoliv algoritmus používaný při klasickém učení s učitelem. K nejjednodušším vkládajícím funkcím patří funkce minimum, maximum či aritmetický průměr.

MILES (srov. Chen; Bi et al., 2006) předpokládá existenci nějakého slovníku vstupních objektů $\mathcal{D}$ a definuje vkládající funkci jako míru podobnosti tašky s objekty ve slovníku, tedy jako

$$\phi : \mathcal{B} \to \mathbb{R}^{|\mathcal{D}|}$$

definovanou po složkách jako

$$\phi_i(b) = \sum_{x \in b} k(x, d_i) \quad \text{kde} \quad d_i \in \mathcal{D}$$

kde

$$k(x, d) = \begin{cases} e^{-\frac{1}{\sigma^2}\|x-d\|^2} & \text{pokud } d \text{ je nejbližším sousedem } x \text{ v } \mathcal{D} \\ 0 \quad \text{jinak} \end{cases}$$

Výběr vstupních objektů ve slovníku $\mathcal{D}$ je prováděn pomocí algoritmu 1-class SVM (srov. Zhu et al., 2004), který má ovšem nevýhodu v podobě velké výpočetní náročnosti.

Mezi další přístupy pomocí paradigmatu vloženého prostoru patří například Cheplygina et al., 2015, Chen; J. Z. Wang, 2004, Foulds, 2008 a M.-L. Zhang et al., 2009.

2.4 Alternativní stochastický formalismus

V následující podkapitole je popsán alternativní formalismus pro mutli-instanční učení, převzatý z Muandet et al., 2012.

Pro prostor vstupních objektů $\mathcal{X}$ nechť existuje měřitelný prostor $(\mathcal{X}, \mathcal{A})$, kde $\mathcal{A}$ je σ-algebrou prostoru $\mathcal{X}$. Nechť $\mathcal{P}^{\mathcal{X}}$ je množinou všech pravděpodobnostních měr prostoru $(\mathcal{X}, \mathcal{A})$. Na každou tašku b pak lze pohlížet jako na realizaci nějaké náhodné veličiny s pravděpodobnostní distribucí $P\left(p_b, y_b\right)$, kde $p_b \in \mathcal{P}^{\mathcal{X}}$ je rozdělení pravděpodobnosti instancí v b a $y_b \in \mathcal{Y}$ je třída této tašky. Prostor všech tašek lze zapsat jako

$$\mathcal{B} = \left\{ x_i \middle| \left(\forall i \in \hat{n}\right) \left(x_i \sim p \in \mathcal{P}^{\mathcal{X}}\right) \right\}$$

Hledaná klasifikační funkce je v tomto formalismu tvaru $f : \mathcal{P}^{\mathcal{X}} \to \mathcal{Y}$.

Je tedy předpokladem, že pravděpodobnostní distribuce jsou odlišné v závislosti na třídě, tj. v případě binární klasifikace $p_+ \neq p_-$ kde $p_+ \sim P(p, +1)$ a $p_- \sim P(p, -1)$.

Muandet et al., 2012 definují jádrovou funkci na prostoru tašek, což je v tomto stochastickém formalismu poněkud složitější. Nechť k je jádrovou funkcí prostoru $\mathcal{X}$, tedy $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. Nechť $\mathcal{H}$ je Hilbertovým prostorem funkcí $\mathcal{X} \to \mathbb{R}$. Tento prostor je Hilbertovým prostorem reprodukujícím jádro k. Dále nechť je definované zobrazení μ jako

$$\mu : \mathcal{P}^{\mathcal{X}} \to \mathcal{H} : p \mapsto \int_{\mathcal{X}} k\left(x, \cdot\right) \mathrm{d}p\left(x\right)$$

Pomocí tohoto zobrazení lze definovat jádrovou funkci $K : \mathcal{P}^{\mathcal{X}} \times \mathcal{P}^{\mathcal{X}} \to \mathbb{R}$ jako

$$K\left(p, q\right) = \left\langle \mu\left(p\right), \mu\left(q\right) \right\rangle_{\mathcal{H}} = \iint k\left(x, y\right) \mathrm{d}p\left(x\right) \mathrm{d}q\left(y\right)$$

2.5 Výhody paradigmatu vloženého prostoru

Pro modelování klasifikační funkce úlohy popsané v kapitole 1 bylo použito paradigmatu vloženého prostoru. Dle autorova názoru nemá smysl definovat pro jednotlivé části adresy URL, zda spadají do pozitivní či negativní třídy, třída je v této úloze dobře definována pouze na úrovni celých adres URL. To je důvodem, proč nebylo zvoleno paradigma prostoru instancí. Důvody pro upřednostnění paradigmatu vloženého prostoru před paradigmatem prostoru tašek jsou dva. Prvním z nich je vyšší výpočetní náročnost algoritmů využívajících paradigmatu prostoru tašek. Vzhledem ke složitosti úlohy a množství dostupných dat je výpočetní náročnost natolik důležitá, že ovlivňuje i výběr algoritmu. Druhým důvodem je možnost použití standardních algoritmů pro učení s učitelem při použití paradigmatu vloženého prostoru.

Poznámka 2.10. Pevny et al., 2016b poukazují na podobnost mezi paradigmatem prostoru tašek a paradigmatem vloženého prostoru, kde jádrovou funkci lze chápat jako funkci vkládající do nějakého Hilbertova prostoru a naopak jádrovou funkci lze aproximovat skalárním součinem v eukleidovském prostoru.

Kapitola 3

Použitý model

Jelikož úloha uvedená v kapitole 1 je úlohou, v níž je adresa URL rozdělena na části, a ty jsou posléze rozděleny na tokeny, je na model použitý k aproximaci klasifikační funkce kladen požadavek, aby reflektoval tuto hierarchickou strukturu vstupních dat. Použitý model využívá přístupu pomocí multi-instančního učení [1], popsaného v kapitole 2, aplikovaného dvakrát na sebe sama. Tím je umožněno modelovat nejprve tři vybrané části adresy URL (doménu, cestu a dotaz) z jejich tokenů, a následovně z těchto tří částí modelovat samotnou adresu URL. V následující kapitole je předpokládána základní znalost fungování umělých neuronových sítí v rozsahu prvních dvou částí I. Goodfellow et al., 2016.

3.1 Reprezentace adresy URL pomocí tašek vektorů

V kapitole 1 byla definována abeceda Σ všech znaků přípustných v adrese URL. Každý token adresy URL je slovem nad touto abecedou. Neuronové sítě, které byly použity, však potřebují vstup v podobě vektoru čísel, dále zvaného **vektor příznaků** (*feature vector*). K tomu je použita funkce ψ, projektující tokeny do prvků eukleidovského prostoru $\mathbb{R}^n$. Tím je definován prostor všech vektorů příznaků jako

$$\mathcal{X}_1 = \psi\left(\Sigma^*\right) \subset \mathbb{R}^n$$

kde Σ^* značí množinu všech slov nad abecedou Σ. Prostor $\mathcal{B}_1 \subset \mathcal{P}^M\left(\mathcal{X}_1\right)$ je konstruován tak, že každá taška v $\mathcal{B}_1$ odpovídá vektorům příznaků tokenů jedné části jedné adresy URL. Za použití paradigmatu vloženého prostoru, popsaného v kapitole 2.3.3, lze najít nějakou vkládající funkci

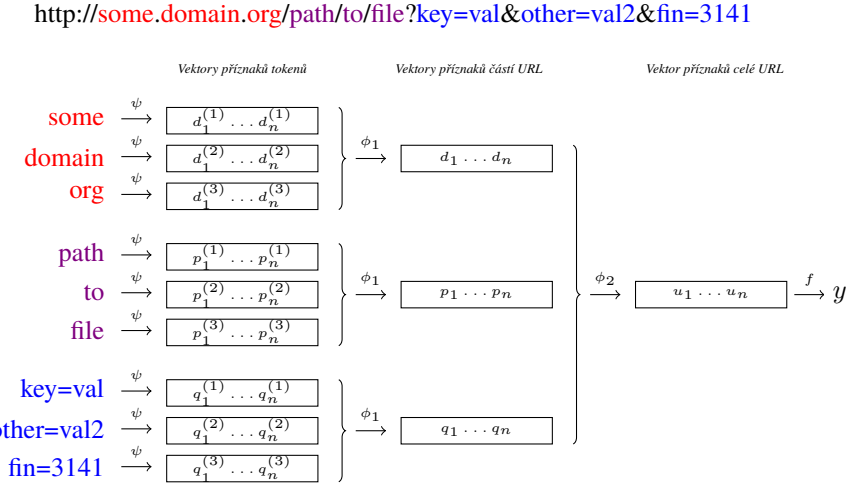
$$\phi_1 : \mathcal{B}_1 \to \mathcal{X}_2$$

kde $\mathcal{X}_2$ je opět nějaký vstupní prostor, který může, ale nemusí být totožný s $\mathcal{X}_1$. Nad tímto prostorem je konstruován prostor tašek $\mathcal{B}_2 \subset \mathcal{P}^M\left(\mathcal{X}_2\right)$ tak, že každá taška v prostoru $\mathcal{B}_2$ odpovídá vektorům příznaků tří částí jedné adresy URL. Opětovným použitím paradigmatu vloženého prostoru lze nalézt vkládající funkci

$$\phi_2 : \mathcal{B}_2 \to \mathcal{X}_3$$

kde $\mathcal{X}_3$ je opět nějaký vstupní prostor, který obecně nemusí být totožný s prostory $\mathcal{X}_1$ a $\mathcal{X}_2$. Tím je celá adresa URL vyjádřena jedním vektorem příznaků z prostoru $\mathcal{X}_3$. Celá tato konstrukce je graficky znázorněna na obrázku 3.1.

[1] resp. jeho speciální varianty, popsané v kapitole 3.2.



Obrázek 3.1: Model adresy URL

3.2 Modifikace MIL pro tuto úlohu

Každá adresa URL musí obsahovat doménu, ale i adresy, kterým chybí cesta nebo dotaz (nebo dokonce cesta i dotaz), mohou být validními adresami URL ve smyslu Berners-Lee et al., 1994. Vzhledem k tomu, jak byl prostor $\mathcal{B}_2$ konstruován, je tedy zřejmé, že platí

$$(\forall b \in \mathcal{B}_2)\,(1 \leq |b| \leq 3)$$

Chybějící části adresy URL je ale možné reprezentovat jedním tokenem odpovídajícím prázdnému slovu[2] abecedy Σ. Díky tomuto rozšíření lze předchozí odhad velikosti tašek v prostoru $\mathcal{B}_2$ zesílit na

$$(\forall b \in \mathcal{B}_2)\,(|b| = 3)$$

Protože doména, cesta a dotaz mají v adrese URL fundamentálně odlišný účel, nedává smysl, aby jim odpovídající vkládající funkce byly totožné. Rozšíření multi-instančního přístupu navržené v předchozím odstavci umožňuje nahradit funkci ϕ_1 třemi různými vkládajícími funkcemi ϕ_D, ϕ_P a ϕ_Q, odpovídajícími doméně, cestě a dotazu respektive. Rovněž předpis pro vkládající funkci ϕ_2 se díky konstantní velikosti tašek z prostoru $\mathcal{B}_2$ zjednoduší na

$$\phi_2 : \mathcal{X}_2^3 \to \mathcal{X}_3$$

Takto modifikovaná abstrakce je graficky znázorněna na obrázku 3.2.

3.3 Metody hledání vkládající a klasifikační funkce

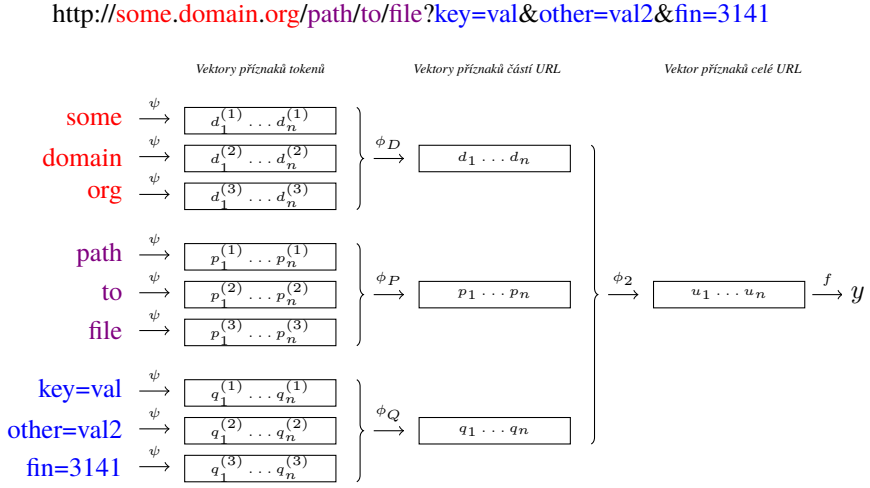
Obecně lze v multi-instančním učení definovat vkládající funkci ϕ jako

$$\phi : \mathcal{B} \to \bar{\mathcal{X}} : b \mapsto g\left(\{k\left(x\right) | x \in b\}\right) \tag{3.1}$$

kde

$$k : \mathcal{X} \to \mathbb{R}^l$$
$$g : \bigcup_{m=1}^{+\infty} \left(\mathbb{R}^l\right)^m \to \bar{\mathcal{X}}$$

[2] Což nutně nemusí znamenat, že jsou reprezentovány nulovým vektorem příznaků.



Obrázek 3.2: Modifikovaný model adresy URL

Za g lze volit například funkce minimum, maximum, aritmetický průměr. Funkce k je realizována neuronovou sítí, která je trénována libovolným algoritmem pro učení s učitelem. Pokud jsou funkce k a f obě realizovány neuronovou sítí, jsou tyto neuronové sítě trénovány společně.

V dalším textu budou předpokládány již konkrétní prostory, a sice

$$\begin{aligned}\mathcal{X}_1 &= \mathbb{R}^n \\ \mathcal{X}_2 &= \mathbb{R}^n \\ \mathcal{X}_3 &= \mathbb{R}^{3n} \\ \mathcal{Y} &= \{-1, +1\}\end{aligned}$$

pro nějaké $n \in \mathbb{N}$.

Přístup pomocí (3.1) byl využit pro vkládající funkce první úrovně, tedy

$$\begin{aligned}\phi_D &: \mathcal{B}_1 \to \mathbb{R}^n : b \mapsto g_D\left(\{k_D\left(x\right) | x \in b\}\right) \\ \phi_P &: \mathcal{B}_1 \to \mathbb{R}^n : b \mapsto g_P\left(\{k_P\left(x\right) | x \in b\}\right) \\ \phi_Q &: \mathcal{B}_1 \to \mathbb{R}^n : b \mapsto g_Q\left(\{k_Q\left(x\right) | x \in b\}\right)\end{aligned}$$

Za vkládající funkci druhé úrovně byla volena funkce konkatenace vektorů, tedy funkce $\phi_2 : (\mathbb{R}^n)^3 \to \mathbb{R}^{3n}$ s funkčním předpisem

$$\begin{aligned}\phi_2\left(\left(x_1^{(1)}, \ldots, x_n^{(1)}\right), \left(x_1^{(2)}, \ldots, x_n^{(2)}\right), \left(x_1^{(3)}, \ldots, x_n^{(3)}\right)\right) = \\ = \left(x_1^{(1)}, \ldots, x_n^{(1)}, x_1^{(2)}, \ldots, x_n^{(2)}, x_1^{(3)}, \ldots, x_n^{(3)}\right)\end{aligned} \quad (3.2)$$

Takto lze vkládající funkci ϕ_2 volit pouze díky modifikaci navržené v kapitole 3.2.

Klasifikační funkce $f : \mathbb{R}^{3n} \to \{-1, +1\}$ je realizována neuronovou sítí, která je trénována společně s neuronovými sítěmi realizujícími funkce k_D, k_P a k_Q.

Takto navržené vkládající funkce neberou v potaz pořadí objektů v jednotlivých taškách. Díky své relativní jednoduchosti byl zvolen tento přístup, přestože pořadí objektů v taškách má vliv na výsledný klasifikátor (srov. Vinyals et al., 2015).

3.4 Použité přenosové funkce

V modelech požitých k řešení úlohy popsané v kapitole 1 byly použity čtyři typy neuronů s různými **přenosovými funkcemi** (*activation function*).

3.4.1 Lineární přenosová funkce

Jedním z nejjednoduších typů neuronů je neuron s lineární přenosovou funkcí, mající pro vstup neuronu x tvar (bez váhy a vychýlení)

$$f : \mathbb{R} \to \mathbb{R} : x \mapsto x$$

3.4.2 ReLU

Neurony typu **ReLU** (*Rectified linear unit*) používají přenosovou funkci (bez váhy a vychýlení), která pro vstup neuronu x dává výstup

$$f : \mathbb{R} \to \mathbb{R} : x \mapsto \max\{0, x\}$$

3.4.3 Maxout

Pro neurony typu **maxout** (srov. I. J. Goodfellow et al., 2013) nelze přenosovou funkci definovat stejně jednoduše jako v případě neuronů typu ReLU. Přenosová funkce h v tomto případě nebývá definována pro jednotlivé neurony, ale pro celou vrstvu neuronů po složkách $h_i : \mathbb{R}^n \to \mathbb{R}$ jako funkce (včetně váhy $\mathbf{W}$ a vychýlení b)

$$h_i(x) = \max_{j \in [1,k]} z_{ij}$$

kde

$$z_{ij} = x^T \mathbf{W}_{\cdot ij} + b_{ij}$$

kde

$$\mathbf{W} \in \mathbb{R}^{n,m,k} \qquad b \in \mathbb{R}^{m,k}$$

Lze tedy vrstvu maxout chápat jako k lineárních vrstev, z nichž je vybráno maximum výstupů.

3.4.4 Softmax

Přenosová funkce pro neurony typu **softmax** je také definována pro celou vrstvu po složkách $h_i : \mathbb{R}^n \to \mathbb{R}$ jako funkce (bez váhy a vychýlení)

$$h_i(x) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}}$$

Výhodou této přenosové funkce pro její využití spolu s objektivní funkcí popsanou v kapitole 3.5 je její jednoduchá derivace tvaru

$$\frac{\partial h_i}{\partial x_j} = \begin{cases} h_i(1 - h_i) & \text{pro} \quad i = j \\ -h_i h_j & \text{jinak} \end{cases}$$

3.5 Použitá objektivní funkce

Při využití neuronů s přenosovou funkcí softmax jako poslední vrstvy neuronové sítě lze za objektivní funkci položit funkci, která pro parametry modelu θ, vstupní vektor x a skutečnou třídu y maximalizuje věrohodnostní funkci

$$\mathcal{L}(\theta|y,x)$$

Tuto věrohodnostní funkci lze zapsat jako sdruženou pravděpodobnost, kterou naopak lze zapsat pomocí podmíněné pravděpodobnosti, tedy platí

$$\mathcal{L}(\theta|y,x) = P(y,x|\theta) = P(y|x,\theta)\,P(x|\theta)$$

Pro pevné hodnoty parametrů θ tedy platí

$$\mathcal{L}(\theta|y,x) = P(y|x) = \prod_{i=1}^{n} P(y_i|x)^{y_i}$$

Hodnota $P(y_i|x)$ je v poslední vrstvě aproximována neurony typu softmax s přenosovou funkcí h_i, tedy je maximalizována hodnota

$$\prod_{i=1}^{n} h_i^{y_i}$$

Úloha maximalizace této hodnoty je ekvivalentní úloze minimalizace jejího záporného logaritmu. Je tedy minimalizována hodnota takto definované funkce J, dané předpisem

$$J = -\log \prod_{i=1}^{n} h_i^{y_i} = -\sum_{i=1}^{n} y_i \log h_i$$

Funkce J je nazývána **objektivní funkcí minimalizující cross-entropii**. V případě binární klasifikace ($n = 2$) lze tuto funkci zapsat jako

$$J = -y_i \log h_i - (1 - y_i) \log (1 - h_i)$$

Výhodou objektivní funkce minimalizující cross-entropii je její jednoduchá derivace tvaru

$$\frac{\partial J}{\partial x_i} = h_i - y_i$$

Celá tato podkapitola byla zpracována podle Roelants, 2017.

3.6 Metody trénování neuronových sítí

Jednou ze základních metod trénování neuronových sítí je metoda **gradientního sestupu** (srov. Cauchy, 1847). Předpokládá se existence nějaké objektivní funkce $J(\theta)$, jejíž minimum je hledáno. $\theta \in \mathbb{R}^d$ jsou učené parametry modelu. Tyto parametry se opakovaně mění, dokud není nalezeno požadované minimum funkce J. V metodě gradientního sestupu jsou nové parametry $\hat{\theta}$ počítány pomocí vztahu

$$\hat{\theta} = \theta - \eta \nabla_\theta J(\theta)$$

kde η je krok učení.

Při použití metody **stochastického gradientního sestupu** je tato korekce prováděna pro každou dvojici vstupního objektu x a výstupního objektu y, tedy jde o metodu tvaru

$$\hat{\theta} = \theta - \eta \nabla_\theta J\left(\theta; x, y\right)$$

Výhodou této metody je efektivita učení při velkém množství trénovacích dat, na druhou stranu je její nevýhodou velká fluktuace hodnot objektivní funkce.

Kompromisem mezi základním gradientním sestupem a stochastickým gradientním sestupem je metoda **mini-batch gradientního sestupu**. Tato metoda je srovnatelně efektivní jako stochastický gradientní sestup při vyšší stabilitě hodnot objektivní funkce. Při použití metody mini-batch gradientního sestupu je z dostupných vstupních a výstupních objektů vybírána podmnožina zvaná **mini-batch** a jsou určovány nové parametry jako

$$\hat{\theta} = \theta - \eta \nabla_\theta J\left(\theta; x^{(i,\ldots,i+n)}, y^{(i,\ldots,i+n)}\right)$$

Metoda gradientního sestupu je základem pro mnoho moderních metod trénování neuronových sítí (srov. Duchi et al., 2011, Zeiler, 2012, Tieleman et al., 2012 a Kingma et al., 2014). Použitá metoda je další metodou založenou na gradientním sestupu.

ADAM (srov. Kingma et al., 2014) je variantou stochastického gradientního sestupu s proměnlivým krokem učení. K správnému nastavení tohoto kroku je v této metodě přenášena hodnota gradientu a jeho čtverce mezi jednotlivými kroky. Metoda ADAM je parametrizována třemi parametry β_1, β_2 a ε. V každém kroku t jsou počítány momenty

$$m_t = \beta_1 m_{t-1} + \left(1 - \beta_1\right) \nabla_\theta J\left(\theta\right)$$

$$v_t = \beta_2 v_{t-1} + \left(1 - \beta_2\right) \left(\nabla_\theta J\left(\theta\right)\right)^2$$

Vzhledem k tomu, že počáteční hodnoty jsou $m_0 = 0$ a $v_0 = 0$, bylo by učení zpočátku velice pomalé. Tomu lze předejít použitím opravných hodnot

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$

Nové hodnoty parametrů modelu jsou nalezeny jako

$$\hat{\theta} = \theta - \frac{\eta \hat{m}_t}{\sqrt{\hat{v}_t} + \varepsilon}$$

Kingma et al., 2014 navrhují hodnoty parametrů metody

$$\beta_1 = 0.9 \qquad \beta_2 = 0.999 \qquad \varepsilon = 10^{-8}$$

Metoda ADAM byla vybrána z důvodů použití adaptivního kroku učení, vedoucího ke stabilnější výsledné hodnotě a počáteční rychlosti díky využití opravných hodnot. Mezi další výhody patří jednoduché použití díky dobře fungujícím hodnotám parametrů, které byly navrženy autory metody.

Kapitola 4

Metody vyhodnocení

Úloha popsaná v kapitole 1 je úlohou **binární klasifikace**. To znamená, že jejím cílem je zařadit vstupní objekty do jedné ze dvou tříd, označovaných jako pozitivní a negativní. Předpokládá se existence nějaké klasifikační funkce $f : \mathcal{X} \to \{-1, +1\}$, která pro každý vstupní objekt určí příslušnou třídu, symbolizovanou hodnotou -1 nebo $+1$. Model popsaný v kapitole 3 se snaží tuto neznámou klasifikační funkci aproximovat. Za účelem zhodnocení kvality uvedené aproximace je v této kapitole nejprve zavedeno několik pojmů vztahujících se k binární klasifikaci a následně popsáno několik metod vizualizace kvality binárních klasifikátorů.

Definice 4.1. Nechť $r \in \{-1, +1\}$ je správným výsledkem pro nějaký vstupní objekt v úloze binární klasifikace. Nechť $p \in \{-1, +1\}$ je aproximací r. Odhad p je nazýván

i. **Pozitivní** (*Prediction positive*) pokud $p = +1$.

ii. **Negativní** (*Prediction negative*) pokud $p = -1$.

iii. **Pravdivě pozitivní** (*True positive*) pokud $r = +1 \wedge p = +1$.

iv. **Pravdivě negativní** (*True negative*) pokud $r = -1 \wedge p = -1$.

v. **Falešně pozitivní** (*False positive*) pokud $r = -1 \wedge p = +1$.

vi. **Falešně negativní** (*False negative*) pokud $r = +1 \wedge p = -1$.

Falešně pozitivní odhad je také nazýván **chybou prvního druhu**, falešně negativní odhad **chybou druhého druhu**.

Standardním způsobem, jak jsou tyto hodnoty zapisovány, je pomocí **kontingenční tabulky**. Její podoba je znázorněna v tabulce 4.1.

Tabulka 4.1: Kontingenční tabulka

		Reálný výsledek	
		Pozitivní	Negativní
Předpověď	Pozitivní	Počet pravdivě pozitivních odhadů	Počet falešně negativních odhadů
	Negativní	Počet falešně pozitivních odhadů	Počet pravdivě negativních odhadů

4.1 Indikátory kvality binárního klasifikátoru

Cílem jakéhokoliv odhadu v úloze binární klasifikace je minimalizovat počet falešně pozitivních a falešně negativních odhadů. Ne vždy je ale možné minimalizovat obě tyto hodnoty a je třeba o nich uvažovat vždy najednou – aproximace, která označí všechny objekty jako pozitivní, nebude mít žádné falešně negativní odhady, ale v praxi nemá taková aproximace žádnou hodnotu. Obdobně aproximace označující všechny objekty za negativní nebude produkovat žádné falešně pozitivní odhady, ale opět to není vhodná aproximace. – Aby tedy bylo možno uvažovat o kvalitě nějakého binárního klasifikátoru, jsou dále zavedeny některé **indikátory kvality**.

Definice 4.2. Nechť tp je počet pravdivě pozitivních odhadů nějakého binárního klasifikátoru na daném souboru dat. Nechť pp je celkový počet pozitivních odhadů tohoto binárního klasifikátoru (tedy počet pravdivě pozitivních a falešně pozitivních). Jako **přesnost** (*precision*) tohoto klasifikátoru je označována hodnota

$$\text{precision} = \frac{\text{tp}}{\text{pp}}$$

Přesnost je jednou z nejdůležitějších vlastností zvláště v oboru počítačové bezpečnosti. Nízká přesnost například antivirového software by znamenala velké množství programů chybně označených za malware, což může mít za následek, že uživatel antivirovému programu přestane věřit, či ho dokonce vypne úplně (srov. Vejmelka, 2017).

Definice 4.3. Nechť tp je počet pravdivě pozitivních odhadů nějakého binárního klasifikátoru na daném souboru dat. Nechť rp je celkový počet pozitivních vzorků v datovém souboru (tedy počet pravdivě pozitivních a falešně negativních). Jako **odezva** (*recall*) tohoto klasifikátoru je označována hodnota

$$\text{recall} = \text{tpr} = \frac{\text{tp}}{\text{rp}}$$

Odezva je také nazývána **pravdivě pozitivní mírou** (*true positive rate*).

V oboru počítačové bezpečnosti se odezva jeví jako velice důležitá, zachytit nežádoucí software je totiž většinou původní motivací pro tvorbu klasifikátorů. Je tedy cílem dosáhnout co největší odezvy při co nejmenším snížení přesnosti.

Definice 4.4. Nechť fp je počet falešně pozitivních odhadů nějakého binárního klasifikátoru na daném souboru dat. Nechť rn je celkový počet negativních vzorků v datovém souboru (tedy počet pravdivě negativních a falešně pozitivních). Jako **falešně pozitivní míra** (*false positive rate*) tohoto klasifikátoru je označována hodnota

$$\text{fpr} = \frac{\text{fp}}{\text{rn}}$$

Falešně pozitivní míra koresponduje s pravděpodobností, že objekt klasifikátorem označený jako pozitivní je ve skutečnosti negativní, je to tedy opět indikátor důležitý hlavně při problémech, kdy je falešně pozitivní odhad vysoce nežádoucí, například v medicíně (srov. Mac Namee et al., 2002).

Definice 4.5. Nechť fn je počet falešně negativních odhadů nějakého binárního klasifikátoru na daném souboru dat. Nechť rp je celkový počet pozitivních vzorků v datovém souboru (tedy počet pravdivě pozitivních a falešně negativních). Jako **falešně negativní míra** (*false negative rate*) tohoto klasifikátoru je označována hodnota

$$\text{fnr} = \frac{\text{fn}}{\text{rp}}$$

Věta 4.6. *Platí*

$$\text{tpr} + \text{fnr} = 1$$

Definice 4.7. Nechť precision je přesnost nějakékého binárního klasifikátoru a recall je jeho odezva. Jako **F-skóre** či také **$\mathbf{F_1}$ skóre** tohoto klasifikátoru je označován harmonický průměr jeho přesnosti a odezvy, tedy

$$F_1 = 2\,\frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

F-skóre je jedinou zde zavedenou hodnotou, která dává do spojitosti jak falešně pozitivní, tak falešně negativní odhady. Proto může být použito jako primární indikátor použitý při volbě pracovního bodu, popsaného v kapitole 4.2.

4.2 Volba pracovního bodu

Přestože v úloze binární klasifikace jsou objekty zařazované do jedné ze dvou tříd, může nastat případ, kdy aproximace klasifikační funkce má jiný obor hodnot než množinu $\{-1, +1\}$. V obecném případě může být oborem hodnot této aproximace celý obor $\mathbb{R}$. Je tedy třeba rozhodnout, které hodnoty binárního klasifikátoru budou považovány za pozitivní a které za negativní. Zřejmě se nabízí přístup, kdy je stanoven nějaký práh (taktéž zvaný **pracovní bod**) a všechny hodnoty vyšší než tento práh jsou považovány za pozitivní odhad, všechny hodnoty nižší než tento práh jsou považovány za negativní odhad. Indikátory kvality lze poté chápat jako funkce takového prahu. Volba pracovního bodu ovšem není triviálním úkolem.

Jedním z možných řešení problému volby pracovního bodu je přístup pomocí určování hodnoty indikátorů kvality binárního klasifikátoru pro všechny možné prahy. Vzhledem k předpokladu konečnosti klasifikovaného souboru dat jsou tyto indikátory (považované za funkce prahu) po částech konstantní a díky tomu je postačuje vyhodnotit v konečně mnoha bodech. Pro velké soubory dat (například ten popsaný v kapitole 1.3) není ale ani tato metoda výpočetně možná. Zvoleným řešením je nalézt kvantily všech možných hodnot pracovního bodu a vyhodnotit indikátory kvality v těchto bodech.

4.3 Křivky zobrazující vlastnosti binárního klasifikátoru

V následující podkapitole jsou popsány čtyři křivky, pomocí kterých lze vizualizovat vlastnosti binárních klasifikátorů. V celé podkapitole jsou indikátory kvality považovány za funkce prahu.

4.3.1 PR křivka

PR křivka (*Precision-Recall curve*) je křivkou zobrazující vztah mezi přesností a odezvou klasifikátoru pro různé prahy. PR křivka grafem množiny

$$\{(\text{recall}\,(x)\,, \text{precision}\,(x)) | x \in \mathbb{R}\}$$

Jde tedy o obor hodnot funkce $\mathbb{R} \to \mathbb{R}^2$, nikoliv o graf funkce $\mathbb{R} \to \mathbb{R}$.

PR křivka má vysokou důležitost právě pro problémy z oblasti počítačové bezpečnosti, dále pro vyhodnocování vyhledávačů, sběru dat, systémů navrhujících doporučení zákazníkům, tedy obecně problémů, kde je kladena jiná důležitost na přesnost a odezvu. V takovém případě PR křivka umožňuje vybrat pracovní bod, který je nejlepší pro konkrétní úlohu.

4.3.2 ROC křivka

ROC křivka (*Receiver operating characteristic curve*) je křivkou zobrazující vztah mezi falešně pozitivní mírou a pravdivě pozitivní mírou klasifikátoru pro různé prahy. ROC křivka je grafem množiny

$$\{(\text{fpr}(x), \text{tpr}(x)) | x \in \mathbb{R}\}$$

V grafu ROC křivky je vyznačena i množina všech bodů (x, x), neboť náhodně volená klasifikační funkce by měla ROC křivku blízkou této množině. Osa X grafu ROC křivky je v logaritmickém měřítku.

ROC křivka je standardním způsobem vyhodnocování kvality binárních klasifikátorů a je používána napříč obory, ve kterých se vyskytují problémy binární klasifikace

4.3.3 DET křivka

DET křivka (*Detection error tradeoff curve*, srov. Martin et al., 1997) je modifikací ROC křivky. DET křivka je grafem množiny

$$\{(\text{fpr}(x), \text{fnr}(x)) | x \in \mathbb{R}\}$$

Z věty 4.6 je zřejmé, že DET křivka je ROC křivkou překlopenou podle osy Y. Rozdílem je měřítko, v jakém je tato křivka vykreslována. Na obě osy grafu je aplikována funkce

$$q(x) = \sqrt{2} \cdot \text{erf}^{-1}(2x - 1)$$

kde

$$\text{erf}(x) = \frac{1}{\sqrt{\pi}} \int_{-x}^{x} e^{-t^2} \mathrm{d}t$$

DET křivka je nejméně častou ze zde popisovaných křivek. Jde o modifikaci ROC křivky, mezi jejíž výhody patří nelineární měřítko, umožňující porovnávat i velice podobné klasifikátory. DET křivka je (na rozdíl od ROC křivky) relativně přímá a její tvar je indikátorem preference mezi falešně pozitivními a falešně negativními odhady.

4.3.4 Křivka F-skóre

Křivka F-skóre je grafem funkce $x \mapsto F_1(x)$.

4.4 Důvody vedoucí k vlastní implementaci

K výpočtu křivek popsaných v kapitole 4.3 bylo vyzkoušeno několik softwarových balíčků dostupných pro jazyk Julia (Zea, 2015, Leeuwen, 2016 a Lin, 2017). Žádný z těchto balíčků však neumožňoval výpočet indikátorů kvality z důvodů příliš vysoké výpočetní či paměťové náročnosti. Všechny tyto balíčky rovněž neumožňují paralelní výpočet indikátorů kvality. Proto byl navržen vlastní balíček s názvem EduNetsEvaluate.jl, který umožňuje výpočet indikátorů kvality popsaných v kapitole 4.1 i pro soubory dat, které jsou srovnatelně velké jako soubor popsaný v kapitole 1.3. Výpočet všech indikátorů kvality je z velké části paralelizovaný, což umožňuje plně využít výpočetní kapacity moderních vícejádrových procesorů a procesorů využívajících technologii Hyper-threading.

4.5 Implementace

Následující podkapitola obsahuje přehled metod použitých k implementaci paralelního výpočtu PR křivky a ROC křivky.

Vzhledem k tomu, že použitý soubor dat je rozdělen do mnoha částí, je možné každou část zpracovat zvlášť a následně agregovat výsledky. Mnoho částí použitého souboru dat obsahuje pouze objekty negativní třídy. Pokud je předpokládáno, že všechny objekty jsou seřazené podle hodnoty jejich předpovědi, při postupném procházení se při přechodu od negativního objektu k přímo následujícímu negativnímu objektu nezmění hodnota odezvy. Tedy pokud bude PR křivka počítána pouze z pozitivních objektů, na její tvar to nebude mít žádný vliv. Pro ROC křivku platí stejné pozorování. Tím je umožněn výpočet vhodných hodnot prahů (ve smyslu kapitoly 4.2) pouze ze souborů, v nichž jsou nějaké pozitivní objekty. Tento výpočet je popsán v algoritmu 4.1.

Algoritmus 4.1 Výpočet prahů

Require: $mixed\_files$ ▷ Soubory obsahující pozitivní i negativní objekty

$thresholds \leftarrow$ ARRAY $[\,]$ [SIZE$(mixed_files)$] ▷ Pole polí
parallel for $file \in mixed\_files$ **do**
  $thresholds\,[\,] \leftarrow$ GET_PREDICTED$(file)$
end for
$thresholds \leftarrow$ CONCATENATE$(thresholds)$
$thresholds \leftarrow$ QUANTILES$(thresholds)$

Následně lze spočítat odezvu binárního klasifikátoru, k jejímuž výpočtu opět postačují pouze soubory, obsahující nějaké pozitivní objekty. Paralelizace tohoto výpočtu je umožněna přístupem, kdy pro každý soubor je spočítán celkový počet pozitivních objektů v souboru a pro každý práh počet pravdivě pozitivních odhadů. Tyto hodnoty jsou poté sečteny přes všechny soubory a na jejich základě je spočítána odezva. Tento výpočet je popsán v algoritmu 4.2. Pro výpočet přesnosti je již potřeba zohlednit všechny soubory včetně těch, které obsahují pouze negativní objekty. Pro každý soubor je zvlášť spočítán počet pravdivě pozitivních odhadů a celkový počet pozitivních odhadů. Hodnoty jsou poté sečteny přes všechny soubory a na jejich základě je spočítána přesnost. Výpočet je analogický s výpočtem odezvy.

Algoritmus 4.2 Výpočet odezvy

Require: $mixed\_files$ ▷ Soubory obsahující pozitivní i negativní objekty
Require: $thresholds$ ▷ Prahy

$RP \leftarrow$ ARRAY [SIZE$(mixed_files)$] ▷ Pole čísel
$TP \leftarrow$ ARRAY [SIZE$(thresholds)$] [SIZE$(mixed_files)$] ▷ Pole polí
parallel for $file \in mixed\_files$ **do**
  $RP\,[\,] \leftarrow$ GET_REAL_POSITIVES$(file)$
  $TP\,[\,] \leftarrow$ GET_TRUE_POSITIVES$(file, thresholds)$
end for
$RP \leftarrow \sum RP$
$TP \leftarrow element\_wise \sum TP$
$recall = \frac{TP}{RP}$

Pokud jsou objekty nejprve seřazeny podle hodnoty jejich předpovědi, lze počet pravdivě pozitivních i počet pozitivních odhadů počítat s asymptotickou složitostí $\mathcal{O}(n)$, což spolu se složitostí řazení dává cel-

kovou asymptotickou složitost $\mathcal{O}(n \log n)$. Tento způsob efektivního výpočtu počtu pravdivě pozitivních odhadů je popsán v algoritmu 4.3.

Algoritmus 4.3 Výpočet počtu pravdivě pozitivních odhadů se složitostí $\mathcal{O}(n)$

Require: $predicted$ ▷ Odhady pro nějaký soubor objektů, seřazené vzestupně
Require: $real$ ▷ Skutečné třídy pro tento soubor ve stejném pořadí
Require: $thresholds$ ▷ Prahy

```
TP ← ARRAY[SIZE(thresholds)]
TPcounter ← COUNT_POSITIVES(real)
THcounter ← 1
for i ← 1 to SIZE(thresholds) do                ▷ Odhady menší než nejmenší z prahů
    if predicted[1] < thresholds[THcounter] then
        break
    end if
    TP[THcounter] ← TPcounter
    THcounter ← THcounter + 1
end for
if real[1] = +1 then
    TPcounter ← TPcounter − 1
end if
j ← 1
while j < SIZE(predicted) − 1 do                ▷ Odhady mezi nejmenším a největším prahem
    if predicted[j] < thresholds[THcounter] ∧ predicted[j + 1] ≥ thresholds[THcounter]
then
        TP[THcounter] ← TPcounter
        THcounter ← THcounter + 1
    else
        if real[j + 1] = +1 then
            TPcounter ← TPcounter − 1
        end if
        j ← j + 1
    end if
end while
for i ← THcounter to SIZE(TP) do                ▷ Odhady větší než největší z prahů
    TP[i] = 0
end for
```

Výpočet celkového počtu pozitivních odhadů je obdobný. Ve skutečné implementaci jsou tyto hodnoty počítány kvůli vyšší efektivitě najednou. Ze stejného důvodu je najednou počítána i přesnost s odezvou. Výpočet ROC křivky je obdobný jako výpočet PR křivky.

K výpočtu křivky F-skóre jsou využity hodnoty vypočítané pro PR křivku. K výpočtu DET křivky jsou využity hodnoty vypočítané pro ROC křivku a věta 4.6.

Kapitola 5

Výsledky

Navržený model je velmi obecný a vyžaduje nastavení mnoha volných parametrů. Jedná se zejména o parametry použité umělé neuronové sítě, jako je počet vrstev, počet neuronů v každé vrstvě, volba přenosových funkcí aj. Dále o volbu a parametry trénovacího algoritmu, k nimž patří volba objektivní funkce a váha pozitivní třídy. Lze nastavit také parametry zobrazení ψ, projektujícího tokeny do vektorů příznaků, jimiž jsou délka vektoru příznaků a délka podslov využívaných k jejich generování.

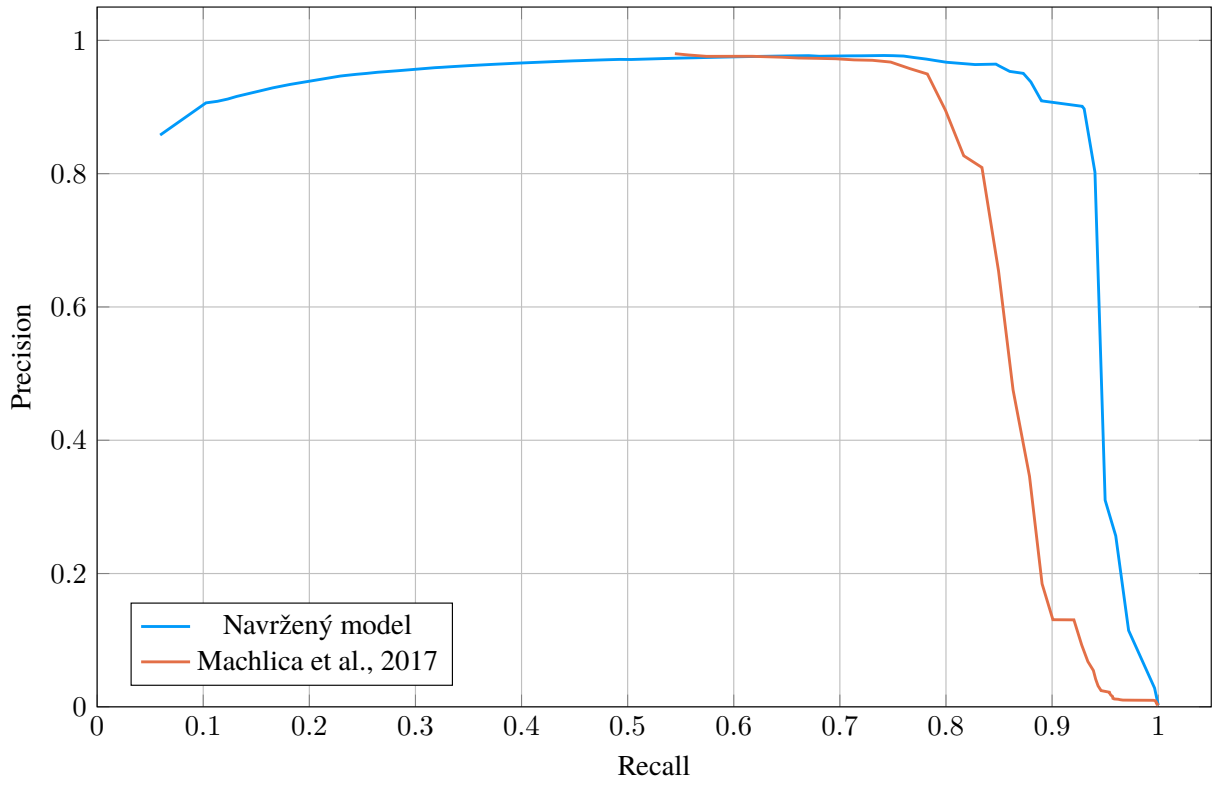
V následujících podkapitolách je nejprve porovnáno nejlepší navržené řešení s Machlica et al., 2017 a následně srovnán vliv některých parametrů modelu na jeho kvalitu.

5.1 Srovnání s nejlepším předchozím modelem

Obrázek 5.1 srovnává PR křivky nejlepšího modelu (srovnání parametrů v následujících podkapitolách) s nejlepším předchozím modelem pro soubor dat popsaný v kapitole 1.3 (srov. Machlica et al., 2017). Nejlepší model má vstupní vektor příznaků o délce 2053, příznaky jsou generované z trigramů. Všechny tři funkce operující na úrovní částí adresy URL jsou shodně realizovány jednou vrstvou 20 neuronů s přenosovou funkcí ReLU, následovanou agregací tašky funkcí aritmetického průměru. Funkce operující na úrovni celé adresy URL obsahuje jednu vrstvu 60 neuronů s lineární přenosovou funkcí, následovanou výstupní vrstvou 2 neuronů s přenosovou funkcí softmax. Síť byla učena metodou ADAM minimalizující cross-entropii, ukončenou po 25 000 krocích. Váha na pozitivní třídě byla 0.5. Mini-batch obsahovaly 100 pozitivních a 100 negativních vzorků. Srovnávaný předchozí model využívá rozšířenou nepublikovanou sadu 557 příznaků z Machlica et al., 2017. Model obsahoval 3 vrstvy 128 neuronů typu ReLU, následované výstupní vrstvou 2 neuronů s přenosovou funkcí softmax. Síť byla učena metodou ADAM minimalizující cross-entropii, ukončenou po 100 000 krocích. Váha na pozitivní třídě byla 0.5. Mini-batch obsahovaly 1000 pozitivních a 1000 negativních vzorků. Všechny křivky v této kapitole byly počítány s kvantilizací pomocí percentilů (ve smyslu kapitoly 4.2).

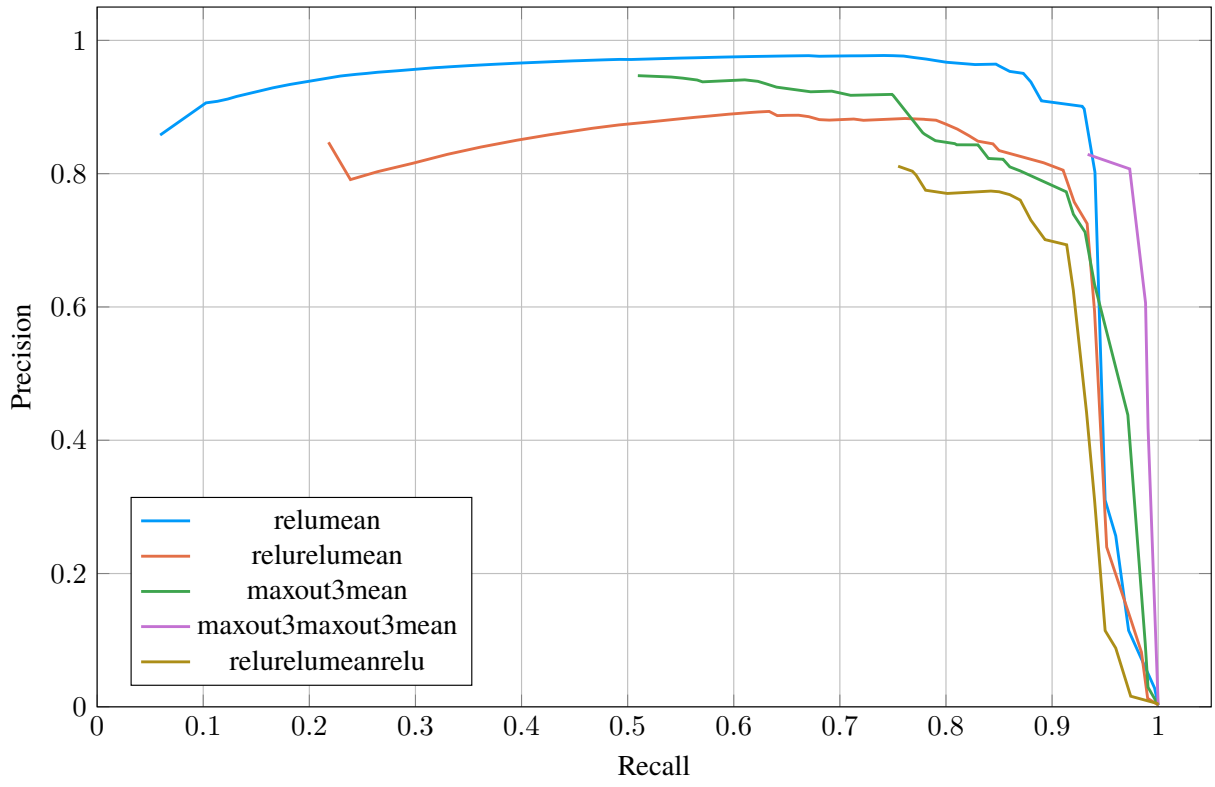
5.2 Srovnání vlivu topologie sítě

Bylo vyzkoušeno 5 různých topologií neuronových sítí, odpovídajících modelu popsanému v kapitole 3. Všechny tyto topologie používají tři identické neuronové sítě realizující funkce na úrovni částí adresy URL. Topologie označované jako **relumean**, **relurelumean**, **maxout3mean** a **maxout3maxout3mean** shodně realizují klasifikační funkci na úrovni celé adresy URL jednou vrstvou neuronů s lineární přenosovou funkcí a jednou vrstvou s přenosovou funkcí softmax. Topologie relumean používá pro funkce na úrovni částí adresy URL jednu vrstvu typu ReLU, následovanou agregační funkcí aritmetického průmě-



Graf 5.1: Srovnání PR křivek s nejlepším předchozím modelem

ru. Topologie relurelumean používá dvě vrstvy typu ReLU následované agregační funkcí aritmetického průměru. Topologie maxout3mean používá jednu vrstvu typu maxout3 (tj. maxout s parametrem $k = 3$) následovanou agregační funkcí aritmetického průměru. Topologie maxout3maxout3mean využívá 2 vrstvy typu maxout3 následované agregační funkcí aritmetického průměru. Topologie **relurelumeanrelu** má funkce na úrovni částí adresy URL shodné s topologií relurelumean, avšak funkci na úrovni celé adresy URL realizuje jako jednu vrstvu typu ReLU následovanou jednou vrstvou s lineární přenosovou funkcí následovanou jednou vrstvou typu softmax. Ostatní parametry jsou shodné s parametry popsanými v kapitole 5.1. PR křivky všech těchto topologií jsou zobrazeny na obrázku 5.2.



Graf 5.2: Srovnání PR křivek různých topologií

5.3 Srovnání vlivu parametrů generátoru příznaků

Algoritmus 5.1 Generátor vektorů příznaků

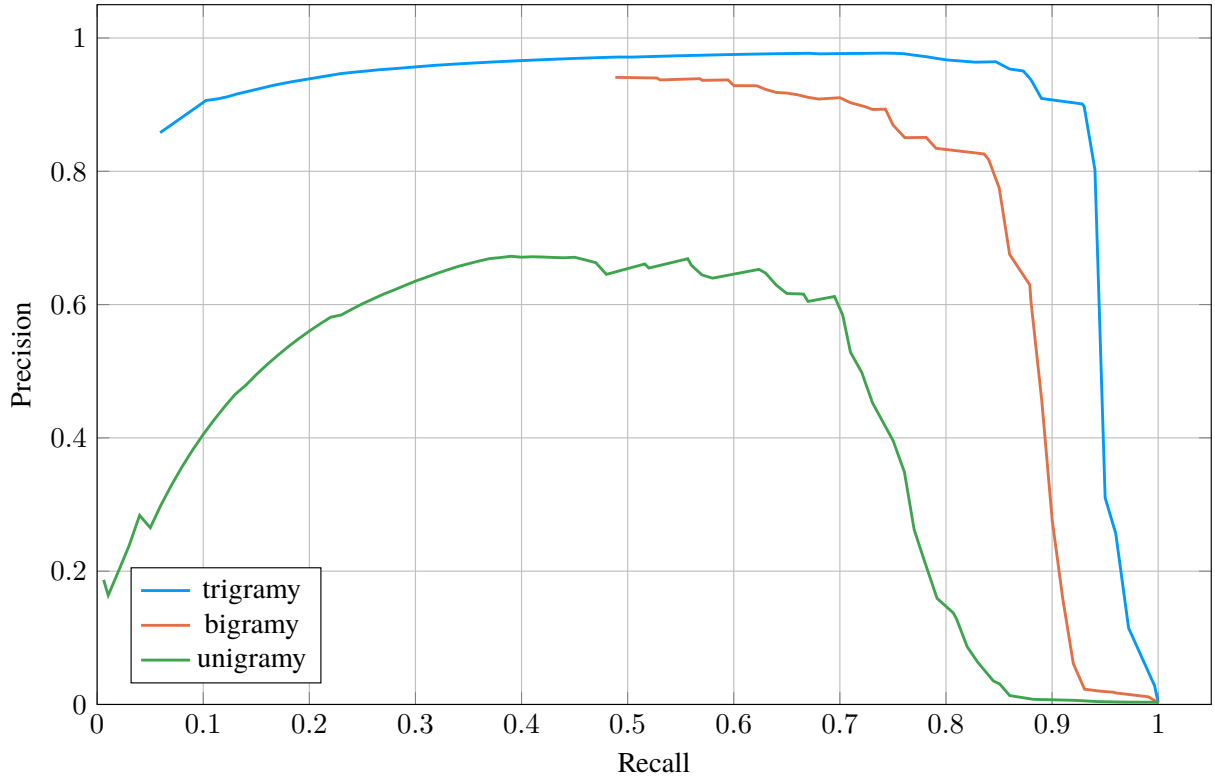
Require: $input$ ▷ Řetězec, ze kterého bude vektor příznaků generován
Require: $length$ ▷ Délka vektoru příznaků
Require: n ▷ Velikost příznaků

$feature\_vector \leftarrow$ ZERO_VECTOR$(length)$
for $i \in$ NGRAMS$(input, n)$ **do** ▷ Funkce vracející všechna podslova délky n
  $hash \leftarrow$ HASH(i) ▷ Standardní hash funkce jazyka Julia
  $index \leftarrow hash \mod length$
  $feature\_vector\,[index] \leftarrow feature\_vector\,[index] + 1$
end for

Jako funkce ψ projektující tokeny adresy URL do vektorů příznaků byla použita funkce popsaná algoritmem 5.1. Tato funkce každý token rozloží na posloupnost n**-gramů** (podslov délky n) a tyto n-gramy převede na indexy odpovídající pozicím ve vektoru příznaků. Vektor příznaků je pak vektorem četnosti výskytů jednotlivých trigramů. Tento algoritmus je nastaven dvěma parametry, a sice délkou vektoru příznaků (využívanou jako modulo při převodu n-gramů na indexy) a délkou podslov n. Všechny ostatní parametry byly ponechány na hodnotách z kapitoly 5.1.

Pro srovnání vlivu hodnoty n byly naučeny tři rozdílné modely využívající unigramy ($n = 1$), bigramy

($n = 2$) a trigramy ($n = 3$). Všechny tyto modely mají délku vektoru příznaků 2053. Na obrázku 5.3 jsou srovnány PR křivky pro tyto tři modely.

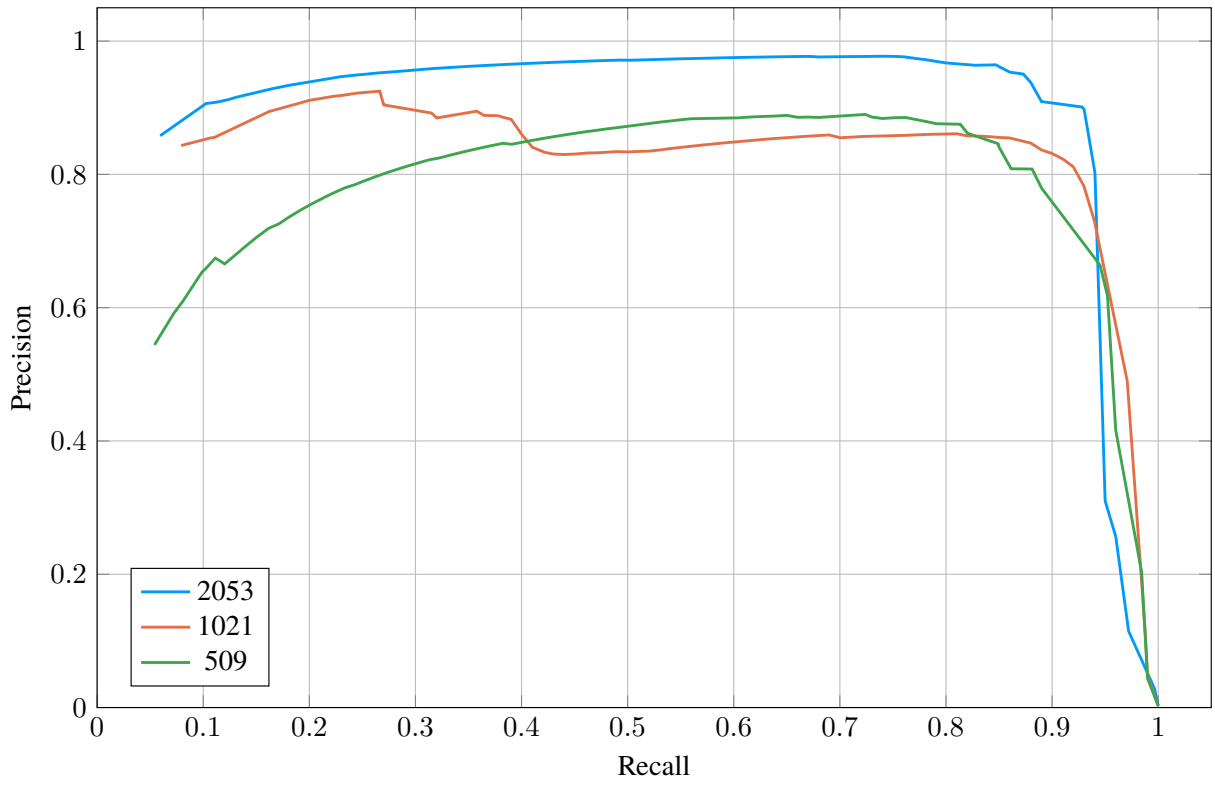


Graf 5.3: Srovnání PR křivek pro různé velikosti příznaků

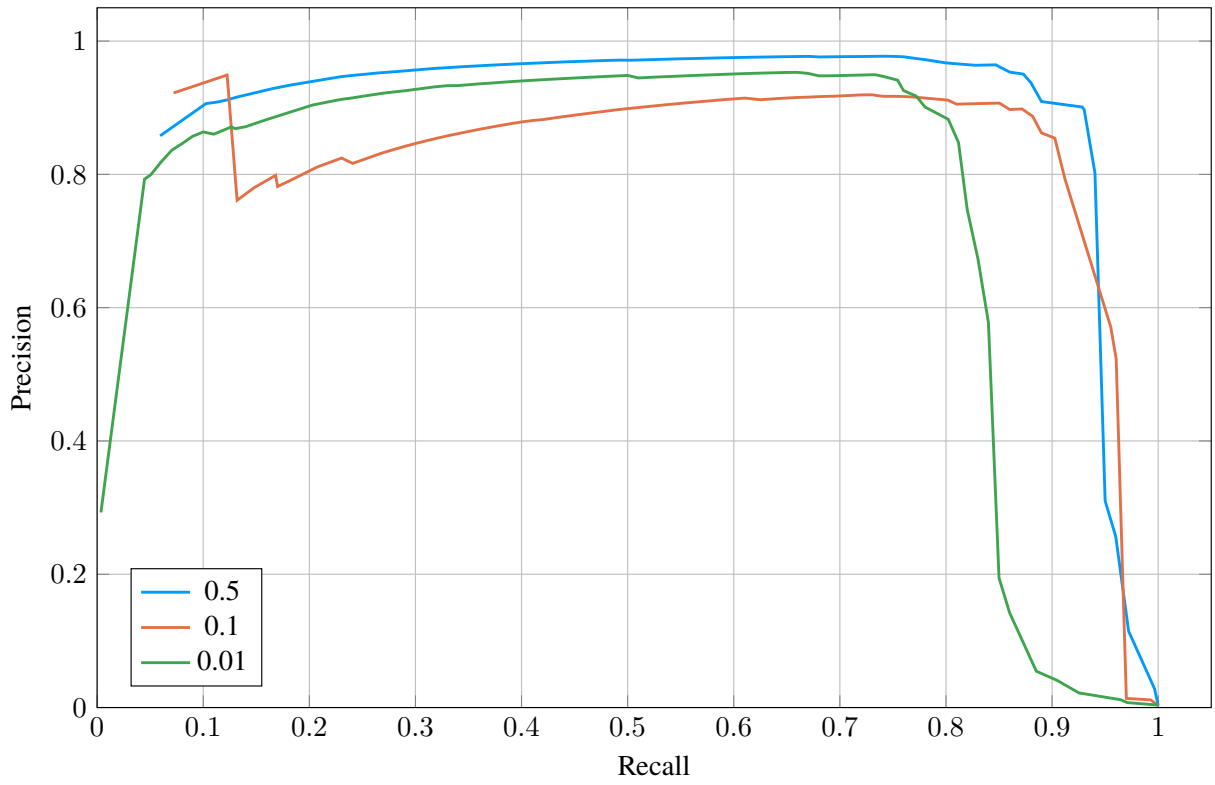
Pro srovnání vlivu délky vektoru příznaků byly naučeny modely s délkou 509, 1021 a 2053. Všechny tyto modely využívají trigramy. Na obrázku 5.4 jsou srovnány PR křivky těchto modelů.

5.4 Srovnání vlivu váhy na pozitivních taškách

Při trénování umělé neuronové sítě lze pozitivním a negativním taškám přiřadit různé váhy, které odpovídají tomu, jak nežádoucí jsou falešně negativní a falešně pozitivní odhady. Výchozí váha pozitivní třídy (tj. 0.5) neodpovídá souboru dat představenému v kapitole 1.3, kde pozitivních vzorků je výrazně méně, přestože cílem je právě jejich detekce. Proto byly vyzkoušeny modely s váhou (pozitivní třídy) 0.5, 0.1 a 0.01 se všemi ostatními parametry ponechanými na hodnotách z kapitoly 5.1. PR křivky těchto modelů jsou srovnány na obrázku 5.5.



Graf 5.4: Srovnání PR křivek pro různé počty příznaků



Graf 5.5: Srovnání PR křivek pro různé váhy pozitivní třídy

Závěr

V předložené práci byl navržen model pro automatickou detekci síťového provozu pocházejícího z aktivity malware, navrženy metody srovnání tohoto modelu s předchozími pracemi v oboru a provedeno experimentální vyhodnocení na reálných datech. Navržený model má srovnatelnou přesnost s nejlepším předchozím modelem, avšak výrazně vyšší odezvu. Tím byl dosažen a dokonce překonán vytyčený cíl práce, tj. aby navržený model měl alespoň stejnou kvalitu jako předchozí modely, využívající ručně vytvořeného seznamu příznaků. Navržený klasifikátor umožňuje detekci nežádoucího software s přesností i odezvou přesahující 90%. Bylo tedy ukázáno, že přístup pomocí multi-instančního učení je v praxi využitelnou alternativou ke klasickému přístupu a poskytuje velmi dobré výsledky.

Jako možný další postup k vylepšení navrženého modelu se jeví opětovná aplikace mutli-instančního učení na tuto úlohu, výsledkem které by byl klasifikátor, který určuje přítomnost nežádoucího software na úrovni klientů (narozdíl od současného modelu, klasifikujícího na úrovni síťových spojení). Dalším možným krokem je opětovné použití multi-instančního učení na tento model, tentokrát na úrovni síťových toků, to jest sekvencí síťových spojení mířících ke stejné destinaci. Výsledný model by byl tvořen čtyřmi vnořenými multi-instančními úlohami. Jako další možné vylepšení navrženého modelu se jeví využití některých dalších polí HTTP hlavičky či techniky zrychlující proces učení. Mezi ně patří například využití přenosových funkcí obsahujících šum (srov. Gulcehre et al., 2016), sebe-normalizujících neuronových sítí (srov. Klambauer et al., 2017) či využití učení pomocí syntetických gradientů (srov. Jaderberg et al., 2016 a Czarnecki et al., 2017). Mezi další možná rozšíření patří využití navženého přístupu při *semi-supervised* či *one-shot* učení.

Seznam obrázků

Seznam grafů

Seznam tabulek

Seznam algoritmů

Seznam literatury

AMORES, Jaume, 2013. Multiple instance classification: Review, taxonomy and comparative study. *Artificial Intelligence* [online]. Roč. 201, s. 81–105 [cit. 2017-05-31]. ISSN 0004-3702. Dostupné z DOI: 10.1016/j.artint.2013.06.003.

ANDREWS, Stuart; TSOCHANTARIDIS, Ioannis; HOFMANN, Thomas, 2003. Support vector machines for multiple-instance learning. In: *Advances in neural information processing systems* [online], s. 577–584 [cit. 2017-07-04]. Dostupné z: http://papers.nips.cc/paper/2232-support-vector-machines-for-multiple-instance-learning.pdf.

BERNERS-LEE, Tim; MASINTER, Larry; MCCAHILL, Mark P., 1994. *Uniform Resource Locators (URL)* [online] [cit. 2017-06-05]. Dostupné z: https://tools.ietf.org/html/rfc1738. CERN.

CAUCHY, Augustin, 1847. Méthode générale pour la résolution des systemes d'équations simultanées. *Comp. Rend. Sci. Paris* [online]. Roč. 25, č. 1847, s. 536–538 [cit. 2017-07-04]. Dostupné z: https://www.cs.xu.edu/math/Sources/Cauchy/Orbits/1847%20CR%20536(383).pdf.

2014. *Cisco 2014 Annual Security Report*. Dostupné také z: https://www.cisco.com/web/offer/gist_ty2_asset/Cisco_2014_ASR.pdf. Cisco Systems, Inc.

CZARNECKI, Wojciech Marian; ŚWIRSZCZ, Grzegorz; JADERBERG, Max; OSINDERO, Simon; VINYALS, Oriol; KAVUKCUOGLU, Koray, 2017. Understanding Synthetic Gradients and Decoupled Neural Interfaces. *arXiv:1703.00522 [cs]* [online] [cit. 2017-03-27]. Dostupné z arXiv: 1703.00522.

DASARATHY, Belur V., 1991. *Nearest neighbor (NN) norms: nn pattern classification techniques*. IEEE Computer Society Press. ISBN 978-0-8186-5930-0. Google-Books-ID: k2dQAAAAMAAJ.

DEMPSTER, A. P.; LAIRD, N. M.; RUBIN, D. B., 1977. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)* [online]. Roč. 39, č. 1, s. 1–38 [cit. 2017-07-01]. ISSN 0035-9246. Dostupné z: http://www.jstor.org/stable/2984875.

DIETTERICH, Thomas G.; LATHROP, Richard H.; LOZANO-PÉREZ, Tomás, 1997. Solving the multiple instance problem with axis-parallel rectangles. *Artificial Intelligence* [online]. Roč. 89, č. 1, s. 31–71 [cit. 2017-05-31]. ISSN 0004-3702. Dostupné z DOI: 10.1016/S0004-3702(96)00034-3.

DUCHI, John; HAZAN, Elad; SINGER, Yoram, 2011. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *Journal of Machine Learning Research* [online]. Roč. 12, s. 2121–2159 [cit. 2017-07-04]. ISSN ISSN 1533-7928. Dostupné z: http://www.jmlr.org/papers/v12/duchi11a.html.

FOULDS, James Richard, 2008. *Learning Instance Weights in Multi-Instance Learning* [online] [cit. 2017-07-06]. Dostupné z: http://researchcommons.waikato.ac.nz/handle/10289/2460. Thesis. The University of Waikato.

GÄRTNER, Thomas; FLACH, Peter A.; KOWALCZYK, Adam; SMOLA, Alexander J., 2002. Multi-instance kernels. In: *ICML* [online]. Sv. 2, s. 179–186 [cit. 2017-07-04]. Dostupné z: http://sci2s.ugr.es/keel/pdf/algorithm/congreso/2002-Gartner-ICML.pdf.

GOODFELLOW, Ian J.; WARDE-FARLEY, David; MIRZA, Mehdi; COURVILLE, Aaron; BENGIO, Yoshua, 2013. Maxout networks. *arXiv preprint arXiv:1302.4389* [online] [cit. 2017-07-02]. Dostupné z: http://www.jmlr.org/proceedings/papers/v28/goodfellow13.pdf.

GOODFELLOW, Ian; BENGIO, Yoshua; COURVILLE, Aaron, 2016. *Deep Learning*. Cambridge, Massachusetts: The MIT Press. ISBN 978-0-262-03561-3.

GULCEHRE, Caglar; MOCZULSKI, Marcin; DENIL, Misha; BENGIO, Yoshua, 2016. Noisy Activation Functions. *arXiv:1603.00391 [cs, stat]* [online] [cit. 2017-03-06]. Dostupné z arXiv: 1603.00391.

HAUSSLER, David, 1999. *Convolution kernels on discrete structures* [online] [cit. 2017-07-04]. Dostupné z: https://www.soe.ucsc.edu/sites/default/files/technical-reports/UCSC-CRL-99-10.pdf. Technical report, Department of Computer Science, University of California at Santa Cruz.

CHEN, Yixin; BI, Jinbo; WANG, J. Z., 2006. MILES: Multiple-Instance Learning via Embedded Instance Selection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. Roč. 28, č. 12, s. 1931–1947. ISSN 0162-8828. Dostupné z DOI: 10.1109/TPAMI.2006.248.

CHEN, Yixin; WANG, James Z., 2004. Image Categorization by Learning and Reasoning with Regions. *Journal of Machine Learning Research* [online]. Roč. 5, s. 913–939 [cit. 2017-07-06]. ISSN ISSN 1533-7928. Dostupné z: http://www.jmlr.org/papers/v5/chen04a.html.

CHEPLYGINA, Veronika; TAX, David M. J.; LOOG, Marco, 2015. Multiple instance learning with bag dissimilarities. *Pattern Recognition* [online]. Roč. 48, č. 1, s. 264–275 [cit. 2017-07-04]. ISSN 0031-3203. Dostupné z DOI: 10.1016/j.patcog.2014.07.022.

JADERBERG, Max; CZARNECKI, Wojciech Marian; OSINDERO, Simon; VINYALS, Oriol; GRAVES, Alex; KAVUKCUOGLU, Koray, 2016. Decoupled Neural Interfaces using Synthetic Gradients. *arXiv preprint arXiv:1608.05343* [online] [cit. 2017-07-06]. Dostupné z: https://arxiv.org/abs/1608.05343.

KINGMA, Diederik P.; BA, Jimmy, 2014. Adam: A Method for Stochastic Optimization. *arXiv:1412.6980 [cs]* [online] [cit. 2017-04-19]. Dostupné z arXiv: 1412.6980.

KLAMBAUER, Günter; UNTERTHINER, Thomas; MAYR, Andreas; HOCHREITER, Sepp, 2017. Self-Normalizing Neural Networks. *arXiv preprint arXiv:1706.02515* [online] [cit. 2017-07-06]. Dostupné z: https://arxiv.org/abs/1706.02515.

KNUTH, Donald E., 1968. *The Art of Computer Programming*. Addison-Wesley. ISBN 0-201-03801-3.

KWOK, James T.; CHEUNG, Pak-Ming, 2007. Marginalized Multi-Instance Kernels. In: *IJCAI* [online]. Sv. 7, s. 901–906 [cit. 2017-07-04]. Dostupné z: http://www.aaai.org/Papers/IJCAI/2007/IJCAI07-145.pdf.

LEEUWEN, David van, 2016. *ROCAnalysis.jl: Receiver Operating Characteristics and functions for evaluation probabilistic binary classifiers* [online] [cit. 2017-07-03]. Dostupné z: https://github.com/davidavdav/ROCAnalysis.jl. original-date: 2014-03-13T08:23:30Z.

LIN, Dahua, 2017. *MLBase.jl: A set of functions to support the development of machine learning algorithms* [online]. Julia Statistics [cit. 2017-07-03]. Dostupné z: https://github.com/JuliaStats/MLBase.jl. original-date: 2013-02-10T15:50:23Z.

MAC NAMEE, B.; CUNNINGHAM, P.; BYRNE, S.; CORRIGAN, O. I., 2002. The problem of bias in training data in regression problems in medical decision support. *Artificial Intelligence in Medicine* [online]. Roč. 24, č. 1, s. 51–70 [cit. 2017-07-07]. ISSN 0933-3657. Dostupné z DOI: 10.1016/S0933-3657(01)00092-6.

MACHLICA, Lukas; BARTOS, Karel; SOFKA, Michal, 2017. Learning detectors of malicious web requests for intrusion detection in network traffic. *arXiv:1702.02530 [cs, stat]* [online] [cit. 2017-06-29]. Dostupné z arXiv: 1702.02530.

MARON, Oded; LOZANO-PÉREZ, Tomás, 1998. A framework for multiple-instance learning. In: *Advances in neural information processing systems* [online], s. 570–576 [cit. 2017-07-01]. Dostupné z: http://papers.nips.cc/paper/1346-a-framework-for-multiple-instance-learning.pdf.

MARTIN, A.; DODDINGTON, G.; KAMM, T.; ORDOWSKI, M.; PRZYBOCKI, M., 1997. *The DET Curve in Assessment of Detection Task Performance* [online] [cit. 2017-07-02]. Dostupné z: http://www.dtic.mil/docs/citations/ADA530509. National Institute of Standards a Technology.

MOCKAPETRIS, Paul V., 1987. *Domain names - concepts and facilities* [online] [cit. 2017-06-24]. Dostupné z: https://tools.ietf.org/html/rfc1034. Network Working Group.

MUANDET, Krikamol; FUKUMIZU, Kenji; DINUZZO, Francesco; SCHÖLKOPF, Bernhard, 2012. Learning from Distributions via Support Measure Machines. In: *Advances in neural information processing systems* [online], s. 10–18 [cit. 2017-06-29]. Dostupné z: http://papers.nips.cc/paper/4825-learning-from-distributions-via-support-measure-machines.

PEVNY, Tomas; SOMOL, Petr, 2016a. Discriminative Models for Multi-instance Problems with Tree Structure. In: *Proceedings of the 2016 ACM Workshop on Artificial Intelligence and Security* [online]. New York, NY, USA: ACM, s. 83–91 [cit. 2017-03-01]. AISec '16. ISBN 978-1-4503-4573-6. Dostupné z DOI: 10.1145/2996758.2996761.

PEVNY, Tomas; SOMOL, Petr, 2016b. Using Neural Network Formalism to Solve Multiple-Instance Problems. *arXiv:1609.07257 [cs, stat]* [online] [cit. 2017-03-01]. Dostupné z arXiv: 1609.07257.

ROELANTS, Peter, 2017. *How to implement a neural network Intermezzo 2* [Peter's notes] [online] [cit. 2017-07-08]. Dostupné z: https://peterroelants.github.io/posts/neural_network_implementation_intermezzo02/.

TIELEMAN, Tijmen; HINTON, Geoffrey, 2012. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*. Roč. 4, č. 2, s. 26–31.

VEJMELKA, Martin, 2017. *Fighting malicious software with machinel learning* [Přednáška na konferenci]. Dostupné také z: https://www.youtube.com/watch?v=ZL-j8p718G8. Machine Learning Prague.

VINYALS, Oriol; BENGIO, Samy; KUDLUR, Manjunath, 2015. Order Matters: Sequence to sequence for sets. *arXiv:1511.06391 [cs, stat]* [online] [cit. 2017-04-19]. Dostupné z arXiv: 1511.06391.

WANG, Jun; ZUCKER, Jean-Daniel, 2000. Solving Multiple-Instance Problem: A Lazy Learning Approach. In: LANGLEY, Pat (ed.). *Proceedings of the Seventeenth International Conference on Machine Learning* [online]. Stanford University, Stanford, CA, USA: Morgan Kaufmann, s. 1119–1125 [cit. 2017-07-01]. Dostupné z: http://cogprints.org/2124/.

ZEA, Diego Javier, 2015. *ROC.jl: Receiver Operating Characteristic (ROC) Curve for Julia Language* [online] [cit. 2017-07-03]. Dostupné z: https://github.com/diegozea/ROC.jl. original-date: 2014-04-26T08:56:44Z.

ZEILER, Matthew D., 2012. ADADELTA: An Adaptive Learning Rate Method. *arXiv:1212.5701 [cs]* [online] [cit. 2017-07-04]. Dostupné z arXiv: 1212.5701.

ZHANG, Cha; PLATT, John C.; VIOLA, Paul A., 2006. Multiple instance boosting for object detection. In: *Advances in neural information processing systems* [online], s. 1417–1424 [cit. 2017-07-04]. Dostupné z: http://papers.nips.cc/paper/2926-multiple-instance-boosting-for-object-detection.pdf.

ZHANG, Min-Ling; ZHOU, Zhi-Hua, 2009. Multi-instance clustering with applications to multi-instance prediction. *Applied Intelligence* [online]. Vol. 31, no. 1, s. 47–68 [cit. 2017-07-06]. ISSN 0924-669X, 1573-7497. ISSN 0924-669X, 1573-7497. Dostupné z DOI: 10.1007/s10489-007-0111-x.

ZHANG, Qi; GOLDMAN, Sally A., 2002. EM-DD: An Improved Multiple-Instance Learning Technique. In: *Advances in neural information processing systems* [online], s. 1073–1080 [cit. 2017-07-01]. Dostupné z: http://papers.nips.cc/paper/1959-em-dd-an-improved-multiple-instance-learning-technique.pdf.

ZHOU, Zhi-Hua; SUN, Yu-Yin; LI, Yu-Feng, 2009. Multi-instance Learning by Treating Instances As non-I.I.D. Samples. In: *Proceedings of the 26th Annual International Conference on Machine Learning* [online]. New York, NY, USA: ACM, s. 1249–1256 [cit. 2017-07-06]. ICML '09. ISBN 978-1-60558-516-1. Dostupné z DOI: 10.1145/1553374.1553534.

ZHOU, Zhi-Hua; ZHANG, Min-Ling, 2002. Neural Networks for Multi-Instance Learning. In: *Proceedings of the International Conference on Intelligent Information Technology, Beijing, China* [online], s. 455–459 [cit. 2017-06-26]. Dostupné z: http://cs.nju.edu.cn/zhouzh/zhouzh.files/publication/techrep02.pdf.

ZHU, Ji; ROSSET, Saharon; TIBSHIRANI, Robert; HASTIE, Trevor J., 2004. 1-norm support vector machines. In: *Advances in neural information processing systems* [online], s. 49–56 [cit. 2017-07-01]. Dostupné z: http://papers.nips.cc/paper/2450-1-norm-support-vector-machines.pdf.