Využitie GAN sietí v detekcii anomálií

Bc. Laura Vištanová

Vedúci práce: RNDr. Ľubomír Antoni, PhD.

Úvod

V posledných rokoch sa čoraz väčšia pozornosť upriamuje na zber dát. Je to spôsobené informatizáciou mnohých oblastí nášho života a zlacňovaním pamäťových jednotiek, čo nám umožňuje uchovávať veľké množstvo dát.

Zbieranie a uchovávanie dát síce máme k dispozícii, aj samotných dát je dostatočne veľa, ozajstnou výzvou ale ostáva spracovanie týchto dát. Vďaka rôznorodosti dát a prostredí z ktorých pochádzajú, sa v posledných rokoch vyvinuli rôzne metódy a prístupy analýzy dát a každý deň sa vymýšľajú nové. Neexistuje jedinečný spôsob akým extrahovať informácie z dát a to robí odvetvie analýzy dát tak zaujímavým. Všetky prístupy musíme prispôsobiť prostrediu z ktorých dáta pochádzajú, štruktúre dát a ich vlastnostiam. Napriek tomu, že dnes je známych už mnoho algoritmov a prístupov na spracovanie dát, najčastejšie nestačí iba aplikovať jednotlivé metódy, ale potrebujeme ich mierne modifikovať, aby sme s vhodne predspracovaných dát získali čo najviac poznatkov. Nie je nezvyčajné, aby viaceré prístupy zlyhali predtým, ako sa nájde ten správny pre danú dátovú sadu. Aj takéto zlyhania nás však vedia posunúť dopredu ak dokážeme aspoň čiastočne odhaliť príčinu ich zlyhania. V oblasti analýzy dát sa často používajú aj rôzne kombinácie algoritmov, o čo sme sa pokúsili aj v tejto diplomovej práci.

Detekcia anomálií je rozšírenou podoblasťou analýzy dát. Využíva sa takmer v každom odvetví, ktoré si vieme predstaviť. V tejto práci sme sa zamerali na detekciu podvodníkov v reálnych bankových dátach. Keďže podvodných transakcií je výrazne menej ako tých platných, tak podvodné transakcie môžeme uvažovať ako anomálie. Dáta sme prevzali z Kaggle súťaže a na nich sme aplikovali nový prístup detekcie anomálií.

Najväčším problémom detekcie anomálií je fakt, že anomálie nie sú časté v dátach a teda je ťažké sa ich učiť. Hlavnou myšlienkou bolo aplikovanie GAN sietí, ktoré umožňujú

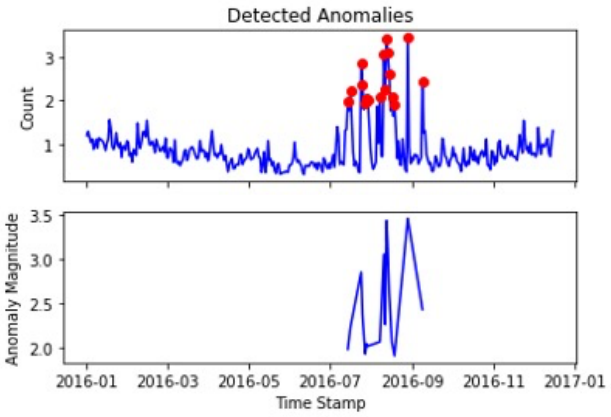
generovanie umelých dát na základe reálnych. Takto vieme doplniť dátovú sadu o umelých podvodníkov, vďaka čomu zvýšime ich pomer v dátach a tým uľahčíme učenie anomálií.

V tejto práci prezentujeme postup detekcie anomálií, ako aj podrobnejší popis jednotlivých častí postupu. V prvej časti predstavíme čo sú to anomálie, aké typy anomálií poznáme. V druhej časti sa venujeme algoritmom na vyhľadávanie anomálií. Popisujeme najmä 3 metódy, a to rozhodovacie stromy, náhodný les a metódu XGBoost. Tretia kapitola je venovaná GAN sieťam, ich základnej myšlienke, rôznym typom sietí. Štvrtá kapitola je venovaná GAN sieťam pre tabuľkové dáta. V nej popisujeme ťažkosti spojené s generovaním tabuľkových dát a 2 modely TGAN a CTGAN. V závere práce sa predstaví dátová sada, popíše presný postup detekcie anomálií s medzivýsledkami a následným porovnaním konečných výsledkov.

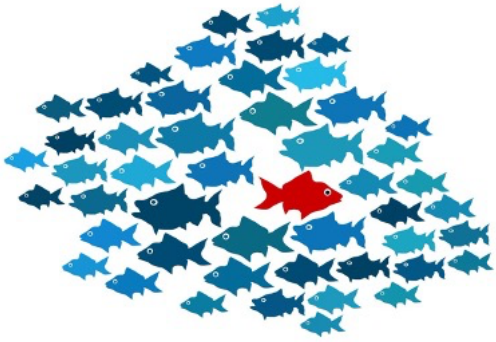
1 Definícia anomálie

Anomáliou nazývame také pozorovanie v dátach, ktoré vnímame ako ťažko vysvetliteľnú odchýlku od normálneho stavu. Táto definícia je celkom ťažkopádna, nehovorí nám toho veľa o anomáliách. Dôvodom je to, že anomália vo výsledku EEG vyšetrenia bude vyzerať inak, ako v prípade anomálie v spotrebe domu, keďže aj normálne správanie týchto systémov je výrazne odlišné.

Môžeme sa pozrieť na obrázok 1 a obrázok 2, na ktorých vidíme 2 rôzne situácie a v každej z nich anomália znamená niečo iné. Na obrázku 1 máme priebeh spotreby domu v jednej miestnosti, červené body sa vyhodnotili ako anomálie. Voľným okom vidíme, že naozaj v tých bodoch nadobúda modrá funkcia vyššie hodnoty, ako vo väčšine bodov. Na druhom obrázku (prevzatý z [1]) by asi každý hneď odhalil anomáliu, je to červená rybka plávajúca opačným smerom ako ostatné.



Obrázok 1 [Anomálie v spotrebe domu]



Obrázok 2 [Ďalší príklad anomálie]

Anomálie majú jednu významnú vlastnosť, ktorá definuje základný problém detekcie anomálií. Anomálie sú z definície také pozorovania, ktoré sa výrazne odlišujú od normy, z čoho vyplýva, že tieto pozorovania sa v dátach objavujú iba zriedka. Keďže ich zastúpenie je v dátach veľmi malé, je náročné sa naučiť ich odhaliť.

1.1 Typy anomálií

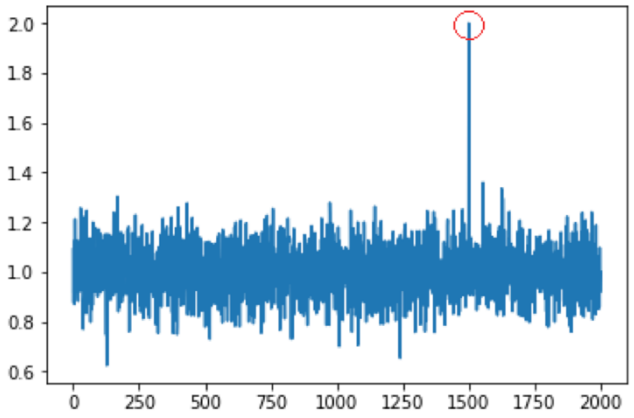
Anomálie podľa ich vlastností delíme do troch skupín:

- bodové anomálie
- kontextuálne anomálie
- kolektívne anomálie

1.1.1 Bodové anomálie

Bodovou anomáliou nazývame pozorovanie, ktoré sa dá považovať za anomáliu v porovnaní s ostatnými dátami. Najčastejšie sa skúmajú práve anomálie tohto typu. Na obrázku môžeme vidieť systém, ktorého namerané hodnoty normálne nepresahujú hodnotu 1.4. V bode 1500 ale vidíme pozorovanie, v ktorom je nameraná hodnota 2.0, teda výrazne presahuje normálne namerané hodnoty a teda tento bod nazveme anomáliou.

Ako príklad z reálneho sveta si vieme predstaviť platobné transakcie. Ak svojou kartou stále platíme menej ako 100 eur, tak ak nám pribudne platba v hodnote 500 eur, tak ju považujeme za bodovú anomáliu, ktorá môže poukázať či už na odcudzenie karty, alebo výnimočný nákup.

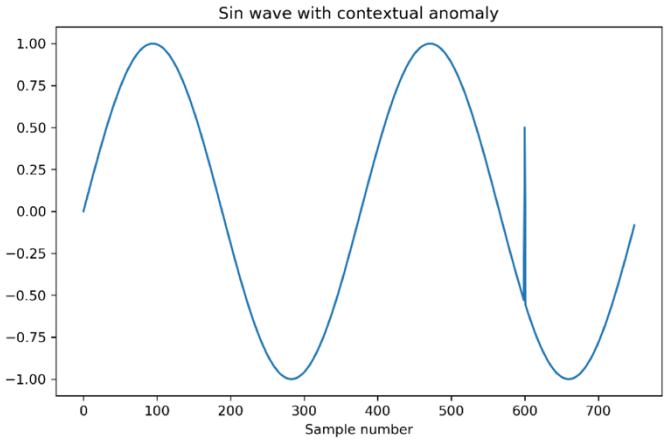


Obrázok 3 [Príklad bodovej anomálie]

1.1.2 **Kontextuálne anomálie**

Kontextuálnou anomáliou nazývame pozorovanie, ktoré síce nie je anomáliou v porovnaní s ostatnými dátami, ale je anomáliou v kontexte dát. Ak sa pozrieme na obrázok, tak môžeme vidieť, že nameraná hodnota v bode 600 nie je hodnotovo vyššia alebo nižšia ako v iných bodoch, ale nezapadá do kontextu dát. Ak by sme totiž sledovali priebeh hodnôt, tak by sme očakávali nameranú hodnotu okolo -0.5 namiesto nameraných 0.5.

Takéto chyby môžu byť aj výsledkom zlého merania, ale pozrime sa ako by takáto anomália vyzerala pri platbách. Predstavme si, že každý týždeň chodíme v piatok na veľký nákup v hodnote 100 eur a v utorok na malý nákup v hodnote 30 eur. Ak prídu sviatky a my nakúpime v utorok za 70 eur, tak táto hodnota bude kontextuálnou anomáliou. Hodnota nášho nákupu síce nepresiahla zvyčajné maximum 100 eur, ale v ten deň sme očakávali nákup za 30 eur.

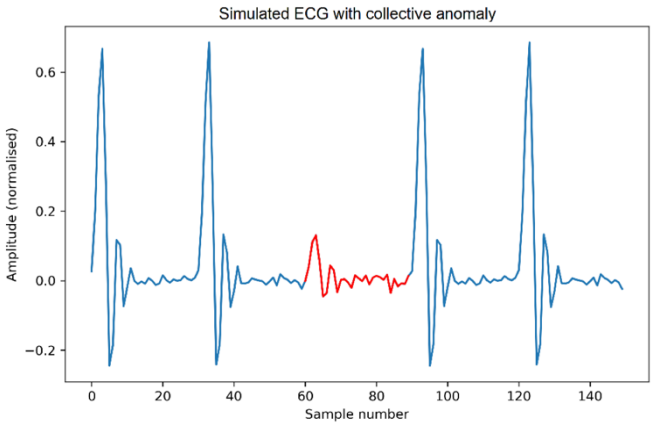


Obrázok 4 [Príklad kontextuálnej anomálie]

1.1.3 **Kolektívne anomálie**

Postupnosť bodov nazývame kolektívnou anomáliou, ak táto postupnosť je anomáliou v porovnaní s ostatnými dátami. Jednotlivé body tejto postupnosti nemusia byť samostatne anomáliami, ale ich spoločný výskyt z nich robí kolektívnu anomáliu.

Ako príklad z reálneho sveta si môžeme predstaviť výsledok EKG vyšetrenie, ktorý je uvedený na obrázku. Ako môžeme vidieť, červená množina bodov je v tomto prípade kolektívnou anomáliou, pretože táto postupnosť nezodpovedá normálnemu správaniu systému (modré časti), aj keď jednotlivé body nemusia byť nutne anomáliami.



Obrázok 5 [Príklad kolektívnej anomálie]

2 Algoritmy na hľadanie anomálií

2.1 Rozhodovacie stromy

Táto podkapitola je spracovaná podľa popisu rozhodovacích stromov v knihe Charu C. Aggarwal – Data Mining.

Rozhodovacie stromy patria medzi známe metódy na riešenie klasifikačných úloh. Sú jednou z najrozšírenejších metód aj vďaka tomu, že je jednoduché pomocou nich popísať proces klasifikácie.

Táto metóda vytvára na základe tréningovej množiny strom. V strome sa nachádzajú dva typy vrcholov - vnútorné vrcholy a listy (koreň sa považuje za vnútorný vrchol). V každom kroku vytvárania stromu sa dáta delia podľa deliaceho pravidla. Deliace pravidlo nám dá podmienku, ktorú majú spĺňať dáta v ľavom a pravom podstrome. Ak máme dáta o ľuďoch, tak deliť ich môžeme napríklad podľa pohlavia, a teda do ľavého podstromu zaradiť všetky dáta zodpovedajúce ženám a do pravého podstromu zaradiť všetky dáta zodpovedajúce mužom. Následne môžeme podstromy deliť podľa veku a tak nám v podstromoch vzniknú ďalšie dva podstromy. Ak si zvolíme hranicu 40 rokov, tak sa nám dáta rozdelia do štyroch skupín: ženy do 40 rokov, ženy nad 40 rokov, muži pod 40 rokov, muži nad 40 rokov. Keďže rozhodovacie stromy chceme použiť na klasifikáciu, potrebujeme cieľový atribút, ktorý nám dáta rozdelí do viacerých skupín. Uvážme najjednoduchší prípad, a to binárnu klasifikáciu, kedy sa budeme snažiť určiť, či daná osoba získa hypotéku alebo nie. Ak dáme údaje o osobe na vstup, tak v každom vnútornom vrchole vieme určiť do ktorého podstromu osoba patrí (podľa deliaceho pravidla) a takto sa vieme dostať k listu. V listoch sa uchováva informácia o tom, do ktorej triedy sa daná osoba klasifikuje, ak skončí v danom liste. Aby sme nemali príliš veľký strom a aby nedošlo k preučeniu, potrebujeme pri vytváraní stromov uvážiť aj kritérium zastavenia. Je zrejmé, že keď sa nám do niektorého vrcholu dostanú iba dáta z rovnakej triedy, tak tieto dáta už nepotrebujeme ďalej deliť a teda z daného vrcholu urobíme list, ktorý bude vstupy klasifikovať do tej triedy, do ktorej patria všetky dáta v ňom.

Pozrime sa teraz bližšie na spôsoby delenia dát. Deliaci bod vyberáme tak, aby sme čo najviac zvýšili homogenitu vzniknutých podmnožín príkladov. Homogenitu vieme vypočítať rôznymi spôsobmi, jedným z nich je určiť hodnotu Giniho indexu. Giniho index je vhodné použiť pre kategoriálne atribúty, čím nižšia je hodnota indexu, tým lepšie vie daný atribút rozdeliť príklady do tried.

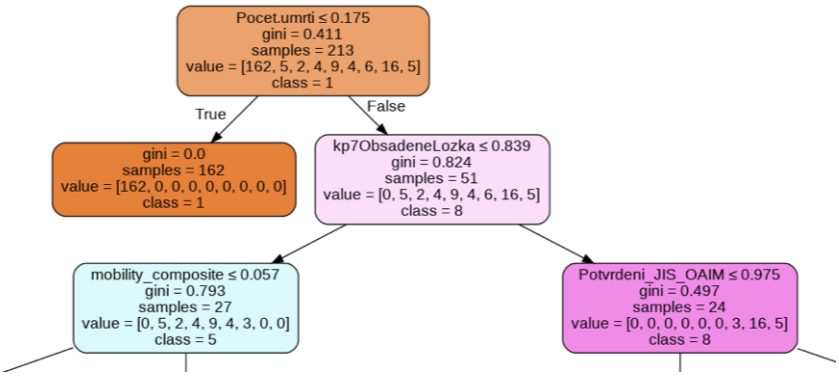
Giniho index v uzle vieme vyjadriť v tvare:

$$G(A) = \sum_{a \in A} p_{A,a,i}^2$$

kde A označuje atribút, a je možná hodnota atribútu a $p^2(A, a, i)$ označuje pravdepodobnosť, že hodnota atribútu A v i-tom príklade tréningovej množiny bude a. Deliť množinu teda budeme podľa toho atribútu, ktorého hodnota Giniho indexu bude najmenšia.

Ukážeme si definované pojmy ešte na jednom príklade, v ktorom úlohou bolo klasifikovať počet potrebných pľúcnych ventilácií v nemocniciach počas pandémie koronavírusu. V tomto prípade sa nerobila regresia, keďže nás nezaujímal presný počet potrebných prístrojov, ale klasfikácia, ktorá určila interval potrebného počtu prístrojov. Závažnosť situácie je teda priamo úmerná triedam v klasifikácii, pretože trieda 1 vyjadrovala najnižší potrebný počet prístrojov a trieda 10 predstavovala množstvo, ktoré by zdravotníctvo už nezvládlo.

Na obrázku môžeme vidieť časť rozhodovacieho stromu. Na vrchu vidíme koreň, v ktorom sa ako deliaci atribút vybral atribút s názvom Pocet.umrti. V danom strome sa homogenita počítala podľa už spomenutého Giniho indexu, ktorý dáta rozdelil do dvoch skupín. Z obrázku vieme tiež prečítať, že väčšina dát sa dostala do ľavého podstromu, v ktorom sa nachádza 162 príkladov z tréningovej množiny (z celkového počtu 213 príkladov) a všetky sa klasifikujú do triedy 1. Môžeme tiež vidieť, že v prípade, keď atribút Pocet.umrti bol nanajvýš 0.175, tak sa automaticky vstupné dáta vyhodnotili ako patriace do triedy 1. V prípade, že bola táto hodnota vyššia, sa pozeralo na hodnotu 7 dňového kĺzavého priemeru obsadených lôžok v nemocniciach, ktorý nám opäť rozdelil dáta na dva podstromy. Takto sa pokračuje ďalej, kým nie je splnené kritérium zastavenia.



Obrázok 6 [Príklad rozhodovacieho stromu]

2.2 Náhodný les

Náhodný les vznikne generovaním viacerých rozhodovacích stromov, ide teda o rozšírenie predchádzajúcej metódy. Rozhodovacie stromy do lesu sa vytvárajú na rôznych tréningových dátach s náhodným výberom atribútov. Výsledná klasifikácia sa uskutoční tak, že každý rozhodovací strom bude klasifikovať a za celkový výsledok zoberieme ten, ktorý sa objavil u najviac stromov.

2.3 XGBoost

Táto podkapitola bola spracovaná podľa [2] a [3].

2.3.1 **Boosting a bagging**

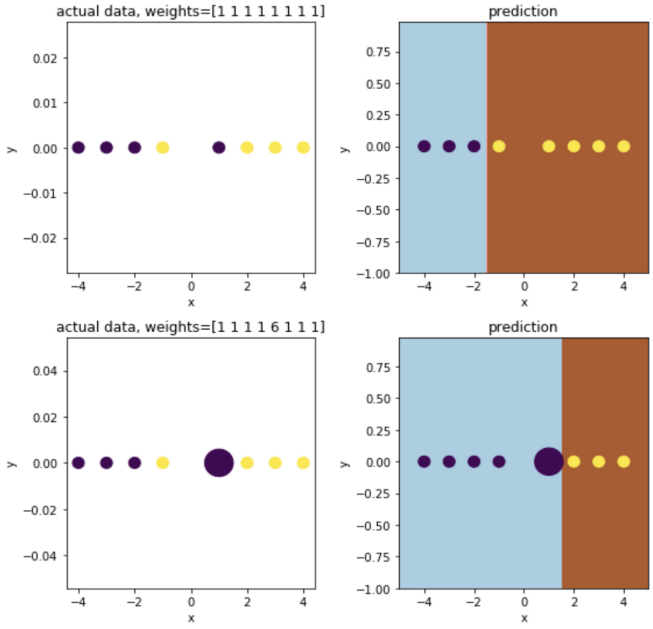
Metódy boostingu a baggingu patria medzi ensemble metódy strojového učenia. Hlavnou myšlienkou v oboch prístupoch je to, že kombinujeme viacero slabých klasifikátorov, čím vytvoríme jeden silný klasifikátor. Obe metódy sa však líšia v tom, ako vytvárajú jednotlivé klasifikátory. V prípade baggingu sa tréningová množina rozdelí na podmnožiny a na každej z nich sa vytvoria rozhodovacie stromy. Takto máme viacero stromov, ktoré klasifikujú vstupné dáta, pričom celkový výstup sa vytvorí pomocou výstupov jednotlivých stromov. Chyba takého modelu je nižšia, ako by bola chyba jedného rozhodovacieho stromu.

Boosting sa skladá tiež z viacerých klasifikátorov, ale tieto sa vytvárajú v postupnosti. Kým v prípade baggingu boli jednotlivé rozhodovacie stromy budované nezávisle od seba, boosting vytvára stromy tak, aby ďalší vytvorený strom opravil chybu predchádzajúceho stromu.

Rozlišujeme dva základné spôsoby boostingu. V prípade AdaBoostu sa budujú stromy vždy z modifikovanej tréningovej množiny, ktorá vznikne upravením váh jednotlivých príkladov. Ak sa pozrieme na aktuálne budovaný strom, tak sa vyhodnotia príklady, na ktorých bola vysoká chyba klasifikácie, zvýši sa váha týchto príkladov a s týmito modifikovanými váhami sa posielajú dáta na vytvorenie nového stromu.

Na obrázku 7 vľavo hore môžeme vidieť tréningovú množinu ôsmich bodov, ktorá obsahuje dáta z dvoch tried (žltá a fialová farba). Za ním nasleduje obrázok, ktorý znázorňuje výstup prvého stromu, teda rozdelenie dát do dvoch množín. Ako môžeme vidieť, tento prvý strom sa pomýlil iba v jednom prípade, a to v prípade piateho fialového bodu, preto sa zvýši váha tohto príkladu v tréningovej množine. Zo zoznamu nad obrázkom vidíme, že sa zvolila

váha 6 namiesto pôvodnej váhy 1. Následne ďalší klasifikátor kvôli vysokej váhe piateho príkladu delil dáta iným spôsobom a teda opravil chybu predchádzajúceho stromu.



Obrázok 7 [Prvé kroky boostingu na príklade]

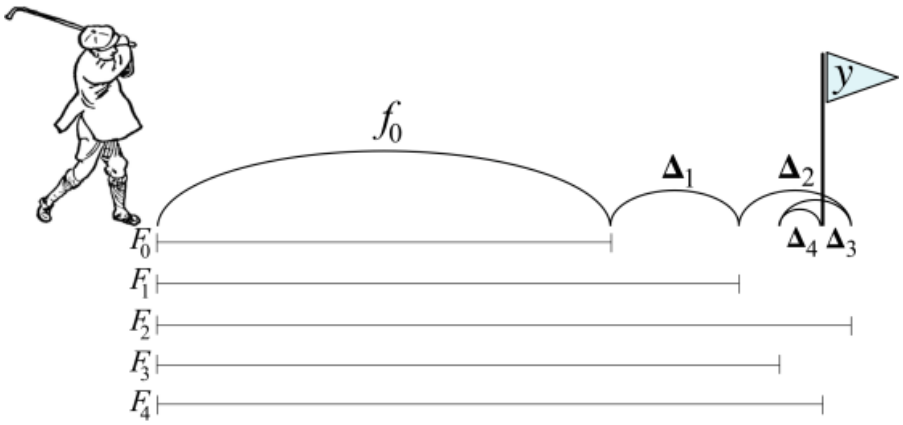
Druhý spôsob sa nazýva gradientný boosting. V ňom sa nemenia váhy jednotlivých príkladov tréningovej množiny, ale pomocou stratovej funkcie sa pozerá na rozdiel medzi výstupom modelu a ozajstnou hodnotou cieľového atribútu. Táto metóda je vhodná ako pre regresiu, tak aj pre klasifikáciu.

Na obrázku 8 môžeme vidieť pseudokód gradientného boostingu so zvolenou stratovou funkciou najmenších štvorcov. Vidíme, že stromy sa budujú postupne, pričom pri každom strome sa určí hodnota stratovej funkcie, vypočíta sa záporný gradient stratovej funkcie a buduje sa model, ktorý ako cieľový atribút používa gradienty. Záporný gradient počítame z dôvodu, že chceme určiť smer, v ktorom stratová funkcia klesá najrýchlejšie. Na konci sa tieto estimátory sčítajú a tak dostaneme výsledok modelu.

$fit\ estimator\ F^1$
$for\ i\ in\ [1, M]$//M weak estimators
$Loss^i = \sum_{j=1}^{n} \left(Y_j - F^i(X_j)\right)^2$ // loss in i^{th} iteration
$calculate\ neg\ gradient: -\frac{\partial L^i}{\partial X_j} = -\frac{2}{n} * \left(Y_j - F^i(X_j)\right) \forall i$
$Fit\ a\ weak\ estimator\ H^i\ on\ \left(X, \frac{\partial L}{\partial X}\right)$
$//\rho\ changes\ the\ step\ size$
$Prediction: F^m(X) = F^i(X) + \rho * H^i(X) = F^1 + \rho * \sum_{i=1}^{m} H^i(X)$

Obrázok 8 [Pseudokód gradientného boostingu]

Tento algoritmus v skutočnosti používame aj v reálnom svete. Na obrázku môžeme vidieť hru golfu, na ktorej na začiatku hráč urobí nejaký odhad f_0, ktorý potom postupne zlepšuje a snaží sa aproximovať hodnotu y.



Obrázok 9 [Aplikácia boostingu v reálnom svete]

3 GAN siete

Úvod tejto kapitoly ako aj obrázky je spracovaný na základe článku [4].

GAN siete patria medzi pomerne mladé metódy v oblasti strojového učenia. Táto metóda bola prvýkrát popísaná v roku 2014 Ianom Goodfellowom a odvtedy sa veľkým tempom zlepšovala a vyvíjala. GAN siete slúžia na vytvorenie umelých dát na základe tréningovej sady. Pôvodne sa používali na generovanie obrázkov, vďaka čomu vieme generovať neexistujúce ľudské tváre. Aby bol rýchly pokrok v tejto oblasti ešte viditeľnejší, uvádzame obrázok (ľudskú tvár), ktorú vygenerovala jedna z prvých GAN sietí v roku 2014 (obrázok 10) a pre porovnanie ďalší vygenerovaný obrázok pomocou StyleGAN z roku 2018 (obrázok 11).

Obrázok 10 [Obrázok z roku 2014]

Obrázok 11 [Obrázok z roku 2018]

Vďaka GAN sieťam už teraz dokážeme generovať umelé obrázky nerozlíšiteľné od pravých pre ľudí, tak, ako to môžeme vidieť na obrázku. Aby čitateľ videl, že tieto siete dokážu generovať akékoľvek reálne vyzerajúce obrázky a nie len ľudské tváre, tak uvádzame príklad ďalších obrázkov so zvieratami, jedlom a prírodou.

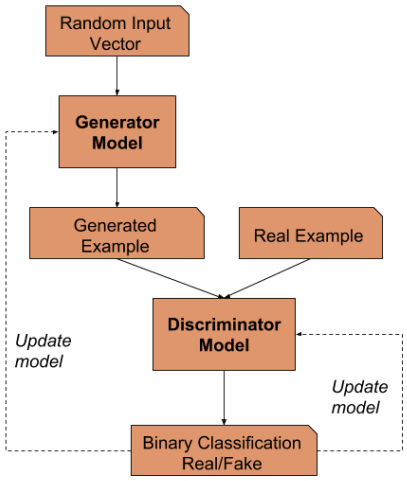
Obrázok 12 [Ďalšie príklady umelo vytvorených fotografií]

V tejto kapitole si predstavíme princíp fungovania GAN sietí. Vzhľadom na to, že GAN siete boli prvotne vytvorené na generovanie obrázkov a väčšina sietí sa zaoberá generovaním obrázkov, v nasledujúcej kapitole predstavíme GAN siete pre tabuľkové dáta a popíšeme v čom sa líšia od tých pôvodných sietí pre obrázky, v čom spočívali ťažkosti navrhnutia takýchto sietí.

3.1 Popis GAN sietí

Táto podkapitola vznikla na základe článku [5].

GAN siete pozostávajú z dvoch neurónových sietí, ktoré súťažia proti sebe v hre s nulovým súčtom (výhra jednej siete je prehrou v druhej). Prvá zo sietí sa nazýva *generátor*. Jej úlohou je generovanie umelých dát. Druhá sieť sa nazýva *diskriminátor* a jej úlohou je rozlíšiť umelo vygenerované vzorky od tých reálnych. Na vstupe dostáva obrázky jednak z tréningovej sady ale aj tie, ktoré vytvoril generátor. Pomocou klasifikačných metód sa snaží obrázky rozdeliť do dvoch množín a to na pravé (z tréningovej sady) a umelé (vygenerované generátorom).



Obrázok 13 [Štruktúra GAN sietí]

Prácu GAN sietí si vysvetlíme pomocou obrázku. Trénujú sa 2 siete (generátor a diskriminátor), pričom ich trénovanie prebieha separátne, teda buď trénujeme generátor alebo diskriminátor. Váhy neurónových sietí sa aktualizujú pomocou spätnej propagácie chyby. Trénovanie diskriminátora prebieha v nasledujúcich krokoch:

I. diskriminátor dostane na vstupe obrázky z tréningovej sady a obrázky vygenerované generátorom

II. klasifikuje vstupné obrázky do dvoch tried – pravé obrázky (z tréningovej sady) a umelo vygenerované obrázky (výstup generátora)
III. vyhodnotí sa chyba diskriminátora a spätnou propagáciou chyby sa aktualizujú váhy diskriminátora

Trénovanie generátora je podobné, najväčší rozdiel spočíva v tom, že váhy neurónovej siete sa upravujú na základe chyby diskriminátora. Kroky trénovania sú nasledovné:

I. generátor dostane na vstupe náhodný vektor
II. generátor generuje umelé obrázky
III. kvalitu vygenerovaných obrázkov určuje diskriminátor, ktorého presnosť rozhoduje o tom, či sú vygenerované obrázky dostatočne dobré
IV. na základe chyby diskriminátora sa spätnou propagáciou aktualizujú váhy generátora

Ako už bolo spomenuté, jednotlivé siete sa trénujú zvlášť a z popisu jednotlivých krokov tréningu je zrejmé, že generátor postupne generuje obrázky čoraz viac podobné tréningovej sade, kým diskriminátor má čoraz väčší problém rozlíšiť odkiaľ obrázky pochádzajú. Otázkou teda ostáva, dokedy má zmysel tieto siete trénovať. Trénovanie končí, keď obrázky generované generátorom nedokáže diskriminátor odlíšiť od ozajstných obrázkov, čiže keď už len tipuje výsledok klasifikácie (teda presnosť klasifikácie je okolo 50 %).

4 GAN siete pre tabuľkové dáta

Kapitola 4 vznikla na základe článkov [6], [7] a [8].

V predchádzajúcej kapitole boli predstavené GAN siete a spôsob, akým generujú tieto siete obrázky. GAN siete boli síce pôvodne vymyslené pre potreby generovania obrázkov, ale v posledných rokoch sa výskum vybral aj smerom generovania iných štruktúr, ako napríklad tabuľkových dát. Ak chceme generovať tabuľkové dáta podobné nejakým reálnym dátam, tak sa potrebujeme vysporiadať s novými problémami, ktoré pri generovaní obrázkov neboli prítomné:

- rôzne typy atribútov
- rôzne tvary rozdelení v dátach
- nevyvážené kategoriálne premenné

Pri vytváraní GAN sietí pre tabuľkové dáta bolo potrebné tieto problémy vyriešiť, aby sme dokázali efektívne generovať reálne, celočíselné alebo kategoriálne dáta. Prirodzenou požiadavkou tiež je, aby vzniknuté dáta mali podobné štatistické vlastnosti, ako tie reálne, preto by tieto algoritmy mali vedieť generovať dáta z ľubovoľného rozloženia dát.
V nasledujúcich podkapitolách si predstavíme vybrané GAN siete, ktoré generujú tabuľkové dáta a ktoré sme aj aplikovali na našu dátovú sadu. Treba podotknúť, že generovanie tabuľkových dát sa objavilo až pár rokov po predstavení GAN sietí, preto je v čase písanie tejto práce ešte relatívne málo článkov a GAN sietí, ktoré sa dajú použiť na generovanie tabuľkových dát.

4.1 TGAN (Tabular Generative Adversarial Network)

Predstavme si tabuľku, v ktorej máme numerické (diskrétne alebo spojité) a kategoriálne dáta s nejakým rozdelením dát X. Chceme vytvoriť model, ktorý bude generovať riadky z tohto rozdelenia tak, že pre novovytvorené dáta budú platiť nasledujúce 2 podmienky:

I. ľubovoľný model strojového učenia má podobnú presnosť na nových dátach ako na pôvodných
II. vzťah medzi ľubovoľnými dvomi atribútmi sa zachová aj v nových dátach

Neurónové siete dokážu efektívne generovať hodnoty z rozdelenia centrovaného na (-1,1) pomocou *tanh* a tiež multinomické rozdelenie pomocou *softmax*. Túto vlasnosť využijeme a urobíme na dátach transformáciu, aby sme mali splnené predpoklady. Tieto transformácie musia spĺňať jednu dôležitú vlastnosť – musia byť reverzibilné. Je to spôsobené tým, že na výstupe chceme mať tabuľku (riadok tabuľky), ktorá je na nerozoznanie od reálnej tabuľky, preto ak urobíme transformáciu na dátach, tak sa spätne musíme vedieť vrátiť do pôvodného tvaru. Transformácie popíšeme zvlášť pre numerické a kategoriálne premenné.

4.1.1 **Numerické premenné**

Pri transformácii numerických premenných sa používa GMM (Gaussian Mixture Model), ktorý je zovšeobecnením normálneho rozdelenia. Dáta totiž relatívne často nemajú normálne rozdelenie, ale vieme ich vyjadriť pomocou viacerých normálnych rozdelení. To znamená, že ak atribút vyjadríme pomocou m normálnych rozdelení, tak získame $\mu_1, \dots, \mu_m$ stredných hodnôt a $\sigma_1, \dots, \sigma_m$ štandardných odchýlok. Pre danú hodnotu atribútu c následne určíme pre každé z m rozdelení normalizovanú pravdepodobnosť, že c pochádza práve z toho

rozdelenia. Získame tak m ďalších hodnôt $u_1, \dots, u_m$. Vyberieme najväčšiu pravdepodobnosť, označme ju u_k a normalizujeme hodnotu c nasledujúcim vzťahom:

$$v = \frac{c - \mu_k}{2\sigma_k} \qquad (*)$$

Hodnotami $u_1, \dots, u_m$ a v následne prezentujeme pôvodnú hodnotu c.

V článku autori rozkladali atribúty na 5 normálnych rozdelení. Môžeme takto uvažovať, ak je ich totiž menej, tak niektoré z nich sa takmer vynulujú a teda dostaneme aj tak dobré rozdelenie pre daný atribút.

4.1.2 **Kategoriálne premenné**

Transformácia kategoriálnych premenných sa udeje v nasledujúcich krokoch:

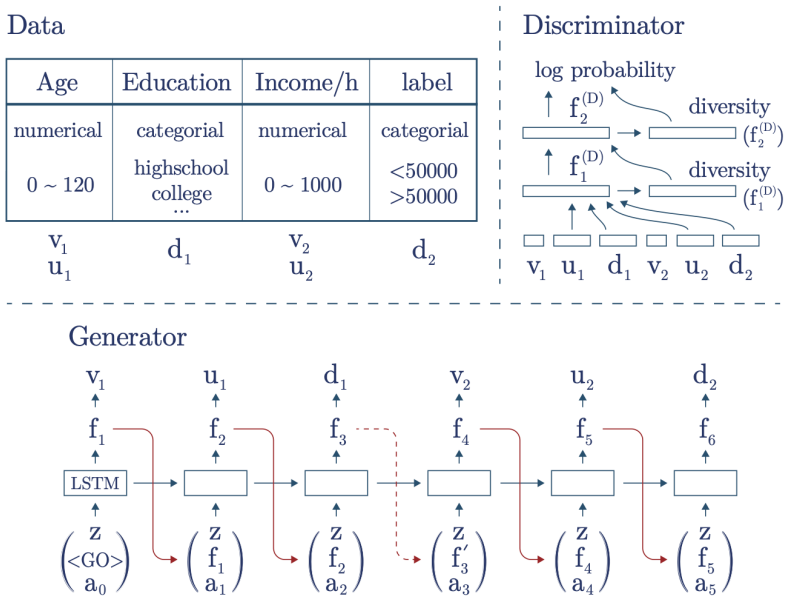
- každú hodnotu d diskrétnej premennej transformujeme na one-hot vektor $\boldsymbol{d}$ dimenzie takej, koľko kategórií máme v danej premennej
- ku každej dimenzii pridáme nejaký malý náhodný šum vzťahom: $\boldsymbol{d} = \boldsymbol{d} + Uniform(\gamma)$, pričom autori zvolili $\gamma = 0.2$
- normalizujeme vektor $\boldsymbol{d}$: $\frac{\boldsymbol{d}}{\sum \boldsymbol{d}}$

Týmto spôsobom máme popísanú vhodnú transformáciu ako pre numerické, tak aj pre kategoriálne premenné. GAN sieť následne generuje hodnoty $u_1, \dots, u_m$ a v pre numerické dáta a vektor $\boldsymbol{d}$ pre kategoriálne dáta. Spätná transformácia pri numerických dátach sa robí pomocou vzťahu $(*)$, z ktorého vieme vyjadriť c. Pri kategoriálnych premenných sa z vektora $\boldsymbol{d}$ vyberie najpravdepodobnejšia kategória.

4.1.3 **Model siete**

V tejto časti popíšeme, ako vyzerá model diskriminátora a generátora. Aby bol popis zrozumiteľnejší, popíšeme ho na príklade z pôvodného článku.

Na obrázku vidíme tabuľku so štyrmi atribútmi (vek, vzdelanie, príjem, označenie). Dva atribúty sú kategoriálne (vzdelanie a označenie), zvyšné dva atribúty sú numerické (vek a príjem).



Obrázok 14 [Príklad generovania tabuľkových dát pomocou TGAN]

Generátor: Z transformácie dát vyplýva, že numerické dáta budeme generovať v dvoch krokoch (zvlášť vektor $\boldsymbol{u}$ a zvlášť skalár v), kategoriálne premenné budeme generovať v jednom kroku (vektor $\boldsymbol{d}$). Tabuľka na obrázku obsahuje dve numerické premenné a dve kategoriálne, teda generovanie jedného riadku tabuľky budeme robiť v šiestich krokoch. Generátor pozostáva z LSTM sietí, pričom na vstupe v ľubovoľnom kroku t má sieť nasledujúce 3 prvky:

- náhodný vektor z
- vektor f_{t-1} alebo f'_{t-1} (závisí od typu predchádzajúceho výstupu, či sme generovali numerický alebo kategoriálny atribút). Na začiatku nemáme vektor z výstupu, teda aplikujeme <GO> vektor, ktorý sa priebežne učíme
- vektor a_{t-1}, v prvom kroku inicializujeme $a_0 = 0$.

Výstup LSTM je vektor h_t, ktorý použijeme pre určenie hodnoty $f_t = \tanh(W_t h_t)$, kde W_t je parameter siete, ktorý sa sieť učí. Následne vektor f_t použijeme, aby sme získali výstup. Výstup závisí od toho, aký typ premennej generujeme, preto platí:

- ak výstup je časťou v pre numerickú premennú, tak výstup určíme vzťahom: $v_i = \tanh(W_t f_t)$ a do ďalšieho kroku pošleme vektor f_t
- ak výstup je časťou $\boldsymbol{u}$ pre numerickú premennú, tak výstup určíme vzťahom $u_i = \mathrm{softmax}(W_t f_t)$ a do ďalšieho kroku pošleme vektor f_t

- ak výstup má byť diskrétna premenná, tak výstup určíme vzťahom $\boldsymbol{d}_i = \text{softmax}(W_t f_t)$ a do ďalšieho kroku pošleme vektor daný vzťahom $f_t' = E_i[arg_k max \boldsymbol{d}_i]$, kde E je reprezentačná matica pre diskrétnu premennú.

Diskriminátor: Pozostáva z plne prepojenej siete s l vrstvami, pričom na vstupe má konkatenáciu výstupov $\boldsymbol{u}, v, \boldsymbol{d}$. Výpočet vo vrstvách sa robí pomocou nasledujúceho predpisu:

$$f_1^{(D)} = \text{LeakyReLU}(\text{BN}(W_1^{(D)}(v_{1:n_c} \oplus u_{1:n_c} \oplus \mathbf{d}_{1:n_d}))),$$
$$f_i^{(D)} = \text{LeakyReLU}(\text{BN}(W_i^{(D)}(f_{i-1}^{(D)} \oplus \text{diversity}(f_{i-1}^{(D)})))), i = 2 : l,$$

Vidíme, že prvá vrstva má na vstupe iba vstupný vektor, všetky ostatné vrstvy používajú výstup predchádzajúcej vrstvy. BN(.) zodpovedá batch normalizácii, aktivačná funkcia je LeakyReLU. Vysvetlíme ešte, čomu zodpovedá diversity(.).

Pri generovaní sa môže stať, že dôjde k tomu, že generátor začne generovať takmer rovnaké dáta pre celý batch, čím sa snaží oklamať diskriminátora. Diskriminátor sa ale naučí, že tieto dáta sú umelé, preto núti generátor aby sa naučil nejaké nové, ten však opäť vyprodukuje skupinu rovnakých dát, teda nesplní požiadavku, že má generovať rôzne dáta z daného rozloženia. Tento jav sa nazýva Helvetica scenario a predchádza sa mu tým, že sa sleduje rozdielnosť dát v jednej skupinke. Diversity je teda vektor, ktorý určuje vzdialenosť vzorky od ostatných vzoriek vrámci batchu.

Stratová funkcia: Pri trénovaní modelu sa použil optimalizátor Adam. Kvôli zlepšeniu modelu sa k stratovej funkcii pridáva aj KL divergencia[1] pre vektor $\boldsymbol{d}$ a $\boldsymbol{u}$ a dostávame tak nasledujúce vzťahy:

Pre generátor máme stratovú funkciu definovanú ako:

$$\mathcal{L}_G = -\mathbb{E}_{z\sim\mathcal{N}(0,1)} \log D(G(z)) + \sum_{i=1}^{n_c} \text{KL}(u_i', u_i) + \sum_{i=1}^{n_d} \text{KL}(\mathbf{d}_i', \mathbf{d}_i),$$

Pre diskriminátor :

$$\mathcal{L}_D = -\mathbb{E}_{v_{1:n_c}, u_{1:n_c}, \mathbf{d}_{1:n_d}\sim\mathbb{P}(\mathbf{T})} \log D(v_{1:n_c}, u_{1:n_c}, \mathbf{d}_{1:n_d}) + \mathbb{E}_{z\sim\mathcal{N}(0,1)} \log D(G(z)).$$

kde u_i' a d_i' zodpovedajú generovaným dátam a u_i a d_i zodpovedajú reálnym dátam.

[1] KL divergencia zodpovedá Kullback-Leibler divergencii, ktorá určuje vzdialenosť medzi rozdeleniami na vstupe.

4.2 CTGAN (Conditional Generative Adversarial Network)

Autori modelu TGAN sa spojili s ďalšími výskumníkmi a krátko po publikovaní článku o TGAN vydali článok CTGAN, v ktorom prezentujú nový spôsob transformácie riadkov.
Kým diskrétne hodnoty vieme jednoducho prezentovať pomocou one-hot vektorov, pri numerických premenných je transformácia dát na vhodný tvar náročnejšia. Dáta môžu pochádzať z ľubovoľného rozdelenia, čo nám sťažuje generovanie dát. Pre numerické premenné je popísaný nasledujúci postup transformácie pre atribút C_i:

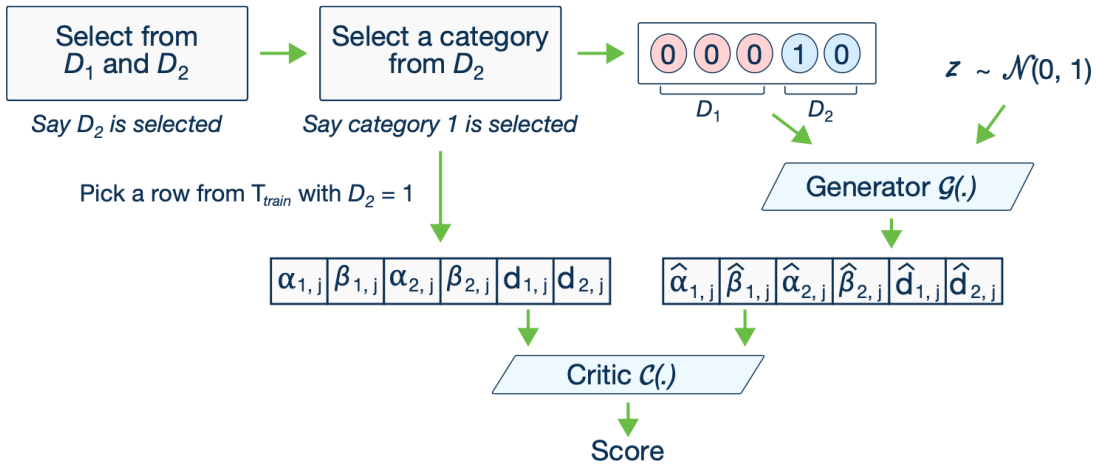
I. Pomocou VGM (variational Gaussian Mixture Model) sa odhadne počet normálnych rozdelení zastúpených v rozdelení premennej C_i, tento počet označíme ako m_i.
II. Pre každú hodnotu $c_{i,j}$ atribútu C_i a každé nájdené normálne rozdelenie určíme pravdepodobnosť, že hodnota pochádza z daného rozdelenia, takto pre každú hodnotu máme m_i pravdepodobností tvoriacich vektor.
III. Z tohto vektora vyberieme niektoré náhodné rozdelenie (s danými pravdepodobnosťami), povedzme, že sme vybrali k-te rozdelenie a definujeme $\alpha_{i,j} = \frac{c_{i,j}-\mu_k}{4\sigma_k}$ a $\beta_{i,j} = [0, \dots, 1, \dots, 0]$, tento vektor má m_i prvkov s 1-kou na k. mieste a všade inde má nuly.

Takto získame reprezentáciu numerickej premennej pomocou hodnoty α a one-hot vektoru β.

V tomto článku sa autori snažili vyriešiť problém generovania nevyvážených kategoriálnych premenných. V reálnych dátach sa často stretávame s nevyváženými kategoriálnymi premennými, čo pri vytváraní modelu môže skresliť výsledok. Ak totiž vyberáme náhodne tréningové dáta, tak sa môže ľahko stať, že v tejto vzorke nebudeme mať obsiahnuté všetky možné kategórie a teda model sa ich nenaučí generovať. Z toho by potom vyplývalo, že generátor negeneruje vzorky z rovnakého rozdelenia aké mali pôvodné dáta, čiže by sme nesplnili jednu z dvoch podmienok, ktoré by mali tieto modely splniť. Túto podmienku vieme vyjadriť nasledovne: uvážme i-tu kategoriálnu premennú D_i a jej konkrétnu hodnotu k. Výstup generátora si označme ako $\hat{\boldsymbol{r}}$. Tieto dáta by mali splniť nasledujúcu podmienku: $\hat{\boldsymbol{r}} \sim P_{\mathcal{G}}(row|\, D_i = k)$, čo je podmienená pravdepodobnosť a odtiaľ pochádza aj názov modelu. To znamená, že diskriminátor má za úlohu naučiť sa podmienené rozdelenie dát, aby platilo $P_{\mathcal{G}}(row|D_i = k) = P(row|D_i = k)$ vďaka čomu vieme skonštruovať pôvodné rozdelenie dát vzťahom: $P(row) = \sum_{k \in D_i} P_{\mathcal{G}}(row| = k)P(D_i = k)$.
Aby sme toto mohli urobiť, potrebujeme reprezentáciu podmienok, teda potrebujeme vyjadriť vzťahy tvaru $D_i = k$.

Uvážme kategoriálne premenné $D_1, \ldots, D_n$ so všetkými hodnotami ktoré môžu nadobudnúť. Vytvoríme nulový vektor pre každý atribút tak, že dimenzia každého vektora zodpovedá počtu možných hodnôt atribútu. Ak chceme vyjadriť vzťah $D_i = k$, tak vo vektore pre atribút D_i na k-tej pozícii zapíšeme 1, čím indikujeme, že dané pozorovanie má túto hodnotu. Tieto vektory nakoniec spojíme, čím dostaneme jeden vektor popisujúci podmienku. Ukážme si tento postup na konkrétnom príklade. Predstavme si, že máme 2 kategoriálne premenné $D_1 = \{1,2,3\}$ a $D_2 = \{1,2\}$ a chceme vyjadriť $D_2 = 1$. Pre atribút D_1 máme vektor $[0, 0, 0]$ a pre atribút D_2 máme vektor $[1, 0]$, ktorých spojením získame vektor, ktorý sa v článku označuje ako $cond = [0, 0, 0, 1, 0]$ a nazýva sa *mask* vektor.

V článku sa ďalej popisuje postup nazývaný training-by-sampling, ktorého kroky popíšeme a ukážeme na príklade.



Obrázok 15 [Príklad k training-b-sampling]

Daný príklad pracuje s tabuľkou, ktorá obsahuje 2 numerické a 2 kategoriálne (D_1, D_2) premenné. Vidíme to z vektorov, ktoré vstupujú do diskriminátora (na obrázku sa nazýva Critic).

Training-by-sampling:

I. pre každú kategoriálnu premennú vytvoríme nulový vektor popísaný vyššie.
II. náhodne vyberieme kategoriálnu premennú D_i. V príklade sa vybrala premenná D_2.
III. vytvoríme distribučnú funkciu pre vybranú kategoriálnu premennú tak, že hodnota distribučnej funkcie bude pre každú hodnotu rovná logaritmu frekvencie výskytu danej hodnoty.

IV. na základe distribučnej hodnoty náhodne zvolíme hodnotu pre vybraný atribút. V príklade sa zvolila hodnota 1.

V. máme podmienku, tú vyjadríme spôsobom popísaným vyššie. Takto vznikne vektor v príklade, ktorý má na 4. pozícii hodnotu 1 a inde 0.

Vzniknutý vektor následne spolu s náhodným vstupom vytvára vstup pre generátor.

Generátor aj diskriminátor majú podobnú štruktúru ako v prípade TGAN.

Ako bolo už popísané, generátor má na vstupe vektor $cond$ spolu s náhodným vektorom. V tomto prípade máme iba 2 skryté vrstvy, následne vytvárame výstup podľa toho, aký typ premennej generujeme (podobne ako pri TGAN).

Štruktúra generátora:

$$\begin{cases} h_0 = z \oplus cond \\ h_1 = h_0 \oplus \texttt{ReLU}(\texttt{BN}(\texttt{FC}_{|cond|+|z|\to 256}(h_0))) \\ h_2 = h_1 \oplus \texttt{ReLU}(\texttt{BN}(\texttt{FC}_{|cond|+|z|+256\to 256}(h_1))) \\ \hat{\alpha}_i = \texttt{tanh}(\texttt{FC}_{|cond|+|z|+512\to 1}(h_2)) & 1 \le i \le N_c \\ \hat{\beta}_i = \texttt{gumbel}_{0.2}(\texttt{FC}_{|cond|+|z|+512\to m_i}(h_2)) & 1 \le i \le N_c \\ \hat{\mathbf{d}}_i = \texttt{gumbel}_{0.2}(\texttt{FC}_{|cond|+|z|+512\to |D_i|}(h_2)) & 1 \le i \le N_d \end{cases}$$

kde FC je lineárna transformácia, ktorá transformuje u-dimenzionálny vstup na v-dimenzionálny výstup a gumbel je aktivačná funkcia Gumbel softmax.

Štruktúra diskriminátora:

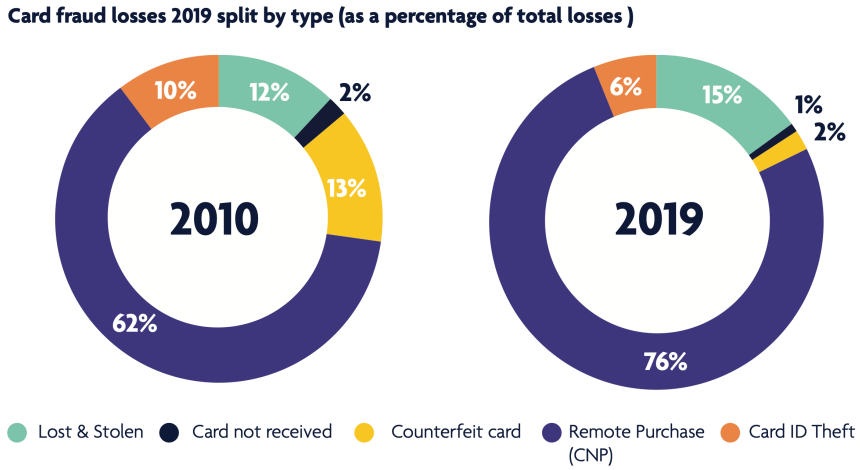
$$\begin{cases} h_0 = \mathbf{r}_1 \oplus \ldots \oplus \mathbf{r}_{10} \oplus cond_1 \oplus \ldots \oplus cond_{10} \\ h_1 = \texttt{drop}(\texttt{leaky}_{0.2}(\texttt{FC}_{10|\mathbf{r}|+10|cond|\to 256}(h_0))) \\ h_2 = \texttt{drop}(\texttt{leaky}_{0.2}(\texttt{FC}_{256\to 256}(h_1))) \\ \mathcal{C}(\cdot) = \texttt{FC}_{256\to 1}(h_2) \end{cases}$$

Ako optimalizátor sa aj tu používa Adam s učiacim pomerom $2 \cdot 10^{-4}$.

5 Dátová sada

5.1 Odhaľovanie podvodníkov v bankových transakciách

Podvodné transakcie predstavujú každý rok vysoké straty ako pre klientov, tak aj pre obchody, ktoré sú zodpovedné za straty. Väčšina podvodov sa stane cez CNP transakcie. CNP transakcie (card-not-present transakcie) sú také platby, počas ktorých nie je fyzicky prítomná karta a vlastník karty. Typické CNP platby sú online platby alebo automaticky opakované mesačné platby. Pomer CNP platieb k ostatným druhom platieb v posledných rokoch neustále rastie, na obrázku 16 môžeme vidieť porovnanie z rokov 2010 a 2019 [9]. Rovnako každý rok rastú aj škody spôsobené podvodníkmi. Motivácia odhaliť podvodníkov pri platbách je vysoká, pretože za všetky straty sú zodpovedné obchody, do ktorých išla platba. Štatistiky z roku 2020 hovoria, že každý jeden dolár v podvodnej transakcii stál obchodníkov v priemere 3.36 dolárov. Je to spôsobené tým, že obchody musia vrátiť peniaze, ale aj prešetriť všetky transakcie, takže aj samotné analyzovanie a vyhodnotenie nahlásených transakcií ich niečo stojí.



Obrázok 16 [Štatistiky podvodov za rok 2010 a 2019]

Banky a obchodníci sa snažia rôznymi spôsobmi chrániť pred podvodníkmi. Bežne sa používa určenie hodnoty risk score v reálnom čase pre transakcie, na základe ktorého sa vyhodnotí, či je transakcia bezpečná. Rozvojom mobilných zariadení sa pridalo aj overenie majiteľa buď cez odtlačok prstu, overenie hlasu alebo rozoznávanie tvárí. Ďalšia možnosť na ochranu pred podvodníkmi je použiť metódy strojového učenia, čo bude aj náš prípad.

5.2 Kaggle súťaž

V roku 2019 bola na Kaggle stránke vypísaná súťaž [10] na odhaľovanie podvodníkov v CNP transakciách. Súťaž bola vypísaná spoločnosťou IEEE Computational Intelligence Society, pričom samotné dáta o transakciách poskytla spoločnosť Vesta, ktorá sa zaoberá odhaľovaním podvodníkov v transakciách. Súťaž trvala pol roka, víťazom sa podarila dosiahnuť presnosť 94,6 %.

Vesta poskytla do súťaže 2 dátové sady, ktoré majú spolu 434 atribútov. Medzi atribútmi sa nachádzajú informácie o kartách, adresách, vzdialenostiach a rôzne iné. Kvôli citlivosti dát nepoznáme význam všetkých atribútov, vieme ale, že niektoré atribúty sú také, ktoré si vytvorila už samotná Vesta, pretože ich považovala za dôležité. Je podstatné poznamenať, že pri všetkých transakciách máme poznačené, či sa jednalo o podvodnú transakciu alebo nie. V praxi máme často pozorovania bez toho, aby sme vedeli, ktoré sa považujú za anomálie a ktoré nie, vtedy je ešte náročnejšie sa ich nejakým spôsobom naučiť.

6 Návrh modelu a výsledky

V poslednej kapitole predstavujeme celkový navrhnutý postup detekcie anomálií na zvolenej dátovej sade. Hlavnou myšlienkou postupu je použitie GAN sietí na generovanie umelých podvodných transakcií, práve preto budeme porovnávať výsledky na pôvodnej tréningovej sade bez umelých podvodníkov s výsledkami rovnakých algoritmov aplikovaných na tréningovú sadu doplnenú o umelých podvodníkov.

6.1 Návrh modelu

Celkový postup, ktorý v tejto práci navrhujeme a aplikujeme na reálne dáta sa dá zhrnúť do nasledujúcich bodov:

I. úprava dát
II. výber atribútov
III. generovanie umelých dát
IV. vytvorenie novej tréningovej množiny
V. klasifikácia dát (podvodná transakcia/platná transakcia)

6.1.1 **Úprava dát**

Pri práci s dátami je prvý krok vždy úprava dát, preto nemohla chýbať ani v tomto prípade. Túto časť sme však nerobili samostatne, totiž v rámci súťaže boli urobené viaceré predspracovania dát, ktorým predchádzali netriviálne úvahy a pozorovania. V popise súťaže navyše neboli kvalitne popísané atribúty dátovej sady, autori súťaže ozrejmovali niektoré fakty len po položení otázky v diskusiách, pričom kvôli citlivosti dát pri niektorých atribútoch ani neprezradili čo predstavujú. Tento fakt urobil vytvorenie dobrého predspracovania ešte náročnejším, preto sme použili postup predspracovania, ktorý používal víťazný tím tejto súťaže.

6.1.2 **Výber atribútov**

Vybraná dátová sada obsahuje okolo 400 atribútov a stovky tisíc riadkov, preto by si práca s takýmito veľkými dátami vyžadovala veľa času. Kvôli zmenšeniu veľkosti dát sme sa rozhodli do postupu zakomponovať výber atribútov. Použili sme 2 algoritmy na výber najdôležitejších atribútov, aby sme výsledky dokázali porovnať aj podľa toho, s ktorými atribútmi sme pracovali. Pre tento účel sme použili algoritmus SelectKBest a ExtraTreesClassifier.

6.1.2.1 **SelectKBest**

Prvá metóda, ktorú sme použili na výber atribútov je SelectKBest z knižnice scikit-learn (https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectKBest.html#sklearn.feature_selection.SelectKBest) (pridám radšej číslo)

Táto metóda sa pozerá na vzťah jednotlivých atribútov a cieľovej premennej pri klasifikačnom probléme. Vyberá tie atribúty, ktoré najviac súvisia s cieľovou premennou. Spôsob, akým vyberá „najlepšie“ atribúty si popíšeme v tejto podkapitole.

Metóda vyžaduje 3 základné vstupy, a to:

X – množinu atribútov, z ktorých chceme vybrať tie „najlepšie“

y – cieľová premenná, podľa ktorej klasifikujeme dáta do tried

k – počet atribútov, ktoré chceme použiť.

SelectKBest počíta 2 hodnoty, konkrétne F-hodnotu a p-hodnotu a to pre každú dvojicu atribút-cieľová premenná. Na základe nich usporiada jednotlivé atribúty a následne metóda vráti prvých k atribútov. Čím vyššie je F-skóre, tým dôležitejší je uvažovaný atribút pri klasifikácii podľa cieľovej premennej.

Ak uvážime klasifikáciu do dvoch tried (+ a -), tak F-skóre i-teho atribútu vieme vyjadriť nasledovným vzťahom:

$$F(i) \equiv \frac{\left(\bar{\boldsymbol{x}}_i^{(+)} - \bar{\boldsymbol{x}}_i\right)^2 + \left(\bar{\boldsymbol{x}}_i^{(-)} - \bar{\boldsymbol{x}}_i\right)^2}{\frac{1}{n_+ - 1}\sum_{k=1}^{n_+}\left(x_{k,i}^{(+)} - \bar{\boldsymbol{x}}_i^{(+)}\right)^2 + \frac{1}{n_- - 1}\sum_{k=1}^{n_-}\left(x_{k,i}^{(-)} - \bar{\boldsymbol{x}}_i^{(-)}\right)^2},$$

kde $\bar{\boldsymbol{x}}_i^{(+)}$ je priemer hodnôt i-teho atribútu v triede +, $\bar{\boldsymbol{x}}_i^{(-)}$ je priemer hodnôt i-teho atribútu v triede – a $\bar{x}_i$ je celkový priemer hodnôt atribútu. Hodnota $x_{k,i}^{(+)}$ vyjadruje hodnotu i-teho atribútu pre k-ty príklad z triedy +, analogicky definujeme hodnotu $x_{k,i}^{(-)}$.

Ako môžeme vidieť, čitateľ obsahuje rozptyly jednotlivých tried (+ a -) v celom i-tom atribúte. Menovateľ sa pozerá na rozptyl v jednotlivých triedach. Čím menší je rozptyl vnútri tried, tým je hodnota menovateľa menšia a teda hodnota zlomku väčšia, preto hľadáme vysoké F-hodnoty. Nevýhodou tohto prístupu je, že sa na jednotlivé atribúty pozerá zvlášť, teda týmto spôsobom neodhalíme skupiny atribútov, ktoré spolu dobre dokážu klasifikovať dáta. Môže sa totiž stať, že dvojica atribútov má zvlášť nízke F-skóre, ale keby sme ich spojili, tak by sme dostali

výrazne lepší výsledok. Práve preto sa nemôžeme spoľahnúť na to, že týmto spôsobom získame najlepšiu možnú *k*-ticu atribútov, ale vieme, že bude aj napriek tomu dostatočne dobrá. (článok: https://www.csie.ntu.edu.tw/~cjlin/papers/features.pdf)

ešte tu je to pekne vysvetlené: https://stats.stackexchange.com/questions/20341/the-disadvantage-of-using-f-score-in-feature-selection

6.1.2.2 **ExtraTreesClassifier**

Metóda ExtraTreesClassifier je implementovaná tiež v knižnici scikit-learn a používa sa nielen na výber atribútov, ale najmä na klasifikáciu. Táto metóda sa podobá na metódu náhodného lesu, ktorá bola popísaná v kapitole 2. Najväčšími rozdielmi sú:

- náhodný les nepracoval naraz s celou tréningovou vzorkou, iba s jej časťou, kým ExtraTreesClassifier pracuje s celou tréningovou vzorkou.
- pre náhodný výber atribútov v náhodnom lese sa pre každý atribút určil deliaci bod, ktorý zabezpečil najlepšiu možnú homogenitu dát. Následne sa zvolil atribút, pre ktorý bolo delenie najlepšie. V tomto prípade sa deliace body vyberajú náhodne, teda sa zrýchľuje výpočet.

Ako môžeme vidieť, v týchto algoritmoch je zabudované určenie dôležitosti jednotlivých atribútov vzhľadom na klasifikáciu dát, práve preto ich môžeme použiť ak chceme určiť najdôležitejšie atribúty.
(https://towardsdatascience.com/an-intuitive-explanation-of-random-forest-and-extra-trees-classifiers-8507ac21d54b)

6.1.3 **Generovanie umelých dát**

Pomocou GAN sietí predstavených v kapitole 4 vytvoríme umelú dátovú sadu na základe reálnej. V tejto časti sme použili siete TGAN a CTGAN s rôznymi nastaveniami parametrov. Pri generovaní umelých dát môžeme skúsiť 2 základné prístupy:

I. generovanie iba podvodných transakcií z dátovej sady, ktorá obsahuje iba podvodné transakcie
II. generovanie celej dátovej sady s následným filtrovaním podvodných transakcií z výslednej dátovej sady

V práci sme využívali najmä prístup II., keďže v prvom prípade strácame celkovú informáciu o dátach, čo vlastne znamená, že prichádzame o informáciu, čo robí anomálie anomáliami medzi ostatnými dátami. Na druhej strane kvôli zriedkavému výskytu dát sa model nemusí vedieť naučiť správne tieto dáta, keďže predstavujú iba zlomok z celého súboru a teda do

celkovej chyby modelu prispievajú iba minimálne. Môže sa nám teda stať, že model bude správne generovať nenahlásené transakcie, ale pri generovaní podvodných nebude dostatočne úspešný aj napriek tomu, že celkový výsledok modelu je dobrý.

6.1.4 Vytvorenie novej tréningovej množiny

Pri tejto časti je dôležité podotknúť, že nové transakcie pridávame iba do tréningovej množiny, testovacia sada ostáva nedotknutá. Je to spôsobené tým, že my chceme vytvoriť model, ktorý dokáže spoľahlivo odhaliť podvodné transakcie, ale chceme ho aplikovať na reálne dáta. Pri vytvorení modelu teda môžeme učeniu pomôcť s tým, že pridáme nejaké umelo vygenerované dáta navyše, ale presnosť klasifikácie už testujeme na reálnych dátach.

Ďalšou zaujímavou otázkou je, ako veľa podvodných transakcií chceme doplniť do tréningovej množiny, teda v akom pomere chceme mať podvodné transakcie a platné transakcie.

6.1.5 Klasifikácia dát

Po vytvorení modifikovanej tréningovej množiny ostáva už len klasifikácia dát. Dáta klasifikujeme do dvoch tried, delíme ich na podvodné transakcie a platné transakcie. Použili sme algoritmy popísané v kapitole 2 a to náhodný strom, náhodný les a XGBoost algoritmus.

6.2 Výsledky

Pomocou algoritmov SelectKBest a ExtraTreesClassifier sme vytvorili 5 súborov s nasledujúcimi vlastnosťami:

- najlepších 10 atribútov podľa ExtraTreesClassifier (e10)
- najlepších 10 atribútov podľa SelectKBest použitím funkcie chi2 (s10chi)
- najlepších 30 atribútov podľa ExtraTreesClassifier (e30)
- najlepších 30 atribútov podľa SelectKBest použitím funkcie chi2 (s30chi)
- najlepších 30 atribútov podľa SelectKBest použitím funkcie f_classif (s30f)

Označenia v zátvorke slúžia na identifikáciu jednotlivých dátových súborov, aby sa ľahšie odkazovalo na jednotlivé súbory.

V nasledujúcej tabuľke uvádzame výstup XGBoost algoritmu pre tieto súbory (teda bez umelých dát):

Súbor	AUC
e10	0.78523
e30	**0.90589**
s10chi	0.70090
s30chi	0.81713
s30f	0.78430

Ako môžeme vidieť, najlepšie výsledky sa dosiahli na súbore e30, teda metóda ExtraTreesClassifier vybrala vhodnejšie atribúty ako SelectKBest.

Pozrime sa na výsledky náhodného lesu:

Súbor	AUC
e10	0.9668307
e30	**0.9768369**
s10chi	0.9649372
s30chi	0.9731628
s30f	0.970940

V tomto prípade sme dosiahli výrazne lepšie výsledky ako pri XGBoost algoritme, opäť zvíťazil súbor e30.

V našich dátach je spolu 590 540 riadkov, z čoho 20 663 je podvodných (čiže ich považujeme za anomálie), čo predstavuje približne 3,5 % anomálií v našich dátach.
Doteraz sme sa zaoberali najmä algoritmom CTGAN. Vytvorili sme viacero modelov pre jednotlivé dátové sady s rôznymi nastaveniami siete. Pôvodne sme využívali bezplatné prostredie Colab, avšak pre náročnosť trénovania GAN sietí sme museli zmeniť prostredie. Trénovanie ešte aj tak trvá viacero hodín, ale výsledok dostaneme v relatívne dobrom čase. Pri doterajších nastaveniach učenia sa nepodarilo výrazne zlepšiť (ale ani zhoršiť) výsledky XGBoost a náhodného lesu pomocou pridania umelých anomálií do tréningovej množiny. Do odovzdania záverečnej práce preto budeme pracovať na vytvorení lepších GAN modelov, prípadne urobíme ešte nejaké zmeny na dátach s ktorými pracujeme.

Treba podotknúť, že ak sa jedná o bankové transakcie, tak má zmysel ak sa nám podarí zlepšiť presnosť algoritmov hoci len o 0,01 %. Vezmime si štatistiky Európskej centrálnej banky z roku 2019, kedy tvorila hodnota podvodných transakcií 1,5 miliardy eur [12]. Je zrejmé, že pri takýchto vysokých hodnotách znamená aj malé zvýšenie presnosti veľké množstvo ušetrených peňazí.

Záver

V tomto článku sme predstavili nový spôsob detekcie anomálií na reálnych dátach. Základ postupu tvorilo použitie GAN sietí pre účely generovania umelých anomálií. Popísali sme pojem anomálie, typy anomálií a tiež algoritmy pomocou ktorých vieme vyhľadávať anomálie. Podstatnú časť článku sme venovali predstaveniu GAN sietí. Popísali sme všeobecnú štruktúru GAN sietí, ktoré boli vytvorené za účelom generovania obrázkov. Predstavili sme nový typ GAN sietí, pomocou ktorých dokážeme generovať tabuľkové dáta, podrobne sme popísali modely TGAN a CTGAN, ktoré sme aplikovali na naše dáta.

Detekcia anomálií zohráva dôležitú úlohu v spracovaní dát, jej využitie nájdeme v rôznych príkladoch z reálneho sveta. Práve preto sme pri výbere dátovej sady dbali na to, aby sme mali reálne dáta, v ktorých by detekcia anomálií prinášala úžitok pre spoločnosť. Odhaľovanie podvodníkov patrí medzi reálne problémy, ktoré potrebujeme riešiť, preto sa nám použitá dátová sada zdala ako vhodná pre účely tejto práce.

Jadro článku tvorili GAN siete a spôsob, akým sme tieto siete použili na odhaľovanie podvodníkov v bankových transakciách. Vychádzali sme so základného problému detekcie anomálií, a to z faktu, že anomálie sa vyskytujú iba zriedkavo v dátach a preto je náročné sa ich učiť. Skúsili sme teda počet anomálií v dátach zvýšiť pomocou GAN sietí, ktoré generujú umelé dáta na základe reálnych. Za predpokladu, že GAN siete sa vedia lepšie naučiť rozdelenie dát a teda aj samotné anomálie, by sme mohli dosiahnuť lepšie výsledky aj na klasických algoritmoch v dátovej sade doplnenej o generovaných podvodníkov. Naše doterajšie pokusy výrazne nezlepšili (ale ani nezhoršili) presnosť algoritmov, avšak máme ešte dostatok času aby sme skúsili dosiahnuť aspoň minimálne zlepšenie pomocou GAN sietí. A ako sme spomenuli na na konci poslednej kapitoly, aj relatívne malé zlepšenie má v našom prípade veľký zmysel.

Zdroje

[1] Sergio Santoyo: A Brief Overview of Outlier Detection Techniques, zdroj: https://towardsdatascience.com/a-brief-overview-of-outlier-detection-techniques-1e0b2c19e561

[2] Zixuan Zhang: Boosting Algorithms Explained, zdroj: https://towardsdatascience.com/boosting-algorithms-explained-d38f56ef3f30

[3] Terence Parr, Jeremy Howard: Gradient boosting: Distance to target, zdroj: https://explained.ai/gradient-boosting/L2-loss.html

[4] Ajay Uppili Arasanipalai: Generative Adversarial Networks - The Story So Far, zdroj: https://blog.floydhub.com/gans-story-so-far/#cgan

[5] Jason Brownlee: A Gentle Introduction to Generative Adversarial Networks (GANs), zdroj: https://machinelearningmastery.com/what-are-generative-adversarial-networks-gans/

[6] Insaf Ashrapov: GANs for tabular data, zdroj: https://towardsdatascience.com/review-of-gans-for-tabular-data-a30a2199342

[7] Lei Xu, Kalyan Veeramachaneni: Synthesizing Tabular Data using Generative Adversarial Networks, zdroj: https://arxiv.org/pdf/1811.11264.pdf

[8] Lei Xu, Kalyan Veeramachaneni, Maria Skoularidou, Alfredo Cuesta-Infante: Modeling Tabular Data using Conditional GAN, zdroj: https://arxiv.org/pdf/1907.00503.pdf

[9] UK Finance: Fraud – The Facts 2020, zdroj: https://www.ukfinance.org.uk/system/files/Fraud-The-Facts-2020-FINAL-ONLINE-11-June.pdf

[10] IEEE – CIS Fraud Detection, zdroj: https://www.kaggle.com/c/ieee-fraud-detection/overview

[11] Tianqi Chen, Carlos Guestrin: XGBoost: A Scalable Tree Boosting System, zdroj: https://dl.acm.org/doi/pdf/10.1145/2939672.2939785

[12] European Central Bank: Seventh Report on card fraud, zdroj: https://www.ecb.europa.eu/pub/cardfraud/html/ecb.cardfraudreport202110~cac4c418e8.en.html