Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Bezpečnosť strojového učenia

Bakalárska práca

2020
Erik Bíly

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Bezpečnosť strojového učenia

Bakalárska práca

Študijný program:	Aplikovaná informatika
Študijný odbor:	Aplikovaná informatika
Školiace pracovisko:	Katedra informatiky FMFI
Školiteľ:	RNDr. Richard Ostertág, PhD.

Bratislava, 2020
Erik Bíly

81871094

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Erik Bíly
Študijný program: aplikovaná informatika (Jednoodborové štúdium, bakalársky I. st., denná forma)
Študijný odbor: informatika
Typ záverečnej práce: bakalárska
Jazyk záverečnej práce: slovenský
Sekundárny jazyk: anglický

Názov: Bezpečnosť strojového učenia
Security of Machine Learning

Anotácia: Vo veľa nových produktoch sa dnes používa strojové učenie. Rozpoznávajú sa hlasové povely, interpretuje sa zmysel obrázkov, sumarizuje sa význam textu. Bezpečnosť strojového učenie je relatívne nová oblasť, ktorá sa zaoberá hľadaním jeho rôznych zraniteľností, napríklad nájdením podobných vstupov, ktoré sú ale diametrálne odlišne klasifikované.
Cieľom tejto práce bude spraviť prehľad a klasifikáciu rôznych útokov na strojové učenie a následne implementovať vybrané z nich a reálne otestovať ich použiteľnosť v praxi. Ďalším cieľom práce je popísať metódy detekcie útokov a možnosti ochrany pred nimi, opäť spolu s implementáciou vybraných metód.

Vedúci: RNDr. Richard Ostertág, PhD.
Katedra: FMFI.KI - Katedra informatiky
Vedúci katedry: prof. RNDr. Martin Škoviera, PhD.

Spôsob sprístupnenia elektronickej verzie práce:
bez obmedzenia

Dátum zadania: 05.11.2019

Dátum schválenia: 06.11.2019 doc. RNDr. Damas Gruska, PhD.
garant študijného programu

.....
študent

.....
vedúci práce

Abstrakt

Posledné roky strojové učenie jednoznačne víťazí vo výkone nad klasickými algoritmami a v niektorých prípadoch aj nad ľuďmi. Kvôli tomuto sa neustále rozširuje využitie strojového učenia do viacerých produktov. V niektorých využitiach ako je napríklad odporúčanie reklám uživateľom sociálnych sietí na bezpečnostni až tak veľmi nezáleží, ale nájdu sa aj iné využitia strojového učenia ako je napríklad biometria alebo autonómne vozidlá, kde je bezpečnosť systému kritická. Bezpečnosť strojového učenia je pomerne mladá oblasť a častokrát sa na ňu v produktoch zabúda. Táto práca sa venuje najrôznejším útokom na strojové učenie, niektoré sú demonštrované na neurónovej sieti a taktiež sa zaoberá obranným mechanizmom proti týmto útokom.

Kľúčové slová: strojové učenie, bezpečnosť

Abstract

Recently machine learning has defeated in performance many classic algorithms and even people in some tasks. Because of this, machine learning is being used in more and more products. In products like advertisement recommendation to social media users, the security of these products might not be as essential, but there are other uses of machine learning like autonomous vehicles or biometrics, where the security is paramount. Security of machine learning is quite young field and therefore the security of such products is often overlooked. This paper goes through various attacks on machine learning, some of which are demonstrated on neural network, but also looks into defense mechanisms to counter some attacks.

Keywords: machine learning, security

Obsah

Zoznam obrázkov

Úvod

Strojové učenie je oblasť informatiky, ktorá sa venuje algoritmom ako sú napríklad rozhodovacie stromy, podporné vektorové stroje a neurónové siete. Spoločným cieľom všetkých týchto algoritmov je riešiť nejaký problém, ale bez toho, aby im bolo explicitne povedané ako majú daný problem počítať. Algoritmy sa teda učia samé iba na základe veľkého množstva trénovacích dát.

Prvé využitia strojového učenia sa datujú už k päťdesiatym rokom minulého storočia, no až nedávno sa začalo využívať vo veľkom. Môže za to najmä neustále zvyšovanie výpočtového výkonu počítačov a pokrok vo vývoji algoritmov a to hlavne v neurónových sieťach.

Neurónové siete sa v súčastnosti využívajú v najrôzneších oblastiach od rozpoznávania obrázkov[15], spracovávanie ľudského hlasu[9], prekladu jazykov[1] až po umelú inteligenciu v niektorých hrách[2] a autonómne vozidlá[3].

Tak ako to častokrát býva s novými technológiami, že okrem benefitov, ktoré tieto technológie prinášajú, prichádzajú aj nové riziká, tak je to aj so strojovým učením. Dalo by sa povedať, že strojové učenie je ešte viac vystavené nebezpečenstvu ako klasické informačné systémy, pretože trénovacie dáta v mnohých prípadoch pochádzajú z celého sveta. Takéto zbieranie globálnych dát býva niekedy kvôli nedostatku zdrojov, ale niekedy je to práveže cieľom. Potencionálny nepriatelia majú teda možnosť zaútočiť na systém ešte pred tým ako bude natrénovaný a zavedený do prevádzky. Okrem toho existuje viacero možností ako zaútočiť priamočiarejšie na strojové učenie, ktoré je už v prevádzke.

Táto práca sa venuje definovaniu, vysvetlovaniu a implementácií najčastejších a najnebezpečnejších útokov a taktiež obranným metódam proti týmto útokom. Útoky prezentujem na neurónovej sieti určenej na rozpoznávanie obrázkov písaných číslic, pretože takýto model je celkom jednoduchý na pochopenie, rýchlo sa natrénuje a výsledok je dobre viditelný na týchto obrázkoch. Skoro každý útok je univerzálny a funguje na všetkých algoritmoch strojového učenia, jediný rozdiel je potom v ich implementácií.

Cieľom tejto práce je ukázať, že aj takáto dokonalá technológia, ktorá dokáže prekonať ľudí a iné klasické algoritmy v rôznych oblastiach má svoje chyby. Tieto bezpečnostné nedostatky by mal poznať každý programátor, ktorý sa čo len okrajovo venuje

strojovému učeniu, aby im vedeľ zabrániť v prípade, že sa to dá. V niektoých prípadoch, keď sa útokom zabrániť nedá, mal by to mať programátor na vedomí a na riešenie problému by mal asi využiť inú technológiu ako strojové učenie.

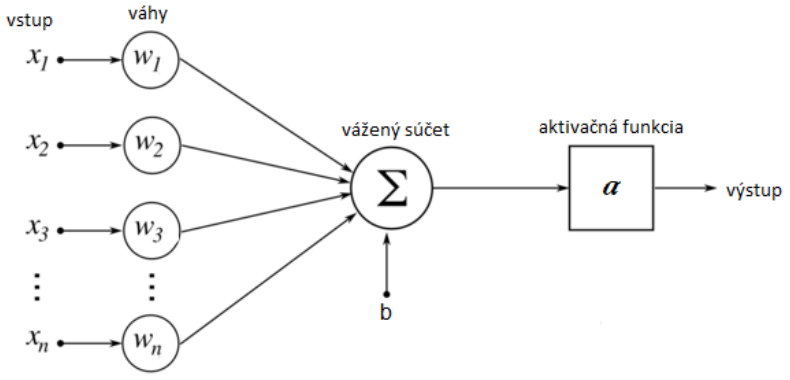
Kapitola 1

Podklad

Strojové učenie je veľmi rozsiahla oblasť umelej inteligencie, pod túto oblasť spáda veľa algoritmov ako napríklad supervised a unsupervised learning a tieto algoritmy sú implementované rôznymi modelmi. Populárne modely sú napríklad neurónove siete, support vector machines a decision trees. Z týchto modelov som najlepšie oboznámený s neurónovými sieťami (NS) a preto bude väčšina útokov v mojej práci predvedená práve na nich. V tejto kapitole stručne popíšem ako NS fungujú a vysvetlím základné pojmy potrebné na pochopenie útokov. Taktiež vymenujem technológie, ktoré využívam na prácu so strojovým učením a spomeniem niektore doterajšie práce venujúce sa bezpečnosti strojového učenia.

1.1 Definície

Tu vysvetlím základy fungovania neurónových sieti. Niektoré útoky spočívajú priamo v tom, že ručne meníme štruktúru NS alebo jej hodnoty (napríklad váhy) na pozmenenie funkcionality.



Obr. 1.1: Neurón

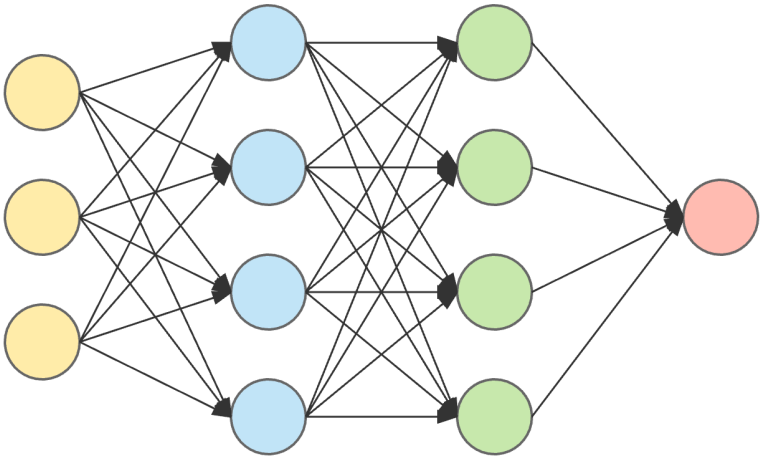
Neurón, pre lepšiu predstavu ilustrovaný na obrázku 1.1, je základná stavebná jednotka neurónových sieti. Dá sa na neho pozerať ako na jednoduchú funkciu, ktorá

dostane niekoľko vstupov $x_1 ... x_n$ a vypočíta výstup $f(\vec{x})$. Táto funkcia vyzerá nasledovne:

$$f(\vec{x}) = a(w_1 x_1 + ... + w_n x_n + b)$$

$w_1 ... w_n$ predstavujú v rovnici **váhy** neurónu k príslušným vstupom a tie hovoria o tom, ako veľmi dôležitý je daný vstup. K tomuto sa môže pripočítavať **bias** (b), ktorý dokáže posunúť funkciu a tým docieliť úspešnejšie učenie neurónov. Vážený súčet plus bias môžu dosahovať ľubovoľnú hodnotu od $(-\infty, \infty)$ a teda kvôli jednoznačnosti, či je neurón aktivovaný alebo nie sa na tento výsledok ešte aplikuje **aktivačná funkcia** (a), ktorá usmerňuje výstup neurónu. Jedna z často používaných aktivačných funkcií je sigmoid, ktorú môžeme vidieť na obrázku 1.3

Spájaním neurónov do jedného celku tak, že výstupy niektorých neurónov sa stávajú vstupmi pre iné neuróny vzniká **neurónova sieť**. Skupinu neurónov na rovnakej úrovni nazývame **vrstva**. Prvá vrstva každej siete sa nazýva **vstupná vrstva** (input layer), posledná vrstva je **výstupná vrstva** (output layer) a všetky ostatné voláme **skryté vrstvy** (hidden layers). Na obrázku 1.2 sa nachádza jednoduchá NS zložená zo štyroch vrstiev.



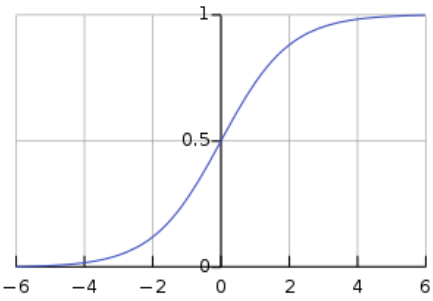
Obr. 1.2: Na tomto obrázku sa nachádza jednoduchá neurónová sieť. Neuróny označené žltou farbou patria do vstupnej vrstvy, červený neurón tvorí výstupnú vrstvu a modré a zelené tvoria dve skryté vrstvy.

Vrstvy neurónov sa kategorizujú nielen podľa pozície v sieti, ale aj podľa spôsobu prepojenia neurónov medzi sebou a teda ich funkčnosti. Najzákladnejší typ sú **plne prepojené** (fully connected) alebo niekedy nazývané aj dense vrstvy, kde sú všetky neuróny tejto vrstvy priamo spojené s každým neurónom na ďalšej vrstve. V NS na

obrázku 1.2 sú všetky vrstvy plne prepojené.
Ďalšie veľmi populárne typy sú **konvolučné** (convolution) vrstvy a **pooling** vrstvy, ktoré sa využívajú najmä pri spracovaní obrazov. Neuróny v konvolučných vrstvach sú prepojené len na malú oblasť neurónov predošlej vrstvy, jednak aby sa zabránilo pretrénovaniu, ale hlavne kvôli rýchlosti výpočtov. Pooling vrstvy sa častokrát nachádzajú priamo za konvolučnou vrstvou. Tieto vrstvy redukujú dimenziu dát v sieti tým, že združujú nejakú množinu výstupov neurónov do jedného výstupu. Dva bežné spôsoby ako sa vyráta výstup z tejto vrstvy sú average pooling, kde sa vyráta priemerná hodnota z množiny a max pooling, kedy sa zoberie maximálna hodnota z množiny. Existuje ešte hromada iných typov vrstiev s rôznymi využitiami, tie v prípade potreby popíšem neskôr.

S týmito vedomostami by sme si vedeli zostaviť neurónovú sieť, ale bez toho aby sme ju natrénovali by fungovala úplne náhodne. **Trénovanie** NS je proces, pri ktorom sieť dostáva vstupy, ktoré sa vyhodnotia a váhy medzi vrstvami sa upravujú tak, aby sa pri každej iterácií zmenšovala **chyba** (čo je rozdiel medzi očakávaným výstupom a reálnym výstupom z NS). Toto je veľmi povrchná definícia a platná len pri učení s učiteľom (supervised learning), ale nebudem to ani rozvíjať, pretože to nie je cieľ tejto práce. Ak by niekoho zaujímalo podrobnejšie vysvetlenie, odporúčam Machine Learning Crash Course od Google, kde je trénovanie a iné pojmy dobre vysvetlené. Pri trénovaní modelov by sme mali dávať pozor na **overfitting** (pretrénovanie). To je situácia, kedy sa model naučí vyhodnocovať trénovacie dáta tak presne, že kvôli tomu bude horšie vyhodnocovať iné dáta v budúcnosti. Underfitting je opačný prípad, teda model nedostatočne presne vyhodnocuje trénovacie dáta napríklad kvôli zle nastavenym hyperparametrom pri trénovaní, alebo kvôli zlej architektúre siete. Ideálne sa snažíme docieliť niečo medzi underfitting a overfitting, teda model bude vyhodnocovať trénovacie dáta dosť presne a zároveň bude aj dobre generalizovať.

Útoky na softvér, v našom prípade na strojové učenie, je možné rozdeliť podľa množstva vedomostí, ktoré máme o softvére a udelených prístupových práv k nemu na white-box a black-box útoky.
Pri **White-box** útokoch, niekedy nazývaných aj clear-box alebo open-box, má útočník plný prístup k celému zdrojovému kódu, alebo v prípade, že sa pracuje už s natrénovaným modelom, útočník vidí architektúru a všetky dáta potrebné na fungovanie tohto modelu. V tomto prípade je možná statická analýza kódu/modelu, ktorá častokrát zjednoduší útoky. V prípade, že máme k modelu aj plné prístupové práva, teda môžeme ho ľubovoľne upravovať, vznikajú možnosti na nové typy útokov.
Black-box situácia je taká, že útočník nemá navyše žiadne informácie o systéme. Stále ale môže posielať vstupy do modelu a prijímať jeho výstupy. V tomto prípade sú útoky častokrát nejako automatizované.



Obr. 1.3: Logistická sigmoida

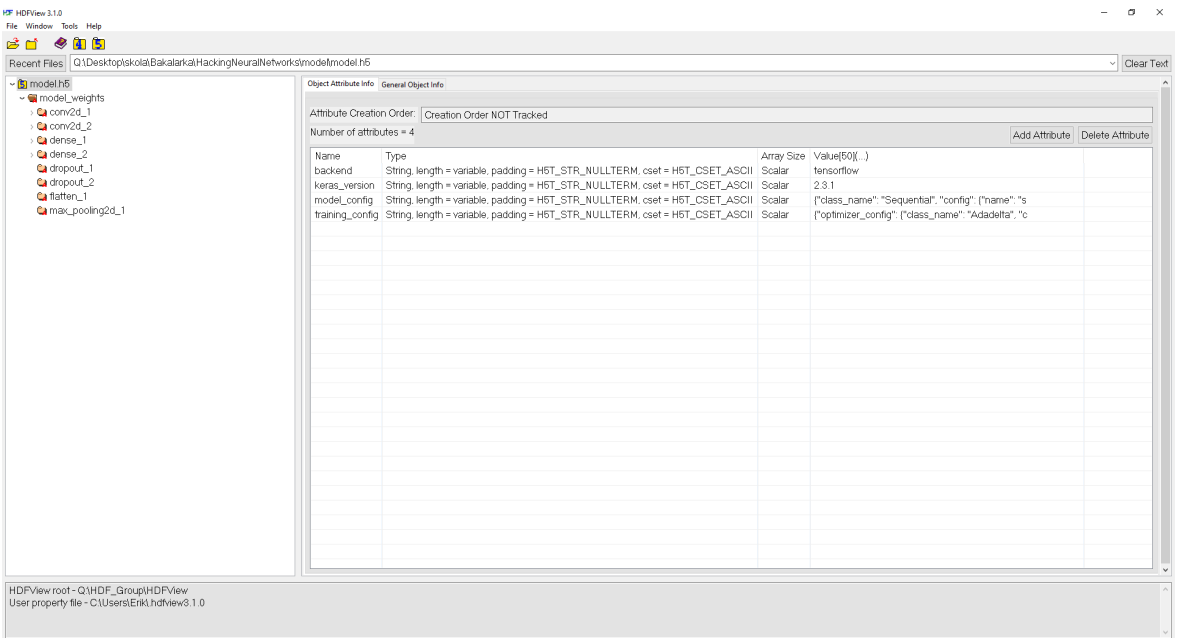
1.2 Využité technológie

Neurónove siete sa dajú implementovať v rôznych jazykoch pomocou rôznych knižníc. Medzi najpopulárnejšie jazyky v tejto oblasti patria Python, C++ a Java, ale nájdu sa aj knižnice pre Haskell (knižnica neural) a iné. Ja som sa rozhodol pre Python a knižnicu Keras, pretože väčšina článkov a online návodov sa venuje práve im.

- **Python a pip.** Python je jeden z najpopulárnejších objektovo-orientovaných programovacích jazykov. Pomocou pip-u je možné nainštalovať rôzne python balíčky.
- **Numpy (NP).** Polia a matice sú základom pri trénovaní neurónových sieti, ale aj neskôr pri ich použití v praxi. Preto používam Numpy, ktorý implementuje mnohé funkcie na maticiach. Je to open-source balík pre Python.
- **TensorFlow (TF)** je open-source API (Application programming interface) od Google, ktorý má okrem iného aj využitie aj v strojovom učení. Priamo s ním nepracujem, ale slúži ako backend pre Keras.
- **Keras** je tiež open-source API pre prácu s neurónovými sieťami ale je ešte o úroveň vyššie. Je schopný využiť aj iné backendy ako TF, ale ja budem používať práve túto kombináciu. Vytváranie modelov NS pomocou Kerasu je veľmi jednoduché a ich trénovanie dokáže byť pomerne rýchle, lebo okrem CPU (Central processing unit) môžeme model trénovať aj na niektorých GPU (Graphics processing unit), alebo na špeciálnom hardvéri TPU (Tensor Processing Unit) od Google.
- **Jupyter Notebook** poskytuje interaktívne prostredie pre programovanie v Pythone. Rozdelovanie programu do viacerých častí - buniek, ktoré je možné samostatne vykonať je obrovskou výhodou pri programoch ako trénovanie modelov strojového učenia. Trénovanie modelu môžeme oddeliť do samostatnej bunky, ktorú vykonáme iba raz a v ďalších bunkých môžeme s modelom pracovať.

- **Google Colaboratory** je cloudová verzia Jupyter Notebooku. Program sa vykonáva na TPU na serveroch od Google, čo v mojom prípade až 12-krát urýchluje trénovanie modelov.

- **HDFView** využívam na prehliadanie a upravovanie uložených modelov neurónových sieti vo formáte HDF5 (Hierarchical Data Format). Na obrázku 1.4 je konvolučná neurónová sieť zobrazená v programe HDFView. Jednotlivé vrstvy sa dajú rozkliknúť pre viac informácií.



Obr. 1.4: Konvolučná neurónova sieť rozpoznávajúca ručne písané číslice, natrénovaná na MNIST datasete s presnosťou približne 99% otvorená v programe HDFView

Kapitola 2

Evasion (Adversarial examples)

Aj napriek tomu, že oblasť bezpečnosť strojového učenia je pomerne mladá, už existuje nespočetne veľa prác venujúcich sa bezpečnosti strojového učenia ako takého a aj konkrétnym útokom, ich implementácií a ochrane proti nim. V tejto kapitole popíšem niektoré z nich. Tento zoznam prác nie je hotový, vo finálnej verzii bakalárskej práce možno neuvediem všetky z týchto prác a na druhú stranu určite pribudne ešte mnoho prác zaoberajúcich sa inými útokmi.

Jedna z najpopulárnejších kategórií útokov na strojové učenie je **adversarial examples** (niekedy nazývaná aj Evasion), čo by sa do slovenčiny dalo preložiť ako "protikladné prípady", alebo "nepriateľské prípady". Nepriateľské prípady sú vstupy pre modely strojového učenia, ktoré útočník zámerne navrhol tak, aby tento model vypočítal iný výstup akoby sme od neho očakávali. Tiež sa to dá chápať ako optická ilúzia pre počítače. Na obrázku 2.1 je znázornené, ako takéto nepriateľské prípady môžu vyzerať pre model, ktorý klasifikuje obrázky do rôznych kategórií.

Christian Szegedy a jeho tím vývojárov sa ako prví venovali tejto téme v roku 2013. V ich práci [21] zaviedli pojem adversarial examples a snažili sa vysvetliť ich existenciu ([17][11] sú ďalšie dva články venujúce sa zdôvodneniu existencie adversarial examples). Taktiež zistili, že nepriateľské prípady vygenerované na oklamanie jednej neurónovej siete dokážu oklamať aj iné NS s odlišnou architektúrou a natrénované na iných dátach. Toto sa považuje za obrovskú dieru v bezpečnosti neurónových sieti, lebo útočník si môže vytvoriť a natrénovať vlastný model, generovať nepriateľské prípady pre tento jeho model a neskôr ich použiť na iné modely. Generovanie týchto prípadov je pomerne jednoduché vo white-box situáciach.
Využitie NS na úlohy ako samostatne jazdiace autá, identifikácia osôb podľa tváre a iných biometrických čŕt, filtrovanie spamov v emailoch a rôzne iné teda vôbec nie je také spoľahlivé, ako sme si mohli myslieť. V tejto štúdii[12] sa môžeme dočítať o rôznych metódach generovania nepriateľských prípadov a ich dopade na využitie stro-

Obr. 2.1: Nepriateľské prípady generované pre model AlexNet [21]. Vľavo je vstup, ktorý tento model klasifikoval správne a všetky obrázky v pravo boli klasifikované ako pštros dvojprstý. V strede je zobrazený farebný rozdiel medzi ľavým a pravým obrázkom.

jového učenia v reálnom svete, kde aplikácie dostávajú vstup priamo z kamier alebo iných senzorov.

Neurónové siete ale rozhodne nie sú jediným typom modelov strojového učenia, na ktoré sa dá uplatniť tento útok. Experimenty v nasledujúcom článku[4] ukazujú, že aj decision tree modely sú zraniteľné. Autori taktiež navrhli nový trénovací decision tree framework, ktorý testovali a ukázali, že tieto modely sú robustnejšie voči nepriateľským prípadom. V tejto práci z novembra 2019[13] je predstavená metóda ako útočiť na K-Nearest Neighbours modely.

Dobrou správou ale je, že už vznikajú aj rôzne obranné stratégie proti nepriateľským prípadom, no ešte žiadna z nich nie je univerzálne akceptovaná, pretože väčšinu stratégií sa podarilo nejako obísť onedlho na to. Toto sú niektoré z nich:

Nepriateľské trénovanie (Adversarial training) je stratégia, pri ktorej sa využívajú nepriateľské prípady pri trénovaní modelu na zníženie nesprávnej klasifikácie.

Autoencoder je typ NS, ktorý najskôr redukuje dimenzionalite vstupov tým, že vstup prevedie cez skryté vrstvy, ktoré majú menšiu dimenziu a potom sa z týchto informácií pokúsi vstup zrekonštruovať. Inak povedané je to skoro ako identické zobrazenie, ale keďže sa dáta najskôr kompresujú a potom rekonštruujú, spravidla tento proces odstráni hluk v dátach a teda ostanú len podstatné črty. Takto teda môže autoencoder pomôcť vymazať malinké zmeny vo vstupoch, kvôli ktorým model robí chybu. Na druhej strane, ak útočník vie o tomto mechanizme, nie je pre neho problém generovať nepriateľské prípady, ktoré počítajú aj s autoencoderom ako súčasť siete a teda ho dokážu obísť.

Medzi posledné dva prístupy, ktoré som našiel patria DeepSafe[8] a destilácia (distillation). Destilácia je metóda, ktorá dokáže podľa autorov tohto článku [14] redukovať

úspešnosť nepriateľských útokov na 5 až 0,5 percent a je veľmi lahko implementovateľná. O týchto dvoch prístupoch zatiaľ moc neviem, podrobnejšie budú rozpísane vo finálnej práci.

Otázkou, či sa nepriateľským prípadom dá vôbec zabrániť, alebo sú naozaj nevyhnutné, sa zaoberá táto práca[16]. Nepriateľské útoky sú aktuálne veľmi horúcou témou, väčšina obranných stratégií je len niekoľko desiatok mesiacov stará a už sa našli spôsoby, ako ich obísť a myslím si, že všetci, čo chcú aplikovať strojové učenie v reálnom svete by mali o tejto téme aspoň niečo vedieť, či už sú to samotní programátori, alebo ich nadriadení.

Kapitola 3

Poisoning

Modely strojového učenia sa stávajú stále viac a viac komplikovanejšími, niektoré v sebe skrývajú až stovky miliónov parametrov a na ich úspešné natrénovanie je potrebný obrovský objem dát. Trénovanie takýchto modelov je mimoriadne výpočtovo náročné a aj napriek tomu, že sa trénovanie vykonáva na viacerých GPU súčasne, tréning môže trvať aj niekoľko dní, týždnov a doknca aj mesiacov[2]. Menšie spoločnosti alebo jednotlivci častokrát nemajú prístup k dostatočnému výpočtovému výkonu, alebo sú v časovej tiesni a tento problém potom riešia outsource-ovaným tréningom[10], ktorý môže prebiehať v dvoch podobách:

- **Plne outsource-ovaný tréning** : V tomto prípade sa trénovanie modelu odohráva na serveroch poskytovateľa takejto služby, ako sú napríklad Google Prediction API[1], Amazon Machine Learning[2] alebo Microsoft Azure Machine Learning[3]. Okrem vysokého výkonu tieto služby poskytujú aj jednoduché API pre menej skúsených programátorov v oblasti strojového učenia. Nie je teda potrebné a v niektorých prípadoch je dokonca až nemožné nastavovať štruktúru alebo typ modelu a trénovací algoritmus[18].

- **Transfer learning** označuje postup, pri ktorom sa najskôr stiahne už natrénovaný model strojového učenia, ale s podobným využitím ako má mať cielový model. Tento stiahnutý model sa následne lokálne dotrénuje na menšom mnostve dát. Napríklad aj knižnica Keras poskytuje mnoho predtrénovaných modelov s desiatkami miliónov parametrov.

Druhý a tiež veľmi rozšírený typ útoku je **poisoning** (otrávenie). Trénovanie modelov pomocou zahlučnených dát je známy problém datovaný ešte k minulému storočiu, čo vo svojej podstate je aj poisoning, ale tieto zahlučnené dáta boli úmyselne vložené

[1]https://cloud.google.com/ai-platform
[2]https://aws.amazon.com/machine-learning/
[3]https://studio.azureml.net

útočníkom medzi trénovacie dáta. Poisoning by sa dal ešte rozdeliť do dvoch kategórií podľa úmyslu útočníka. V prvom prípade útočník mieri na dostupnosť systému. Medzi trénovacie dáta vloží tak veľa otrávených dát, že po trénovaní je model v podstate nefunkčný. Tento útok je aplikovateľný na všetky modely strojového učenia. Napríklad v tomto výskume[20] otrávili len 3% trénovacích dát a za následok to malo zníženie presnosti modelu až o 11%.

Druhá kategória poisoningu je viac sofistikovaná a nenápadnejšia. Útočník stále vkladá zlé dáta do trénovacej množiny, ale teraz nechce model znefunkčniť, ale vytvoriť si zadné vrátka (backdoor), teda pridať nejakú funkcionalitu, ktorú môže neskôr zneužiť. Model bude fungovať aj naďalej presne tak ako má a nikto si tieto zadné vrátka nemusí všimnúť. Napríklad útočník môže doučiť klasifikátor emailov na spam, aby email obsahujúci nejaké konkrétne slovo označil ako normálny email teda nie spam.

Generovanie trénovacích dát, ale aj samotné trénovanie modelov sú stále pomerne drahé záležitosti aj z hľadiska finančného, ale aj časového. Preto sa častokrát využívajú buď verejné dáta na trénovanie (datasety) ako napríklad MNIST a ImageNet, alebo sa využijú už predtrénované modely, ktoré sa neskôr dajú ešte dotrénovať na špecifickejšiu úlohu. (Táto metóda sa nazýva transfer learning.) Človek nemusí byť génius, aby si uvedomil, že takéto datasety alebo modely už môžu byť otrávené. Príklad poisoningu môžeme vidieť na obrázku 3.1

Obr. 3.1: Obrázok je z práce [20], kde otrávili systém detegujúci dopravné značky tak, že systém klasifikoval bežné značky správne, ale ak na nich bola špeciálna nálepka, klasifikoval ich zámerne nesprávne. Tu môžeme vidieť, že stop značka s malou žltou nálepkou bola klasifikovaná ako rýchlostný limit

Kapitola 4

Privacy attacks

4.1 Membership Inference

Membership inference je útok, ktorý odpovedá na jednoduchú otázku: za predpokladu, že máme k dispozícií model strojového učenia a konkrétny záznam (vstup pre tento model), membership inference zistuje, či sa tento záznam nachádzal alebo nenachádzal v datasete, na ktorom bol model trénovaný. V prípade konvolučnej neurónovej siete určenej na rozpoznávanie písaných číslic tento útok neznie vôbec zaujímavo, ani nebezpečne. Nikoho nezaujíma, či takýto model bol trénovaný na konkrétnom obrázku čísla 5 a už vonkocom nie, keď je trénovaci dataset verejne dostupný. No nie všetky datasety sú verejné a napríklad v prípadoch ako je Google's Smart Compose[5], model ktorý navrhuje dokončenie vety pri písaní emailov, môže byť tento útok veľkou hrozbou, pretože tento model je trénovaný na reálnych emailoch.

Shokri et al.[18] sa venovali tomuto útoku v black-box situácií najskôr ale s podmienkou, že útočník má okrem výsledného labelu prístup aj k vektoru pravdepodobností ku všetkým labelom. Útok je založený na zistení, že modely strojového učenia sa správajú inak na dátach, ktoré boli využité na trénovanie modelu, ako na dátach, ktoré vidia po prvýkrát. Nie je ale možné, aby si človek tento rozdiel v správaní všimol, preto je na realizáciu tohto útoku vuyžité strojové učenie, ktoré vyniká v rozpoznávaní takýchto patternov. Útok je teda zedefinovaný ako klasifikačný problém: na základe vektora pravdepodobností rozhodni, či záznam bol v datasete alebo nie.
Na natrénovanie takéhoto útočného modelu je využitá "shadow training technique", ktorá spočíva v tom, že najskôr si útočník vytvorí viacero shadow modelov, ktoré napodobňujú správanie originálneho modelu. Dôležitý rozdiel medzi shadow modelom a originálnym modelom je to, že poznáme trénovací dataset shadow modelu, ktorý vieme napríklad vygenerovať z originálneho modelu aj v black-box situácií. Následne sa na shadow modeloch natrénuje útočný model, ktorý už odpovedá na otázku, či sa záznam

nachádzal v trénovaciom datasete alebo nie.
Membership inference je všeobecne aplikovateľný, teda nezáleží na datasete ani na modeli na ktorý sa útočí. Úspešnosť útoku v tejto štúdii bola 74% až 94% a útok nevytváral žiadne falošné negatívy, čo znamená, že ak membership inference klasifikoval záznam ako patriaci do trénovacieho datasetu, vždy to tak bolo. Úspešnosť tohto útoku je priamo úmerna schopnosti modelu generalizovať a rôznorodosti trénovacich dát. Z modelu, ktorý je preučený a nedostatočne generalizuje, alebo má nevhodné trénovacie dáta uniká viac informácií. Pretrénovanie teda nieje nežiadúce len z pohľadu efektívnosti modelu, ale aj úniku informácií.

Ako pri evasion a aj poisoning, ani tomuto útoku sa nedá úplne zabrániť. Regulačné techniky ako napríklad dropout[19] môžu pomôcť proti pretrénovaniu modelov, čo zníži úspešnosť útoku. Ďalšia jednoduchá a nádejná obranná metóda vyzerala aj zmena výstupu modelu tak, že namiesto celého vektora pravdepodobností model vydá iba triedu s najväčšou pravdepodobnosťou. Neskôr sa ale ukázalo[18], že memebership inference je aplikovatelný aj v tejto (úplne black-box) situácií. Za najúspešnejšiu obrannú metódu sa zatiaľ považuje differencial privacy[**?**], ktorá do určitej miery zabraňuje membership inference a aj iným útokom zameraných na súkromie.

4.2 Model Inversion

Model inversion sa tiež týka súkromia dát, na ktorých bol model strojového učenia natrénovaný. Útočník sa pri tomto útoku snaží extrahovať dáta zastupujúce nejaký label priamo z modelu. Nie je možné extrahovať konkrétnu inštanciu z datasetu, ale iba priemernú reprezentáciu triedy modelu. Na obrázku 4.1 je znázornený tento útok, kde je pomocou model inversion útoku vygenerovaná tvár človeka, ktorá sa dostatočne podobá záznamu z trénovacích dát na to aby niekto určil, že ide o tú istú osobu[6]. Pri aplikácii ako je rozpoznávanie tvárí útok generuje jednotlivé pixeli obrázku, ktoré repzrezentujú človeka k prislúchajúcemu labelu. V tomto prípade model inversion neznie až tak závažne, ale strojové učenie býva využité čím ďalej, tým viac na riešenie najrôznejších problémov a v niektorých prípadoch môže byť uník trénovacích dát oveľa závažnejší. Fredrikson et al.[7]

Obr. 4.1: Obrázok vľavo je extrahovaný z modelu pomocou model inverision útoku a obrázok vpravo patrí do trénovacieho datasetu. Útočník mal v tomto prípade white-box prístup k neurónovej sieti rozpoznávajúcej ľudské tváre.

Záver

V práci som vysvetlil evasion útok, poisoning útok a útoky zamerané na súkromie. Každý útok som predviedol na neurónovej sieti, ktorá rozpoznáva MNIST datasete kvôli jednoduchosti modelu a dobrej vizualizácií útokov. Netreba zabudnúť, že všetky tieto útoky su univerzálne a hrozia aspon do určitej miery aj na všetkých ostatných modeloch strojového učenia. Tieto útoky považujem za najdôležitejšie, pretože v niektorých situáciach môžu napáchať obrovské škody a ťažko sa týmto útokom predchádza. Programátori venujúci sa strojovému učeniu by si mali neustále uvedomovať hrozby týchto útokov a v prípade, že by sa niektorému útoku nedalo zabrániť jeho dopad by mal vážne následky, mali by uvažovať nad využitím inej technológie ako strojové učenie.

Literatúra

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate, 2014. Published as a conference paper at ICLR 2015.

[2] Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Dębiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, Rafal Józefowicz, Scott Gray, Catherine Olsson, Jakub Pachocki, Michael Petrov, Henrique Pinto, Jonathan Raiman, Tim Salimans, Jeremy Schlatter, and Susan Zhang. Dota 2 with large scale deep reinforcement learning, 12 2019.

[3] Keshav Bimbraw. Autonomous cars: Past, present and future - a review of the developments in the last century, the present scenario and the expected future of autonomous vehicle technology. *ICINCO 2015 - 12th International Conference on Informatics in Control, Automation and Robotics, Proceedings*, 1:191–198, 01 2015.

[4] Hongge Chen, Huan Zhang, Duane Boning, and Cho-Jui Hsieh. Robust decision trees against adversarial examples. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 1122–1131, Long Beach, California, USA, 09–15 Jun 2019. PMLR.

[5] Mia Xu Chen, Benjamin N. Lee, Gagan Bansal, Yuan Cao, Shuyuan Zhang, Justin Y. Lu, Jackie Tsay, Yinan Wang, Andrew M. Dai, Zhifeng Chen, Timothy Sohn, and Yonghui Wu. Gmail smart compose: Real-time assisted writing. *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019.

[6] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15, page 1322–1333, New York, NY, USA, 2015. Association for Computing Machinery.

[7] Matthew Fredrikson, Eric Lantz, Somesh Jha, Simon Lin, David Page, and Thomas Ristenpart. Privacy in pharmacogenetics: An end-to-end case study of personalized warfarin dosing. In *23rd USENIX Security Symposium (USENIX Security 14)*, pages 17–32, San Diego, CA, August 2014. USENIX Association.

[8] Divya Gopinath, Guy Katz, Corina S. Pasareanu, and Clark W. Barrett. Deepsafe: A data-driven approach for checking adversarial robustness in neural networks. *CoRR*, abs/1710.00486, 2017.

[9] A. Graves, A. Mohamed, and G. Hinton. Speech recognition with deep recurrent neural networks. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 6645–6649, 2013.

[10] T. Gu, K. Liu, B. Dolan-Gavitt, and S. Garg. Badnets: Evaluating backdooring attacks on deep neural networks. *IEEE Access*, 7:47230–47244, 2019.

[11] Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. Adversarial examples are not bugs, they are features. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 125–136. Curran Associates, Inc., 2019.

[12] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial examples in the physical world. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Workshop Track Proceedings*. OpenReview.net, 2017.

[13] Xiaodan Li, Yuefeng Chen, Yuan He, and Hui Xue. AdvKnn: Adversarial Attacks On K-Nearest Neighbor Classifiers With Approximate Gradients. *arXiv e-prints*, page arXiv:1911.06591, Nov 2019.

[14] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *2016 IEEE Symposium on Security and Privacy (SP)*, pages 582–597, May 2016.

[15] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[16] Ali Shafahi, W. Ronny Huang, Christoph Studer, Soheil Feizi, and Tom Goldstein. Are adversarial examples inevitable? In *International Conference on Learning Representations*, 2019.

[17] Adi Shamir, Itay Safran, Eyal Ronen, and Orr Dunkelman. A Simple Explanation for the Existence of Adversarial Examples with Small Hamming Distance. *arXiv e-prints*, page arXiv:1901.10861, Jan 2019.

[18] R. Shokri, M. Stronati, C. Song, and V. Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 3–18, May 2017.

[19] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 06 2014.

[20] Jacob Steinhardt, Pang Wei Koh, and Percy Liang. Certified defenses for data poisoning attacks. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 3520–3532, Red Hook, NY, USA, 2017. Curran Associates Inc.

[21] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna Estrach, Dumitru Erhan, Ian Goodfellow, and Robert Fergus. Intriguing properties of neural networks. Paper presented at 2nd International Conference on Learning Representations, ICLR 2014, Banff, Canada.

Príloha A: obsah elektronickej prílohy

V prílohe sa nachádza neurónova sieť rozpoznávajúca obrázky písaných číslic, taktiež program, ktorý takúto neurónovu sieť vytvorí a natrénuje a taktiež sa v prílohe nachádza implementácia evasion útoku, poisoning útoku a model inversion útoku na túto sieť.