Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Távközlési és Médiainformatikai Tanszék

DEEP LEARNING MÓDSZEREK INTERPRETÁLHATÓSÁGA

Tanulmány a Magyar Nemzeti Bank részére

Felelős:

Dr. Gyires-Tóth Bálint

Távközlési és Médiainformatikai Tanszék

toth.b@tmit.bme.hu

Budapest, 2021

Tartalomjegyzék

1. Bevezetés

Az elmúlt évtizedek során a gépi tanulás jelentős teret hódított a hagyományos, szakértői rendszerekkel szemben. Szakértői rendszerek esetén egy-egy folyamatra jellemző *szabályokat* manuálisan adunk meg, és ha az adott folyamat megváltozik, akkor a szabályokat ennek megfelelően módosítjuk, töröljük, vagy új szabályokat hozunk létre. Gépi tanuló eljárások esetén *adatokat* rögzítünk az adott folyamatból, és ezeket az adatokat betáplálva a *modellünkbe*, a gépi tanuló eljárás automatikusan alakítja ki a *szabályokat*. Míg szakértői rendszerek esetén explicit módon adjuk meg a szabályokat, és ezekhez teljes mértékben hozzá tudunk férni, addig gépi tanuló eljárások esetén a szabályok a modellekben implicit módon kerülnek kialakításra – és ezekhez közvetlenül nem férünk hozzá. Szakértői rendszerek esetén könnyen magyarázhatóak a döntések, míg gépi tanuló eljárások esetén – az eljárás függvényében beszélhetünk *fehér-, szürke- és feketedoboz* modellekről.

Fehérdoboz modellek esetén a statisztikai eljárások, és a gépi tanuló eljárás döntéseit teljes mértékben meg tudjuk magyarázni, az ok-okozati összefüggés minden aspektusa ismert. *Feketedoboz* modellekben a döntések miértjét nem tudjuk analitikus úton megadni, míg a *szürkedoboz* modellek döntési mechanizmusai részben felderíthetőek. Jellemzően minél összetettebb egy folyamat, annál inkább komplex megoldásra van szükség, így, bár a szakértői rendszerek és a fehér doboz modellek jól magyarázhatóak lennének, mégis, magasabb pontosságuknak köszönhetően gyakran a *szürke- és feketedoboz* modelleket alkalmazzák.

A gépi- és mélytanuló modellek interpretálhatósága (értelmezhetősége) mellett fontos átgondolnunk, hogy mennyire van jelen az élet más területein a műszaki rendszerek interpretálhatósága. Például amennyiben egy számítógép lefagy, akkor újraindítjuk, és ha működik, nem akarjuk, nem tudjuk megérteni, hogy miért fagyott le. Ha sokszor lefagy, lehet, hogy újratelepítjük az operációs rendszert. Ha ezt követően működik, valószínűleg szintén nem akarjuk, és nem is tudjuk megadni a korábbi fagyás okát. Viszont ha továbbra sem működik megbízhatóan, akkor elvisszük a szervízben, ahol várhatóan kicserélnek benne bizonyos egységeket. Ezt követően amennyiben működik, legfeljebb annyit tudunk róla, hogy „elromlott a RAM” vagy „hibás volt az alaplap”, de azt, hogy ez miért következett be, és milyen valószínűséggel fog bekövetkezni a jövőben, nem tudjuk. Szerverkörnyezetben is hasonló a helyzet – azzal a különbséggel, hogy a

szervereket több előzetes tesztnek vetik alá, és egy megfelelő szerverarchitektúrában jelentős a redundancia a biztonságkritikus működés miatt.

Gépi tanulás esetén is megfontolandó hasonló szempontokat szem előtt tartanunk. Nem biztos, hogy a modellek minden részletének ismerete szükséges magas minőségű szolgáltatások nyújtásához, ugyanúgy, ahogy a szerver minden alkatrészét sem ismerik, és az esetleges hibáinak bekövetkezési okait sem ismerik a szolgáltatók. Viszont azt mindenképp kritikus ismerni, hogy milyen hibákra számíthatunk, és ezek elhárítására milyen eszközeink lehetnek.

A fenti gondolatmenet alapján tehát az interpretálhatóság lényegi eleme gépi tanulás esetén a modell működésének és döntéseinek a megértése. Ennek két jelentősége is van. Jogi és társadalmi szempontból az interpretálhatóság hidat képez a matematikai algoritmusok („mesterséges intelligencia", MI) és az azt használó emberek között. Ennek köszönhetően kaphatja vissza az ember a felelősséget az MI döntései felett. Másfelől mérnöki, fejlesztői szempontból az interpretáció segítségével tudjuk kidolgozni az adott folyamatot legjobban leíró gépi- és mélytanuló modellt, és a modellek robusztusságát is így lehetséges növelni.

Az elmúlt évben az *MI 8. sorszámú alprojekt* keretében átadtuk a „Interpretálható felügyelt gépi tanulási modellek" című tanulmányt, mely felelőse Gáspár Csaba (Budapesti Műszaki és Gazdaságtudományi Egyetem, Távközlési és Médiainformatikai Tanszék) volt. Jelen tanulmány elsődlegesen a gépi tanuló eljárások egy kiemelten fontos ágára, a mélytanulásra koncentrál, és a korábbi tanulmány ismerete nélkül is megérthető. Mindamellett a témakör szélesebb megismerése céljából javasoljuk a korábbi tanulmány megismerését is a jelen dokumentum mellett.

1.1. Értelmezhetőség és magyarázhatóság

Az interpretálhatóság magyar jelentése értelmezhetőség. Emellett sokszor előjön a magyarázhatóság (*explainability*) fogalma a gépi tanulásban. A két kifejezés közeli rokonfogalom, és sokszor felcserélik ezek jelentését. Amennyiben külön szeretnénk választani a két kifejezés jelentését, akkor azt az alábbiak szerint tehetjük meg:

- Az **interpretálhatóság** az ok-okozati összefüggéseket keresi gépi tanuló rendszerekben. Ennek célja tehát az, hogy megadjuk, hogy különböző adatok és tanítási beállítások mellett milyen eredményre számíthatunk.

- A **magyarázhatóság** célja egy-egy jelenséget visszavezetni a gépi- és mélytanuló eljárások belső működésére. Ez esetben a pontos indokot keressük.

Például tehát egy pénzügyi mélytanuló modell esetén az interpretálhatóság alatt azt értjük, hogy ismerjük a modell viselkedését különböző beállítások és adatok esetén. Magyarázhatóság alatt pedig azt, hogy meg tudjuk konkrétan határozni azokat a részeit a modellnek, amelyek egy-egy döntéshez vezetnek.

Ezért a fekete- és szürkedoboz modelleket jellemzően nem, vagy csak részben lehet magyarázni, viszont interpretálni lehetséges.

1.2. Az interpretálhatóság taxonómiája

Gépi- és mélytanuló modellek interpretációját végezhetjük az adatok, a predikció és magának a modellnek a segítségével (lásd 1. ábra).

Az **adatok** vizsgálatával feltárhatjuk egyrészt az adatok milyenségét, továbbá a bemenetek és kimenetek közötti kapcsolatot, és ezáltal meg tudjuk vizsgálni, hogy a bemenet különböző változatai (pl. permutációi) mellett milyen predikció várható.

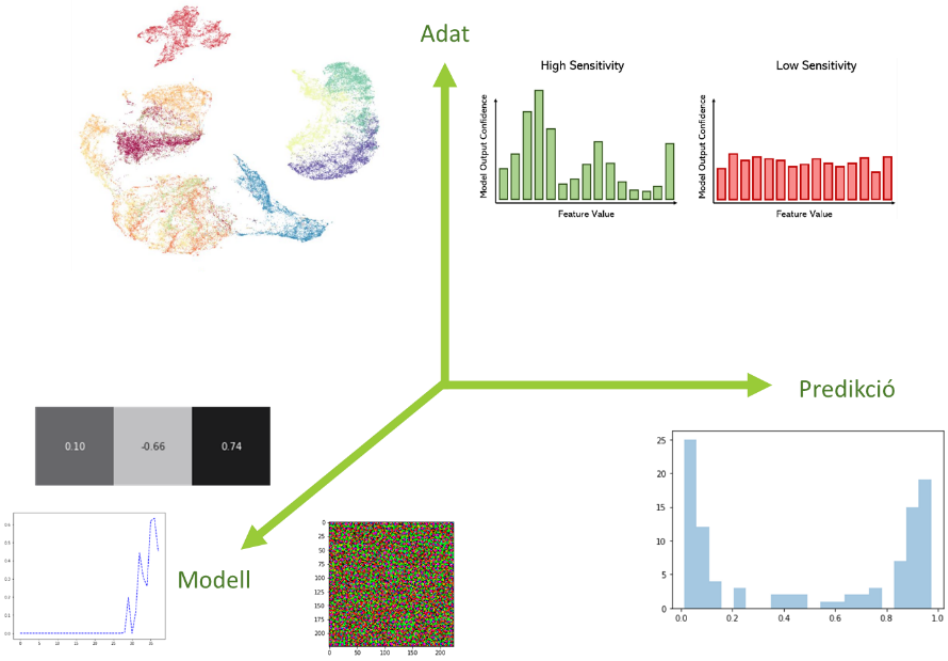
A **predikció** vizsgálata során jellemzően a modell kimenetének milyenségét elemezzük. Klasszifikáció esetén a kimeneti valószínűségi sűrűségfüggvény jellemzőit, illetve az általános osztályozási metrikákat (precision, recall, F1, ROC görbe, AUROC, stb.) elemezzük [1]. Regresszió esetében szintén a vonatkozó metrikákat (négyzetes és abszolút érték hiba, regressziós ábra) vizsgáljuk.

A **modell** elemzése során arra próbálunk magyarázatot találni, hogy mi alapján hozta meg a döntéseket a mély neurális hálózat és mik azok a jellemzők, amiket megtanult a bemenet-kimenet leképzések során. Fontos figyelembe venni, hogy a rétegtípusok magyarázhatósága eltérő: a teljes összeköttetésű rétegek jellemzően nem jól interpretálhatóak, a rekurrens rétegek szintén [2]. Ez alól kivételt képez a sequence-to-sequence (seq2seq) architektúrákban alkalmazott teljes összeköttetésű réteggel megvalósított figyelmi mechanizmus [3]. Ez esetben explicit meg tudjuk állapítani, hogy a bemenet mely részei járultak hozzá jobban vagy kevésbé a döntéshez (*„mely részekre figyelt a modell”).* A konvolúciós rétegek [4] esetén lehetséges a megtanult szűrők értékeit – és ezáltal a szűrők mintázatait elemezni. Szintén egy elterjedt technika konvolúció neurális hálózatok esetén a rétegaktivációk maximalizálása, mely során azt

vizsgáljuk, hogy adott konvolúciós szűrő, vagy szűrők milyen bemenet esetén a legaktívabbak a modellben.

Míg a modellek interpretálása a mély neurális hálózatok belső reprezentációjának jobb megértését segítik, az adatok és predikciók vizsgálata sokszor a gyakorlati alkalmazásokban jelentenek nagy segítséget.

Az interpretálhatóság egy további lehetséges felosztása a globális és lokális interpretálhatóság. A globális interpretálhatóság az adatok egy halmazára vonatkozó átfogó elemzést, míg a lokális egy adott mintának, és az ehhez tartozó modell működésének és predikciónak az elemzését jelenti.



1. ábra: gépi tanulás interpretálhatóságának taxonómiája.

1.3. Technológiák avulása és fejlődése

Az akadémiai és ipari szféra rendkívül nagy érdeklődésének köszönhetően a mélytanulás még az informatikán belül is kimagaslóan gyorsan fejlődik, szinte hónapról hónapra új, state-of-the-art megoldások látnak napvilágot. A tématerület sajátossága, hogy a modellezést inkrementális módon végezzük (lásd 3.1.-es pont), ezért a legújabb technikák mellett jellemzően a korábbi mélytanuló architektúrákkal is készítünk rendszereket a végső megoldás kialakítása során. Ezek gyökerei a 80-as, 90-es évekre

nyúlnak vissza. Ezért az igen gyors fejlődés mellett egy állandóság is jellemzi a mélytanulást.

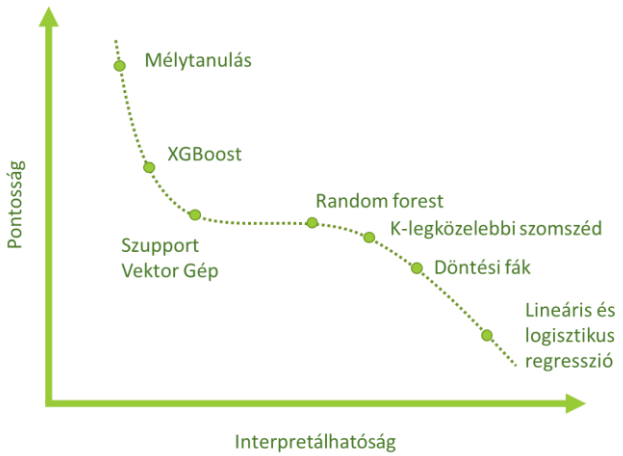
A mélytanuló eljárások interpretációjára rendelkezésre álló eszközök tárháza viszonylag korlátozott, a gyakorlatban elterjedt alapvető megközelítések nagy része jelen dokumentumban röviden bemutatásra kerül. A predikcióra és az adatra vonatkozó módszerek a modell architektúrájától függetlenek, ezért a mélytanuló technológia fejlődése mellett ezek az eszközök a jövőben is használhatóak lesznek. A modellre vonatkozó elemzések részben függenek a modell architektúrájától, de ezen alapelveket várhatóan újtípusú architektúrák esetén is használni lehet.

2. Mély neurális hálózatok interpretálhatósága

A gépi tanuló eljárások témakörben napjainkban a legnagyobb figyelmet talán a mélytanulás (deep learning) kapja. A deep learning rendszerek alapjait a mesterséges neurális hálózatok alkotják. Mesterséges neurális hálózatok kutatása a 40-es években kezdődött el McCulloch és Pitts elemi mesterséges neuronjával, amit számos lépcsőfokon keresztül a 60-as, 70-es években a többrétegű neuronhálók és a hibavisszaterjesztés (backpropagation) eljárás követett. A 80-as évek végén és a 90-es években született meg a konvolúciós neurális hálózat és a mai napig a legelterjedtebb rekurrens neurális hálózati struktúra, a Long Short-Term Memory (LSTM). A modern deep learning rendszerekben ezek a módszerek meghatározó szerepet játszanak, továbbá az elmúlt évtizedben számos tudományos és technológiai újdonság tette rendkívül hatékonnyá ezeket a gépi tanuló eljárásokat. Összehasonlítva más adatvezérelt algoritmusokkal a deep learning előnye, hogy alkalmas reprezentáció tanulásra, ami annyit jelent, hogy kisebb-nagyobb mértékben az adatokból tanulja az adatelőfeldolgozási módszert is, ami együttműködik a modellező eljárással - egy mesterséges neurális hálózaton belül. Más gépi tanuló eljárások esetén az adatelőfeldolgozást jellemzően szakértők végzik, és ehhez illesztik a modellező eljárást. Ilyen esetekben egy határon túl többnyire hiába növeljük az adatok mennyiségét, a modellek pontossága nem javul. Ezzel szemben mély neurális hálózatokban az adatmennyiség növelésével és a modellek méretének és felépítésének skálázásával sok esetben lehetséges tovább javítani a pontosságon. Az elmúlt években számos elméleti nóvum mellett a technológiai oldalon talán az a legnagyobb eredmény, hogy a deep learning rendszerek tanításához szükséges hardver infrastruktúra közel lineárisan

skálázódik. Ez annyit jelent, hogy az adatmennyiség növelésével és a modellek felépítésének skálázásával közel egyenesen arányos többlet számítási kapacitásra van szükség ahhoz, hogy a korábbival azonos idő alatt tudjuk elvégezni a tanításokat. Ez egyben azt is jelenti, hogy azonos adatmennyiség és modellstruktúra mellett a számítási kapacitás növelésével egy nagyságrendben lehet a tanításhoz szükséges időt csökkenteni.

Ma már minden technológiai óriásvállalat (pl. Google, Facebook, Microsoft, IBM, Amazon, Baidu, stb.) komoly deep learning kutató- és fejlesztőrészleggel, és az ezt kiszolgáló szoftver és hardver infrastruktúrával rendelkezik. Ezen cégek számos termékükben sikeresen alkalmaznak deep learning megoldásokat, és ezzel jelentős eredményeket, és egyben bevételnövekedést érnek el. Emellett ma már közép- és kisvállalatokra is egyre jellemzőbb, hogy mélytanuló eljárásokat építenek be termékeikbe. Számos szakértői vélemény szerint most következik a deep learning forradalom második hulláma, mely során a technológiai óriásoknál már sikeresen használt mélytanuló megoldásokat szélesebb körben, különböző területeken fogják alkalmazni (pl. pénzügyek, közszféra, közlekedés, nemzetbiztonság, gyártás, mezőgazdaság, stb.). Napjainkban már minden nagyobb vállalat és ország számára kiemelten fontos, hogy kialakítson egy adat és mesterséges intelligencia stratégiát, hogy megvizsgálja a deep learning lehetőségeit, és hogy sikeres prototípusokon keresztül a korábbi megoldásokon túlmutató deep learning alapú szoftvermegoldásokat fejlesszen.



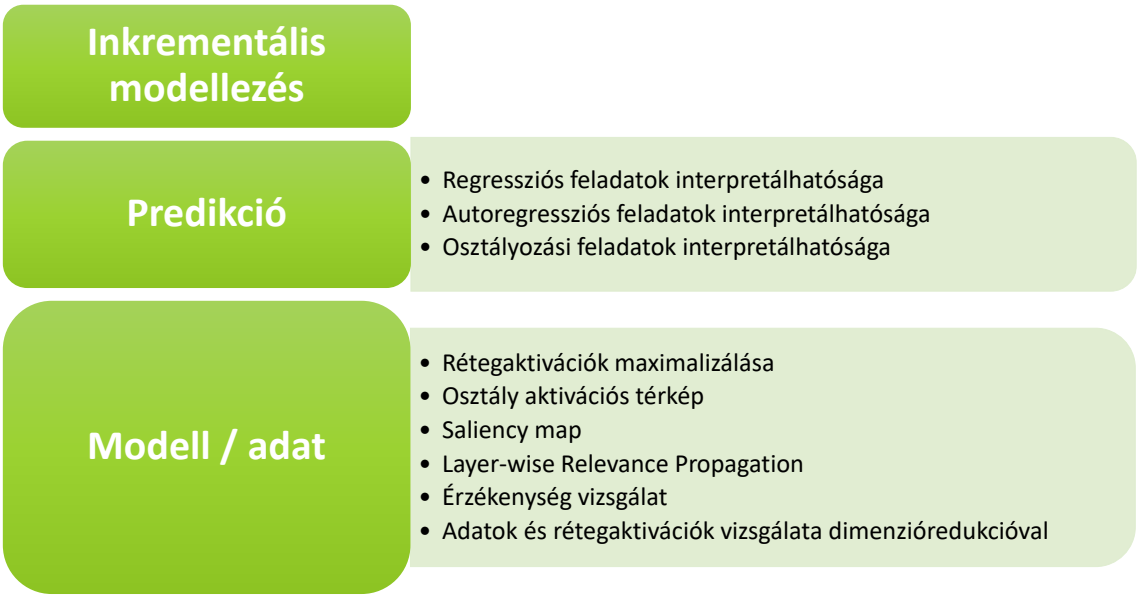
2. ábra: a pontosság és az interpretálhatóság kapcsolata gépi tanuló eljárások esetében.

A mély neurális hálózatok a gépi tanuló eljárások között a legösszetettebb módszerek közé tartoznak. A mélytanuló eljárások *fekete-* és *szürkedoboz* modellek.

Adatbázis függvényében más gépi tanuló eljárások mellett képesek lehetnek nagyobb pontosság elérésére – mely nagyobb pontosság kevésbé interpretálható megoldásokhoz vezet (lásd 2. ábra). Népszerűségük folyamatosan nő. Ezért kiemelten fontos ezen módszerek működését és döntéseit minél jobban megérteni.

3. Módszerek

Az alábbiakban áttekintjük a gyakorlati szempontból legfontosabbnak tartott interpretációs módszereket mélytanulás esetében. Ezek közül több az általános megfontolás, mely más gépi tanuló eljárások esetén is alkalmazható (3.1-3.4 alfejezetek), míg a többi (3.5-3.10 alfejezetek) kizárólag mélytanulás esetén alkalmazhatóak. A bemutatásra kerülő módszerek besorolását az alábbi, 3. ábra foglalja össze.



3. ábra: interpretációs lehetőségek mélytanuló eljárások esetén.

3.1. Inkrementális modellezés

Amennyiben a modellezést fejlett, mélytanuló modellekkel kezdjük, és kapunk egy eredményt, fontos, hogy ezt az eredményt értelmezni tudjuk. Önmagában egy négyzetes hiba, keresztentrópia érték, pontosság, vagy bármilyen egyéb metrika nem elég beszédes ahhoz, hogy meg tudjuk mondani, hogy van-e hozzáadott értéke a mélytanuló modellnek egyéb, gépi tanuló eljárásokkal szemben. Lehetséges, hogy minden más technológián túlmutat, de az is lehet, hogy a sok százezer, vagy millió tanulható paraméteres

mélytanuló modellünk mögött valójában egy „túlbonyolított” lineáris regresszió fut le. Viszont ilyen esetben csak problémánk fakad a mélytanulás alkalmazásából:

- elveszítjük az egyszerűbb modellek magyarázhatóságának a lehetőségét,
- a sokparaméteres, adott adatbázishoz túlméretezett modellek sok esetben rosszabb teljesítmény nyújtanak, mint az egyszerűbb változatok.

Ezért az alábbi táblázatban megadott inkrementális modellezési módszertant javasolt követni. A táblázatban megadott modellezési lépéseket elvégezve a mélytanuló eljárások során kapott pontosságot már el tudjuk helyezni a különböző komplexitású modellek között, törekedve a legjobb pontosságra, a legkisebb hibára.

1. táblázat: Javasolt inkrementális modellezési lépések mélytanulás esetén.

Lépés	Modell	Magyarázat
1.	Baseline, szabály alapú	Amennyiben van lehetőségünk az adott, modellezendő folyamatot szabály alapon leírnunk, akkor érdemes ezzel kezdeni. Ez a lépés főleg abban az esetben javasolt, ha korábban nem automatizált folyamatot szeretnénk gépi tanulás, illetve mélytanulás segítségével modellezni.
2.	Lineáris modellek	Elsődlegesen lineáris és logisztikus regresszió, illetve ezek különböző változatai (Lasso, Ridge, Elastic Net, stb.). Ezen módszerek fehérdoboz modellek, jól magyarázhatóak, és referenciaként jól használhatóak. Célunk, hogy magasabb pontosságot érjünk el vele, mint a baseline, szabály alapú modellekkel, és hogy a komplexebb modellek teljesítményét ezzel össze tudjuk mérni.
3.	Bagging technikák	A lineáris modelleket követően javasolt különböző, un. bagging technikák (pl. döntési fák, random forest) segítségével modelleznünk a folyamatot. Ezek a modellek továbbra is jól magyarázhatóak, például a döntési fák vizsgálatával, illetve a korábbi tanulmányunkban megadott módszerekkel. Mindamellett ezek a modellek már képesek nemlineáris összefüggések modellezésére, ezért bonyolultabb folyamatok esetén jobb eredményt várhatunk az első két megközelítésnél.

4.	Boosting technikák	Elsődlegesen az eXtreme Gradient Boosting (XGBoost) módszert javasoljuk ebben a lépésben használni, mely modellező képessége túlmutat a korábbi három lépésben használt módszereknél, viszont magyarázhatósága már elmarad ezektől.
5.	Mély neurális hálózatok	A mélytanulás alapú modellek az itt felsoroltak közül a legösszetettebbek, a korábbiaknál nehezebben magyarázhatóak, viszont a korábbiaknál jobban skálázhatóak – a modell méretét az adatbázis méretéhez lehet igazítani, legyen szó akár pár megabyte-os, vagy többszáz terrabyte-os adatbázisról. A mély neurális hálózatok esetén is jellemzően kisebb méretű modellekkel érdemes a modellezést elkezdeni, és az így elért eredményeket összevetni az 1-4 lépésben kialakított modellek teljesítményével. Ezt követően lehet növelni a neuronháló méretét, komplex kapcsolatokat (pl. residual, highway, skip) és új rétegtípusokat (pl. dropout, batchnorm, layernorm, groupnorm, stb.) vezethetünk be több lépésben.

3.2. Regressziós feladatok interpretálhatósága

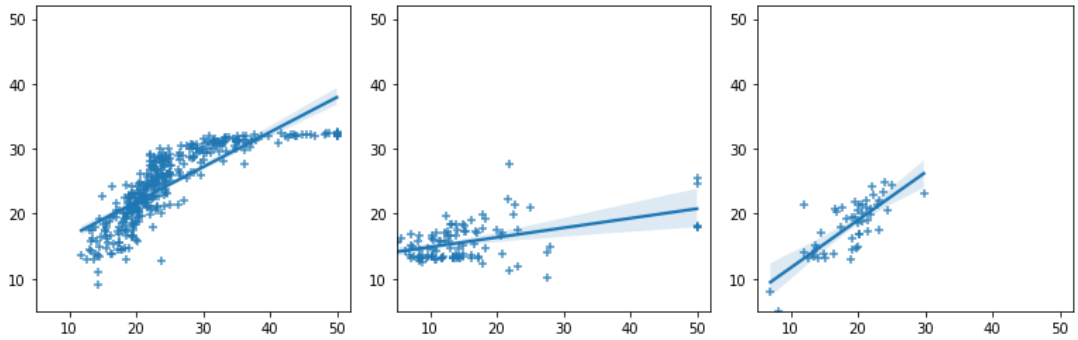
Regresszió esetén a célunk egy vagy több *érték* predikciója. A mélytanuló eljárásoknál bevett módon tanító, validációs, teszt adatbázisokra bontjuk a teljes adathalmazt:

- a *tanító adatbázis* segítségével keressük a háló optimális súlyait,
- a *validációs adatbázison* mérjük vissza tanítás során a hálózat általánosító képességét, és jellemzően, amikor adott iteráción keresztül nem javul a validációs adatbázison mért célmetrika (pl. négyzetes hiba), akkor állítjuk le a tanítás,
- végül pedig a tanító és validációs adathalmaztól független *teszt adathalmazon* mérjük le a megoldás általánosító képességét.

A regresszió jóságának vizsgálatának, és így a predikció interpretációjának egyik alap módszere a valós értékek és a predikciók viszonyának vizsgálata. Ezt gyakran tesszük meg vizuálisan a regressziós ábra segítségével. A 4. ábrán egy példát látunk ingatlan-ár előrejelzésre a Bostoni lakásár adatbázison [5]. Az ábrán a tanító, validációs és teszt

halmazra a predikciók értéke látható. Az X tengely az adatbázisban szereplő célértékeket mutatja, míg az Y tengely a predikciókat. Az ábra alapján megállapítható, hogy mely tartományokban prediktál jellemzően alacsonyabb vagy magasabb értéket a modell a különböző adathalmazok esetén. Például a tanító adathalmazon 10-20 közötti predikció esetén nagyjából átlagosan ugyanannyit téved mindkét irányba, 20-30 közötti értékeknél inkább „felfele” téved, tehát nagyobb értéket predikál, míg 30 felett inkább lefele, azaz a maximális predikció ebben az esetben 30 körül van, magasabb értékek esetén is. A validációs- és teszthalmazok esetben 10-20 között hasonlóan viselkedik a modell, míg 20 felett a tévedés az alacsonyabb értékek irányába nagyobb, 30 felett egyre markánsabb mértékben. Ez alapján nagyvonalakban a következő szempontokat vehetjük figyelembe a modell éles futtatása során:

- amennyiben a predikció 10-20 közé esik, akkor elfogadjuk,
- 20-30 között szakértőknek célszerű lehet megvizsgálni, hogy nem túl alacsony-e a jósolt érték,
- 30-hoz közeli értékek esetén pedig fontosnak tartjuk a szakértői felügyeletet.



4. ábra: célérték (X tengely) – predikció (Y tengely) regressziós ábrák a tanító (bal), validációs (közép) és teszt (jobb) adatbázisok esetén. A használt neurális hálózatnak egy teljes összeköttetésű rejtett rétege volt, és a bostoni lakás árak adatbázissal lett tanítva.

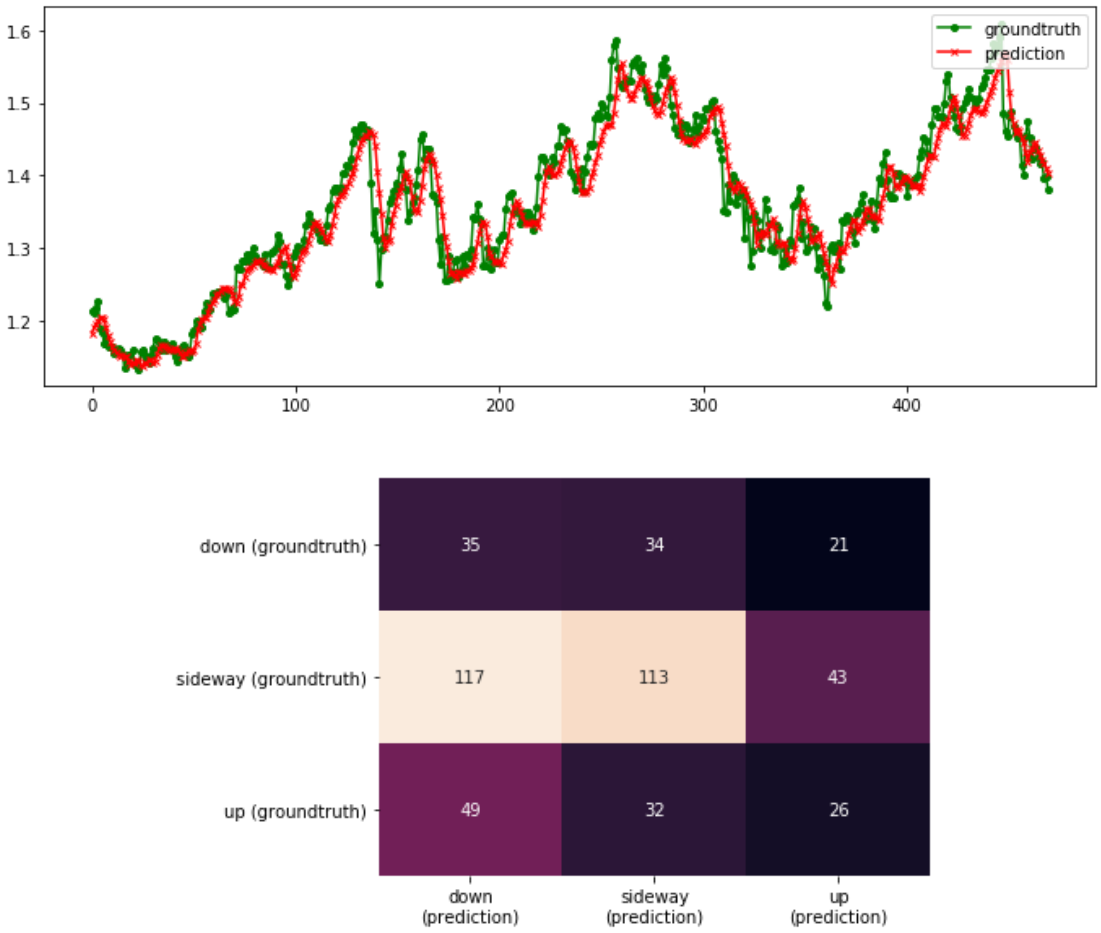
3.3. Autoregressziós feladatok interpretálhatósága

Az autoregresszív modellek a regressziós modelleknek egy speciális ága, a felügyelet nélküli tanulás önfelügyelt ágához tartoznak. Autoregresszív esetben célunk a szekvenciális (sokszor idősoros) adatfolyam egy korábbi állapotából egy jövőbeli értékét előrejelezni. Jellemzően ezt olyan mélytanuló modellek segítségével szoktuk megtenni, amelyek képesek a tér és/vagy időbeli összefüggések leírására. Többek között ilyenek

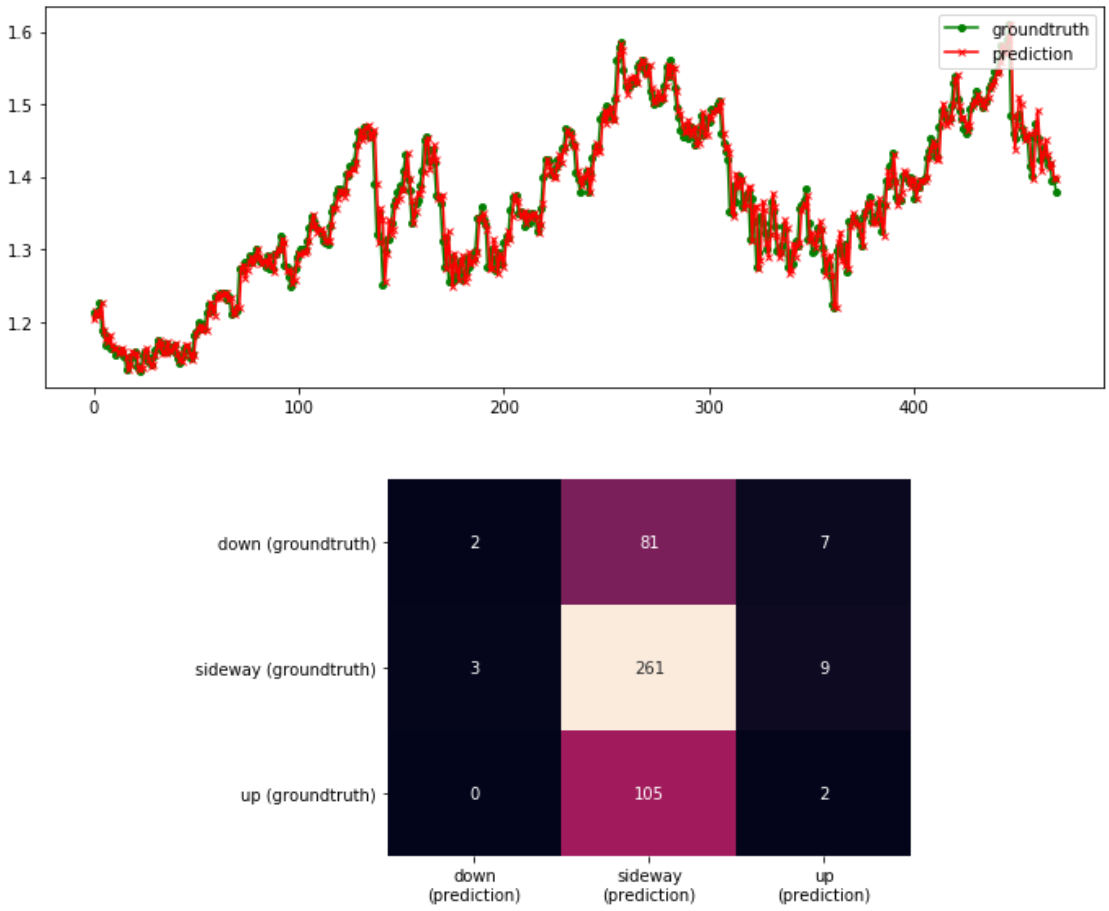
például a rekurrens neurális hálózatok (beleértve a Long Short-Term Memory [6] modelleket), a konvolúciós neurális hálózatok [4], és az önfigyelemre épülő transzformer modellek [7].

Az autoregresszív modellek kiértékelése olyan szempontból speciális, hogy lehetséges, hogy alacsony hiba mellett is nagyon alacsony, vagy nincs valódi predikciós képessége. Ezért az autoregresszív modellek predikciójának interpretációja során érdemes a célváltozó és a predikció értékeit egy ábrán vizsgálni, kiegészítve egy három osztályos tévesztési mátrixal. A három osztály: felfele, oldalazó, lefele mozgás az előző értékhez képest. Az oldalazásnak pedig tématerülettől függő határértékeket adunk meg.

Az 5-ös és 6-os ábrán egy-egy példaelemzés látható. Az 5-ös ábra esetén a tévesztési mátrixot vizsgálva láthatjuk, hogy többször tévedett „lefele” a predikció, mint a másik két irányba. Ezt, és a konkrét értékeket figyelembe véve láthatjuk, hogy bár egy komplex architektúrát használtunk ez esetben, mégis, a neuronháló nagyjából egy mozgó átlag modellt tudott megtanulni. A 6-os ábrán ugyanezen idősor predikcióját látjuk temporális konvolúciós neuronhálóval. Ez esetben a tévesztési mátrixból láthatjuk, hogy szinte minden esetben oldalazást jelzett előre a modell. A konkrét értékeket vizsgálva „jónak” tűnhet a predikció, azonban jobban megfigyelve az ábrát azt láthatjuk, hogy a modell nem csinált egyebet, mint a bemenetnek az utolsó értékét megismételte predikcióként (ezt *naív predikciónak* is hívjuk).



5. ábra: GOOGL részvény előrejelzés egy dimenziós konvolúciós neurális hálózattal, predikció vizsgálata az idősoron (MSE=0.00117) és tévesztési mátrixon



6. ábra: GOOGL részvény előrejelzés temporális konvolúciós neurális hálózattal, predikció vizsgálata az idősoron (MSE=0.00052) és tévesztési mátrixon

Természetesen egyik modellnek sincs igazi modellező képessége, ami arra enged következtetni, hogy az adott adathalmazban a múlt nem írja le a jövőt. Mivel ezekben a példákban napi szintű árfolyamadatokat használtunk fel, ezért ez nem meglepő: pusztán napi árfolyam középértékéből a következő nap árfolyamát adott részvény esetén nem lehetséges a mai modellekkel általánosan leírni ilyen kevés adat esetén.

3.4. Osztályozási feladatok interpretálhatósága

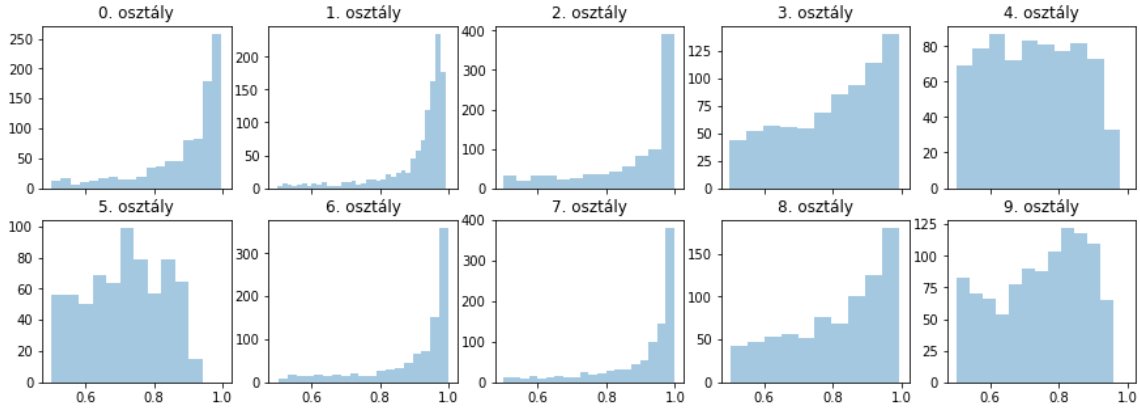
Osztályozási feladatok esetén a mély neurális hálózatok kimenetén sigmoid vagy softmax aktivációs függvények találhatóak. Sigmoid-ot bináris (2 osztályból választjuk ki, hogy a bemenet melyikhez tartozik) vagy sok-címkés (n osztályból választjuk ki, hogy melyik m osztályhoz tartozik a bemenet, ahol $m \leq n$) osztályozási feladatok esetén használunk, míg softmax-ot sok-osztályos esetben (n osztályból adjuk meg azt az 1-et, amihez a bemenet tartozik). Az előnye ezeknek az aktivációs függvényeknek, hogy a kimenetük 0…1-ben

korlátos, így valószínűségként lehet kezelni a kimeneti értéket. A predikciós valószínűségeket sigmoid esetén legtöbbször a 0.5-ös döntési küszöb alapján sorolják osztályokba, míg softmax esetén a legmagasabb érték alapján. Éles rendszerek esetén azonban érdemes a valószínűségeket is figyelembe venni, és ez alapján eldönteni, hogy egy predikcióban mennyire biztos a neurális hálózat.

Amennyiben a predikció értéke 1-hez (pl. *>0.98, >0.99, >0.999*) vagy 0-hoz (pl. *<0.02, <0.01, <0.001*) közeli, a döntést – alkalmazástól függően – feltétel nélkül elfogadjuk. Ha a predikció egyre jobban közelíti a 0.5-ös döntési küszöbhöz, akkor pedig más modellek, esetleg szakértők általi felülbírálat lehet szükséges.

Gyakorlati megoldás, hogy a helyes predikciók esetén megvizsgáljuk a teszt adatbázison, hogy milyen valószínűségeket ad a modellünk, és a későbbi predikciók értékeit ehhez viszonyítjuk. Amennyiben egy adott mintához tartozó predikció a teszt adatokon mért valószínűségektől jelentősen eltér, akkor azt feltételezhetjük, hogy a modell *bizonytalan* a döntésében, amit a valós folyamat függvényében kezelni kell.

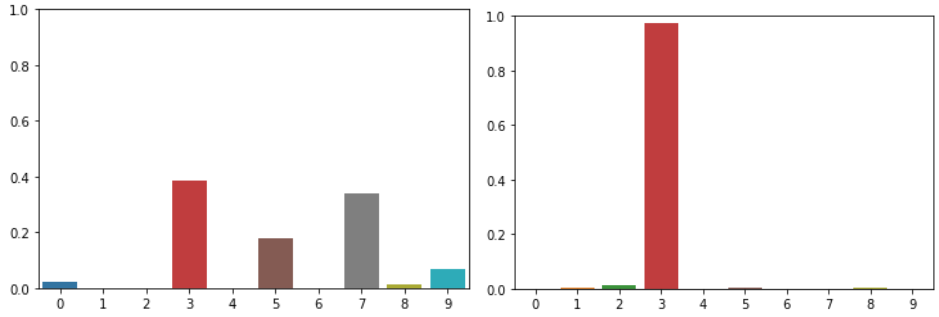
Az alábbiakban látunk egy sok-osztályos példát, *n=10* osztállyal. A 7-es ábra mutatja be, hogy az egyes osztályok esetén a teszt adathalmazon végzett predikció milyen értékeket adott. Itt az 1.0 azt jelenti, hogy 100%-ig biztos a döntésben, és az alacsonyabb értékek pedig, hogy egyre inkább bizonytalanná válik a modell. Az ábrán jól megfigyelhető, hogy a 0, 1, 2, 6, 7 osztályok esetén sokkal biztosabb a modell a döntésében, mint például a 4-es és 5-ös osztályoknál.



7. ábra: valószínűségi eloszlások egy 10 softmax kimenetű neurális hálózat esetén egy adott teszt halmazra (10000 minta). Az egyes ábrák az egyes kimenetek 0.5 feletti értékeit mutatják.

Ezen ábrák (és természetesen a mögöttes numerikus értékek vizsgálatával) megadhatjuk, hogy egy éles rendszer futása során az adott osztályra vonatkozó predikciót milyen valószínűségi érték esetén tekinthetjük „biztos”, és mikor „bizonytalan” predikciónak. Bizonytalan esetben a hibázásnak nagyobb az esélye, ezért a határértékek meghatározása minden esetben a modellezett adatoktól függ.

Ezzel összefügg az egyes mintákra adott predikciók vizsgálata. Amely mintáknál, a fentebbi eloszlások figyelembe vétele mellett, kiugró értékeket látunk, azt erős predikciónak vesszük, míg minél több, hasonló érték szerepel az az adott minta predikciójában, ez annál inkább bizonytalannak tekinthető. Erre mutat példát a 8-as ábra, szintén tíz osztályos predikció esetében. Mint látható, mindkét esetben a legmagasabb valószínűsége a 3-as osztálynak van (piros oszlop), azonban a bal oldalsó esetben a 7-es osztály nagyon hasonló, kicsit kisebb értékkel rendelkezik, és az 5-ös és 9-es osztály valószínűségei sem elhanyagolhatóak. A jobb oldali esetben, amely egy másik teszt mintához tartozik, a 3-as osztály valószínűsége közel 100%, a többi osztályé pedig elhanyagolható. Így ez a predikció sokkal erősebbnek tekinthető.



8. ábra: valószínűségi értékek egy 10 softmax kimenetű neurális hálózat esetén egy adott mintára. Bal: bizonytalan predikció, jobb: erős predikció.

3.5. Rétegaktivációk maximalizálása

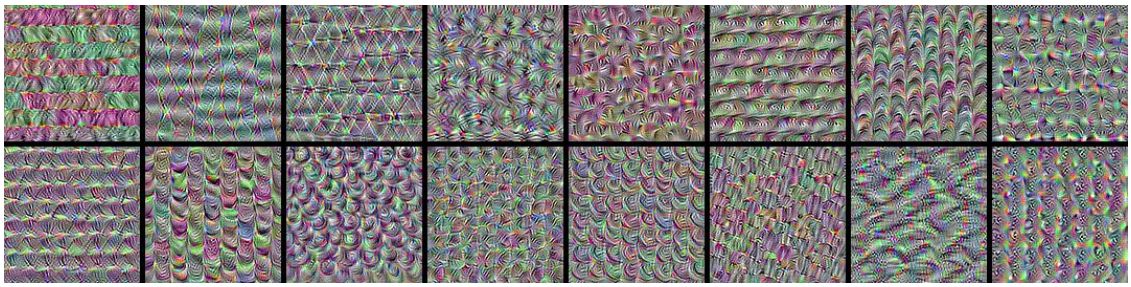
A mélytanuló modellekben kialakított jellemzők vizsgálatának egy lehetséges módszere a rétegaktíválciók maximalizálása. Ez annyit jelent, hogy a kezdetben zajszerű bemenetet úgy módosítjuk, hogy a neurális hálózat adott neuronjának vagy neuronjainak a kimenete maximális legyen. Az eljárás a következő lépésekből áll:

1. A mélytanuló hálózat betanítása és a súlyok befagyasztása.
2. Adott neuron(ok) kiválasztása.

3. Hibafüggvény megadása a kiválasztott neuron(ok) aktivációjára. (Például az aktivációk átlaga.)
4. Bemeneti mintaként normál eloszlású zajt megadása.
5. Kiszámoljuk a gradienseket a hibafüggvény és a bemenet tekintetében, és ezek arányos („*tanulási rátaszorosát*”) hozzáadjuk a bemeneti zajhoz. Ezt a lépést addig folytatjuk, ameddig egy értelmezhető mintázat nem rajzolódik ki a bemeneten.

Az eljárás első változatát konvolúciós mély neurális hálózatok esetén a gépi látásban alkalmazták [8]. A módszert képi és szekvenciális (pl. idősorok) esetén értelmezhető legkönnyebben. TensorFlow Keras alapú implementációjának lényegi része mintegy 10-15 sorból megvalósítható [9].

A generált bemeneti adatok alapján lehet vizsgálni, hogy a neurális hálózat adott rétegei milyen különböző bemenetek esetén a „legaktívabb”, azaz milyen jellemzőket tanult meg a modell. A 9-es ábra egy példát mutat képi adatokon tanított mély konvolúciós hálózat e módon történő vizsgálatára. Az ábrán a különböző rétegek által megtanult jellemző mintázatok láthatóak.



9. ábra: Mély konvolúciós neuronháló különböző szintű rétegeinek maximalizálása alapján generált bemeneti minták (módosítva [9] alapján).

3.6. Osztály aktivációs térkép

Az osztály aktivációs térképek (Gradient Class Activation Map, Grad-CAM, [10]) vizsgálata szintén egy gradiens alapú eljárás, melyet konvolúciós neurális hálózatokban alkalmazunk gépi látás, szekvencia és idősor modellezési feladatokban. A célja, hogy a bemeneti adatokon kijelölje azokat a régiókat, ami alapján a modell az osztályozási döntést meghozta. Az eljárás lépései a következőek:

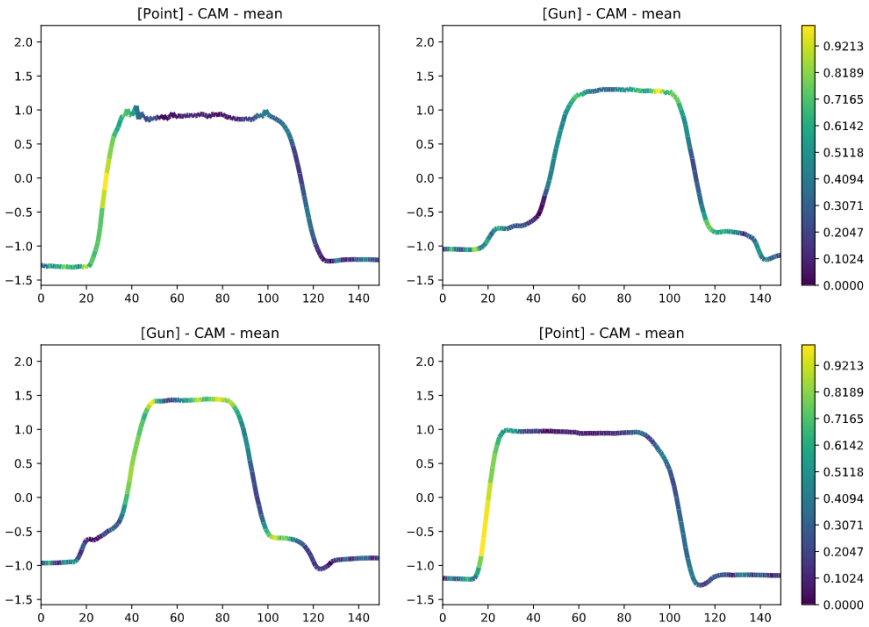
1. Mély konvolúciós neurális hálózat betanítása és a súlyok befagyasztása.

2. Adott, végső konvolúciós réteg aktivációs térképének kiszámítása adott bemenet mellett.
3. A vizsgálandó kimeneti osztályhoz tartozó gradiensek számítása a kiválasztott aktivációs térkép függvényében.
4. A kiszámolt gradiens átlagolása a konvolúciós szűrő dimenziója mentén.
5. Az átlagolt gradiensek összeszorzása az aktivációs térképpel.

Így egy hőtérkép jellegű adatot kapunk, melyet a bemenethez igazítva (ha szükséges, átskálázva) meg tudjuk adni, hogy a bemenet mely részei felelősek jobban és kevésbé az adott döntésért.

A 10-es ábra egy példát mutat idősoros adatok e módon történő elemzésére. Az ábrák két osztályhoz tartozó idősorokat tartalmaznak, ahol a kézmozgás horizontális pozíciójának idősorából szeretnénk megmondani, hogy amikor egy személy előveszi a zsebéből a kezét, akkor van-e benne pisztoly (nincs - *Point: bal felső és jobb alsó,* van - *Gun: jobb felső és bal alsó*). Az ábrán a világosabb színek jelölik, hogy mely részek voltak nagyobb hatással az adott osztály melletti döntésre. Az ábrát vizsgálva megállapítható, hogy a *Point* osztály mellett a felfutó élek mintája miatt, míg a *Gun* osztály mellett részben a más alakzatú felfutó él, illetve a lefutás előtti rész miatt döntött a modell.

A módszer használatát Python alapon idősoros adatokra az alábbi honlap mutatja be [11].



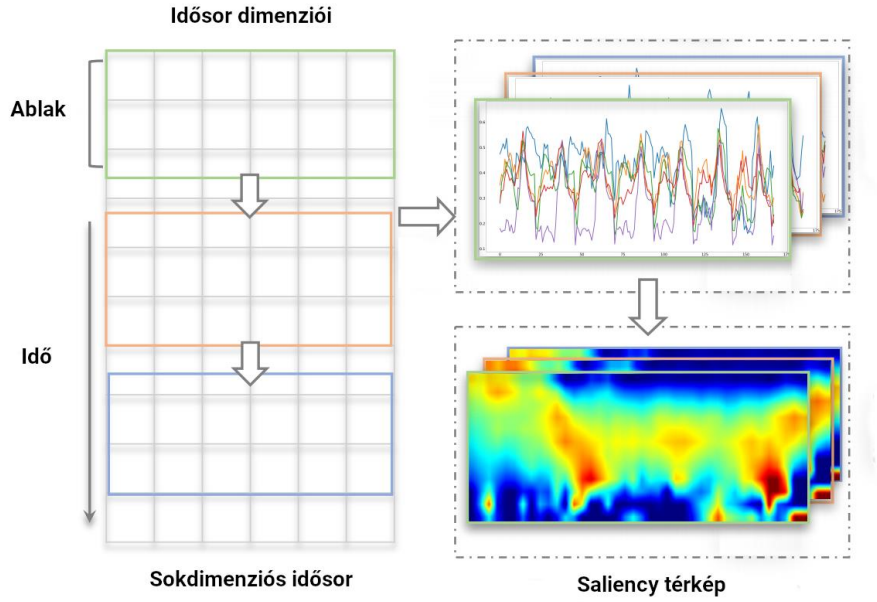
10. ábra: Grad-CAM ábrák idősor, bináris klasszifikáció esetén (módosítva [11] alapján).

3.7. Szembetűnőségi (saliency) térkép

A szembetűnőségi térképek fogalmát eredetileg a gépi látásban vezették be [12]. A térkép célja azt megadni, hogy az adathoz tartozó egyes jellemzők (pl. adatpontok, pixelek, színek, stb.) mennyire járulnak hozzá a kép információtartalmához. Mélytanulás esetén a szembetűnőségi térkép segítségével a mély neurális hálózat döntéseiről lehet következtetéseket levonni [13], mint például a predikcióhoz a bemenet mely részei járulnak hozzá leginkább, vagy például a színcsatornák közül melyik színek a dominánsak. A szembetűnőségi térkép meghatározása mélytanuló rendszerekben az alábbi lépések szerint történik:

1. Modell betanítása és súlyok befagyasztása.
2. Egy adott bemenetre predikció (pl. osztályozás) és hiba számítása
3. A bemenetre vonatkozó gradiensek kiszámítása.
4. A gradiensek maximumának vagy átlagának számítása adott tengely mentén (pl. színek).

A megoldás nem csak képek esetén használható, hanem a fenti eljárás azonos lépései idősorokra is alkalmazhatóak [14]. Ez esetben az idősor modellezését konvolúciós neurális hálózatokkal végezzük.



11. ábra: szembetűnőségi térkép számítása sokdimenziós idősorok esetén (módosítva [14] alapján).

Szembetűnőségi térképeket a Keras mélytanuló keretrendszer esetén a keras-vis csomag segítségével tudunk például generálni [15] .

3.8. Layer-wise Relevance Propagation (LRP)

Az előző módszerekhez hasonlóan a rétegenkénti fontosság terjesztés (Layer-wise Relevance Propagation, LRP) a kimeneti rétegből kiindulva a döntés okait vezeti vissza a bemenetre. Tehát ezzel vizsgálhatjuk, hogy a modell a bemenet mely részei alapján hozta meg a döntést [16]. Az LRP fontos tulajdonsága, hogy a bemenetre leképzett relevancia térkép értékeinek az összege megegyezik a vizsgált kimenet értékével. Az LRP eljárás alapvető elgondolását a következő képlet írja le:

$$R_j = \sum_k \frac{a_j w_{jk}}{\sum_{i=0}^{j} a_i w_{ik}} R_k$$

, ahol R a relevancia térkép értékei, j és k két egymást követő réteg neuronjai. Az R_{-1} (kimeneti réteg relevancia térképe) ismert, ami maga a predikció. A képletben az a az adott neuron aktivációja, w pedig a két neuron közötti súly értéke. A fentebbi képlettel iteratívan, rétegenként haladva a kimenettől visszafele tudjuk számolni az adott réteg, neuron, vagy bemenet relevancia térképét.

A képletben a nevező adja meg, hogy a j-edik neuron milyen mértékben befolyásolja a k-adik neuront, míg a számláló biztosítja, hogy a relevancia térkép értékeinek az összege egyenlő legyen a kimenettel. A képletben a külső szumma pedig megadja, hogy a j-edik neuron relevanciája a következő réteg összes k neuronjának a súlyozott relevanciája legyen (ahol a súlyt az adja meg, hogy a j-edik neuron mennyire befolyásolja a következő réteg neuronjait).

A megoldást igaz, hogy képekre dolgozták ki (konvolúciós hálók esetén), de a korábbiakhoz hasonlóan szekvenciális adatokra és idősorokra is alkalmazható [17], illetve rekurrens rétegekre is kiterjeszthető [18].

Az LRP működését képek esetén jól szemlélteti az alábbi online demonstráció: https://lrpserver.hhi.fraunhofer.de/image-classification

Python alapon a Keras mélytanuló keretrendszerbe az LRP a következő hivatkozás alapján lehet bevezetni [19].

3.9. Érzékenység vizsgálat

Az érzékenység vizsgálat egy általános módszer gépi tanuló algoritmusok elemzésére, mélytanulás esetén is rendkívül hatékony. Az érzékenység vizsgálat azt jelenti, hogy a bemenet adott dimenzióját megváltoztatjuk, és az így kapott predikció értékét összevetjük a módosítás nélküli predikció értékével. Amennyiben jelentős eltérés tapasztalható, akkor az állapítható meg, hogy az adott bemeneti dimenzióra érzékeny a modell, míg ha a kimenetet az adott jellemzőhöz tartozó bemenet módosítása nem befolyásolta, akkor arra kevésbé vagy nem érzékeny a modell. Így a bemeneteket között egy rangsort tudunk felállítani adott adatmintahalmazra. A bemenetek módosítására számos lehetőségünk van, például:

- Jellemző értékének mintavételezése egyenletes eloszlással: az egyes jellemzők eloszlása általában különböző, és a legritkább esetben egyenletes. Ezért érdemes lehet vizsgálni, hogy milyen hatással van a predikcióra, amennyiben az adott jellemzőt a teljes értéktartományon belül mintavételezzük egyenletes eloszlással. Példa: a bemenő adataink banki tranzakciók. Az egyik jellemző az átutalás összege. Az átlagos átutalás összege 5.000-50.000 Ft között van, és az adatok nagy része erre az intervallumra korlátozódik. Tegyük fel, hogy az adatbázisban a minimum érték 1.000 Ft, a maximum pedig 10.000.000 Ft. Ebben az esetben a fentiek szerint minden más jellemző értékét változatlanul hagyjuk az adatbázisban, egyedül az átutalás értékét helyettesítjük be az 1.000…10.000.000 Ft intervallumból egyenletes eloszlással mintavételezett értékekkel.
- Jellemző értékeinek permutációja: ebben az esetben a minták között felcseréljük az adott minták értékeit. Tehát például az első minta adott jellemzőjének az értékét kicseréljük a második minta jellemzőjének az értékével. Az eljárás előnye, hogy nem változtatjuk meg a jellemző értékeinek eloszlását – továbbra is valós értékekkel dolgozunk.
- Jellemző értékeinek törlése: szintén gyakran használt megközelítés, amikor a jellemző értékeit nullára állítjuk, és ezáltal azt szimuláljuk, mintha hiányozna az adott minta esetén az adott jellemző. Fontos ez esetben arra figyelnünk, hogy a valós értéktartományban a nulla ne legyen értelmezett (amennyiben mégis értelmezett, akkor más értéket kell választani).

Mindhárom esetben, ha az értékek módosítása után jelentős mértékben romlik a predikció, akkor az adott jellemző fontosnak tekinthető, Amennyiben a predikció nem változik meg szignifikáns mértékben, akkor az adott jellemzőre nem érzékeny a betanított modell.

Az érzékenység vizsgálatot elvégezhetjük strukturált (pl. tabuláris) és nem strukturált (pl. idősor, képek, videók) adatokon. Nem strukturált adatok esetén az adott mintának egy részét módosítjuk a fenti eljárások szerint. Például idősorok és képek esetén „kitakarjuk" (nullára állítjuk) az adott minta különböző részeit – mely részeket véletlenszerűen, vagy csúszó ablakos módszerrel választhatunk ki. Így az osztály aktivációs térkép mellett ez is egy interpretációs lehetőség annak meghatározására, hogy az adat mely részei járulnak hozzá jobban, illetve kevésbé a predikcióhoz [20].

3.10. Adatok és rétegaktivációk vizsgálata dimenzióredukcióval

Dimenzióredukciós technikákkal is érdemes lehet akár magukat az adatokat, akár a neurális hálózat adott rétegéhez tartozó aktivációkat vizsgálni. Ilyen célokra napjaink leghatékonyabb technikái a t-SNE (t-distributed Stochastic Neighbor Embedding) [21] és UMAP (Uniform Manifold Approximation and Projection) [22]. Ezek sztochasztikus nemlináris módszerek. A korábbi dimenzióredukciós módszerek közül a legismertebb a PCA (Principle Component Analysis, főkomponens elemzés) [23], mely a jellemzők számát úgy csökkenti, hogy a variancia a maximális maradjon.

Az adatok vizsgálata során a dimenzió redukciós eljárásokkal gyorsan meg tudjuk állapítani, hogy van-e valamilyen kiindulási mintázat az adatokban, ami erősíti annak a lehetőségét, hogy lehetséges lesz gépi tanulással összefüggéseket találni az adatokban. A rétegaktivációk dimenzióredikciójával pedig azt vizsgáljuk, hogy a neurális hálózat adott rétege milyen módon alakította át a bemeneteket a célváltozó függvényében.

Dimenzióredukciós eljárások során az adatok és modellek interpretálhatósága céljából a sokdimenziós teret kettő vagy háromdimenzióban jelenítjük meg, és az adatoknak megfelelő pontokat a térben a célváltozó szerint színezzük be.

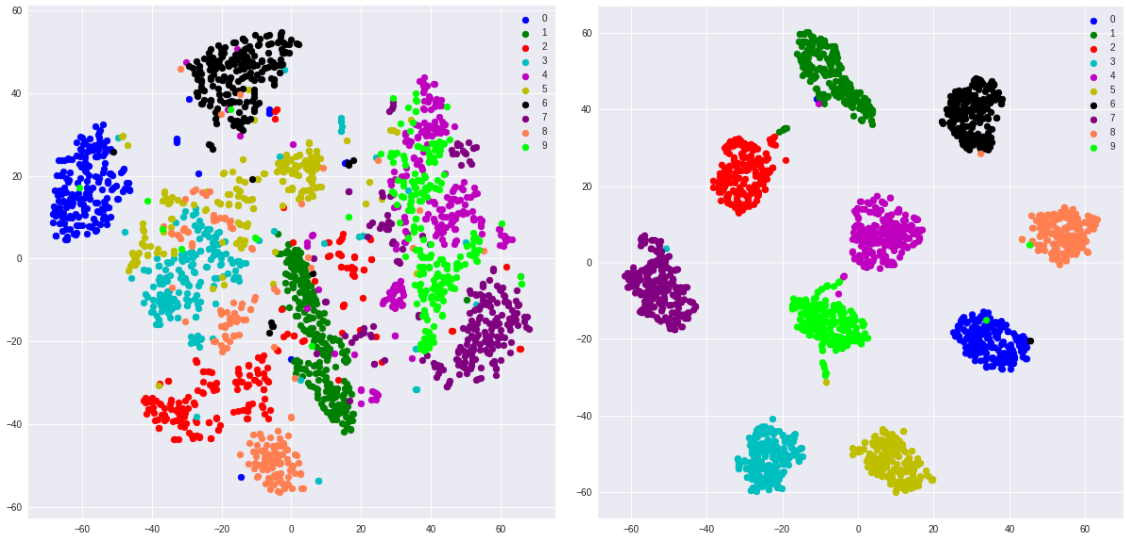
Amennyiben az adatokat vizsgáljuk, akkor az eljárás a következő lépésekből áll:

1. Adatok egy reprezentatív részhalmazának betöltése.

2. Dimenzióredukciós eljárás (kettő vagy három dimenzióra) futtatása a betöltött adatokon.
3. Az eljárás eredményeként előálló két- vagy háromdimenziós tér megjelenítése, az adatpontok színezése a célváltozó függvényében.

A rétegaktivációk esetén a fenti lépések az alábbiak szerint egészülnek ki:

1. Neurális hálózat betanítása.
2. Adatok egy reprezentatív részhalmazának betöltése.
3. A neurális hálózat egy részhálózatának képzése, ahol a kimenet a vizsgálni kívánt réteg.
4. Predikció a betöltött adatokon.
5. A vizsgálni kívánt réteg aktivációján a dimenzióredukciós eljárás (kettő vagy három dimenzióra) futtatása.
6. Az eljárás eredményeként előálló tér megjelenítése, az adatpontok színezése a célváltozó függvényében.



12. ábra: t-SNE dimenzióredukciós algoritmus futtatásának az eredménye. Bal: nyers adatokon (764 dimenzió). Jobb: neurális hálózat kimenethez közeli réteg aktivációján (128 dimenzió), ugyanazokon az adatok felhasználásával. A színek a bemenő adatokhoz tartozó osztályokat jelölik.

A 12. ábra mutat egy példát arra, hogy egy jól modellezhető probléma esetén hogyan néznek ki az adatok két dimenzióban, illetve egy betanított hálózat hogyan formálja ezeket át a minél pontosabb modellezés céljából. Az ábra bemutatja, hogy már a

kiindulási adatok is mintázatot mutattak, a betanított háló adott rétegaktivációja pedig arról tanúskodik, hogy a hálózat korábbi része hatékonyan nyeri ki és formálja a jellemzőket, a későbbi modellezés (ez esetben osztályozás) céljából.

A t-SNE és UMAP algoritmusok elérhetőek a TensorBoard-on belül a Projector felületen [24]. Ennek használatához javasoljuk a tensorboardX Python csomagot [25]. Ezen túl mindkét csomag elérhető különálló Python modulként [26,27].

4. Összefoglalás

A tanulmányban ismertetett módszereket kutatási és ipari projektekben gyűjtött tapasztalatink alapján összegeztük. Több ismertetett módszernek is léteznek különböző változatai, melyek célja jellemzően megegyezik az eredeti módszer céljával. Ezért, ha egy adott projektben meghatároztuk, hogy az adatokat, predikciót vagy a modellt milyen szempontok alapján szeretnénk elemezni, akkor a tanulmányban ismertetett módszerek különböző változatait (amennyiben léteznek) is érdemes lehet alkalmazni.

Az ismertetett módszerek nem csak a megoldások interpretálhatóságában segítenek, hanem segítségükkel jobban elmélyíthetjük a gyakorlati tudásunkat is az adatok és a mély neurális hálózatok világában.

Irodalomjegyzék

[1] Hossin, M., & Sulaiman, M. N. (2015). A review on evaluation metrics for data classification evaluations. International Journal of Data Mining & Knowledge Management Process, 5(2), 1.

[2] Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1985). Learning internal representations by error propagation. California Univ San Diego La Jolla Inst for Cognitive Science.

[3] Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. arXiv preprint arXiv:1409.3215.

[4] LeCun, Y., & Bengio, Y. (1995). Convolutional networks for images, speech, and time series. The handbook of brain theory and neural networks, 3361(10), 1995.

[5] Harrison Jr, D., & Rubinfeld, D. L. (1978). Hedonic housing prices and the demand for clean air. Journal of environmental economics and management, 5(1), 81-102.

[6] Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural computation, 9(8), 1735-1780.

[7] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. arXiv preprint arXiv:1706.03762.

[8] Mordvintsev, A., Olah, C., & Tyka, M. (2015). Inceptionism: Going deeper into neural networks.

[9] Visualizing what convnets learn, URL: https://keras.io/examples/vision/visualizing_what_convnets_learn/, hozzáférés dátuma: 2021.03.26.

[10] Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., & Batra, D. (2017). Grad-cam: Visual explanations from deep networks via gradient-based localization. In Proceedings of the IEEE international conference on computer vision (pp. 618-626).

[11] Model Interpretation using CAM and GRAD-CAM, URL: https://ai-fast-track.github.io/timeseries/univariate_timeseries_CAM/, hozzáférés dátuma: 2021.03.26.

[12] Kadir, T., & Brady, M. (2001). Saliency, scale and image description. International Journal of Computer Vision, 45(2), 83-105.

[13] Simonyan, K., Vedaldi, A., & Zisserman, A. (2013). Deep inside convolutional networks: Visualising image classification models and saliency maps. arXiv preprint arXiv:1312.6034.

[14] Pan, Q., Hu, W., & Zhu, J. (2020). Series Saliency: Temporal Interpretation for Multivariate Time Series Forecasting. arXiv preprint arXiv:2012.09324.

[15] Keras Visualization Toolkit, URL: https://raghakot.github.io/keras-vis/, hozzáférés dátuma: 2021.03.26.

[16] Montavon, G., Binder, A., Lapuschkin, S., Samek, W., & Müller, K. R. (2019). Layer-wise relevance propagation: an overview. Explainable AI: interpreting, explaining and visualizing deep learning, 193-209.

[17] Siddiqui, S. A., Mercier, D., Munir, M., Dengel, A., & Ahmed, S. (2019). Tsviz: Demystification of deep learning models for time-series analysis. IEEE Access, 7, 67027-67040.

[18] Arras, L., Arjona-Medina, J., Widrich, M., Montavon, G., Gillhofer, M., Müller, K. R., ... & Samek, W. (2019). Explaining and interpreting LSTMs. In Explainable ai: Interpreting, explaining and visualizing deep learning (pp. 211-238). Springer, Cham.

[19] Layerwise-Relevance-Propagation, URL: https://github.com/atulshanbhag/Layerwise-Relevance-Propagation, hozzáférés dátuma: 2021.03.26.

[20] Zeiler, M. D., & Fergus, R. (2014, September). Visualizing and understanding convolutional networks. In European conference on computer vision (pp. 818-833). Springer, Cham.

[21] Van der Maaten, L., & Hinton, G. (2008). Visualizing data using t-SNE. Journal of machine learning research, 9(11).

[22] McInnes, L., Healy, J., & Melville, J. (2018). Umap: Uniform manifold approximation and projection for dimension reduction. arXiv preprint arXiv:1802.03426.

[23] Jolliffe, I. T. (2002). Principal Component Analysis. Springer Series in Statistics. New York: Springer-Verlag.

[24] TensorBoard, URL: https://www.tensorflow.org/tensorboard, hozzáférés dátuma: 2021.03.26.

[25] tensorboardX, URL: https://pypi.org/project/tensorboardX/, hozzáférés dátuma: 2021.03.26.

[26] sklearn.manifold.TSNE, URL: https://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html, hozzáférés dátuma: 2021.03.26.

[27] Uniform Manifold Approximation and Projection, URL: https://github.com/lmcinnes/umap, hozzáférés dátuma: 2021.03.26.