Mélytanulási módszerek alkalmazhatóságának vizsgálata nem-életbiztosítási területen

Diplomamunka

Írta: Kővári Péter Viktor

Biztosítási és pénzügyi matematika MSc

Aktuárius specializáció

Témavezető: Dr. Prokaj Vilmos

habil. egyetemi docens, tanszékvezető

Valószínűségelméleti és Statisztika Tanszék

Eötvös Loránd Tudományegyetem, Természettudományi Kar

Budapesti Corvinus Egyetem Eötvös Loránd Tudományegyetem

Budapest, 2025.

Köszönetnyilvánítás

Szeretném megköszönni témavezetőmnek, Dr. Prokaj Vilmosnak, a kutatómunkámhoz nyújtott iránymutatását, a hasznos tanácsait, illetve a segítőkészségét, amivel válaszolt a kérdéseimre. Továbbá szeretnék köszönetet mondani édesanyámnak, aki mindvégig támogatott a diplomamunka megírása alatt.

Tartalomjegyzék

1. Bevezetés

A nem-életbiztosításban fontos feladat a kárszám, illetve a kárnagyság modellezése a termékek árazásához és a kockázatkezeléshez. Ennek érdekében az aktuáriusok számos módszert dolgoztak ki, hogy minél pontosabb becsléseket kaphassanak. A diplomamunkám célja a nem-életbiztosításban használatos hagyományos, illetve mélytanulási módszereken alapuló modellek összehasonlítása és alkalmazhatóságuk vizsgálata.

A dolgozatban először ismertetem a károk modellezéséhez használt hagyományos módszereket mind a kárszám, mind a kárnagyság esetén. A kárszámoknál külön tárgyalom az egyéni, illetve az összetett kockázat modelljét. A matematikai modellek leírása mellett gyakorlati példákat is hozok az alkalmazásukra. A kárnagyságok modellezéséhez szintén ismertetem a matematikai módszereket, illetve gyakorlati példákat is mutatok a használatukra. Itt kitérek egy érdekes felhasználásra tűzkárok esetén, ennél a modellnél gráfelméleti megfontolásokat is alkalmazunk.

A dolgozat következő fejezetében rátérek a mélytanulási módszerek általános bemutatására. Ismertetem a neurális hálók általános felépítését és a tanulási folyamatot, továbbá bemutatom a mélytanulási rendszerek különböző típusait. Itt is hangsúlyt fektetek az alkalmazásokra, és számos példát hozok, hogy hogyan használhatók a mélytanulási rendszerek különféle területeken.

Végül az utolsó fejezetben kifejtem a mélytanulási módszerek számunkra legérdekesebb alkalmazását, az aktuáriusi területen történő használatukat. Bemutatom a GLM-et, amely hagyományosan számos modell kiindulópontja, majd ismertetem, hogy milyen mélytanulási módszereken alapuló modellek használhatók a nem-életbiztosítás területén. A modellek tárgyalása után bemutatom egy lehetséges implementációjukat gépjármű-felelősségbiztosításhoz tartozó kárgyakorisági adatokon. Végül megvizsgálom a mélytanulási módszereken alapuló modellekkel

kapott eredményeket, és összevetem őket a GLM alapú modellek eredményeivel.

2. Károk modellezése a nem-életbiztosításban

A nem-életbiztosítás területén a termékek árazásához meg kell becsülnünk valamilyen módszerrel a károk számát, illetve a károk nagyságát. A károk modellezéséhez alapvetően kétféle megközelítést szoktak alkalmazni. A hagyományos módszer szerint megpróbáljuk megérteni a modellezendő károk természetét, majd valamilyen észszerű feltételezések alapján illesztünk rájuk matematikai módszerekkel modelleket (eloszlásokat). Ezek után teszteljük a modell magyarázó erejét, azaz megvizsgáljuk, hogy összhangban vannak-e a modell által előrejelzett kárszámok és kárnagyságok a valós adatokkal. A másik megközelítés szerint kiindulunk egy tetszőleges modellből, amelyet nem szükséges elemi megfontolásokból levezetni, csupán azt vizsgáljuk meg, hogy megfelelően illeszkedik-e a megfigyelt adatokra. Ennél a megközelítésnél kiválóan alkalmazhatók gépi tanulási módszerek. Fontos, hogy megfelelően (optimálisan) komplex legyen a választott modell, tehát hogy jól illeszkedjen a tanuláshoz használt adathalmazra, illetve megfelelő előrejelzést adjon új (a tanuláshoz nem használt) adatokra. Minél összetettebb a modell, annál jobban illeszkedik a tanuláshoz használt adatokra, azaz kicsi a torzítása, azonban ezzel együtt megnő a modell becsült paramétereinek a varianciája, azaz ha a tanuláshoz új adathalmazt használunk, az így kapott paraméterek varianciája nő a komplexitással. Túl egyszerű modellnél a paraméterek varianciája kicsi, azonban ekkor a modell magyarázó ereje is csökken, azaz nagy lesz a torzítás. A modellező feladata, hogy olyan modellt válasszon, amelynél a torzításból és a varianciából adódó összhiba minimális. Ebben a fejezetben áttekintjük a károk modellezéséhez használt hagyományos módszereket. Az összefoglaló a [17] cikken alapszik.

2.1. Kárszámok modellezése

A kárszámokat modellezhetjük egyéni, illetve csoportos szinten is, így kapjuk

az egyéni, illetve az összetett kockázat modelljét. A kárszámok modellezésére alapvetően háromféle eloszlást használnak az aktuáriusok: binomiális, Poisson-, illetve negatív binomiális eloszlást. Azért szokták jellemzően ezt a három eloszláscsaládot választani, mivel az egyszerűségük miatt jól kezelhetők, illetve az illesztéshez használt adatpontok alacsony száma nem indokol összetettebb eloszlásokkal való modellezést. E mellett a három eloszláscsalád mellett szól az az érv is, hogy a variancia kárszámhoz viszonyított nagysága alapján kiválaszthatjuk, hogy melyik család lesz megfelelő a modellezéshez. A binomiális eloszlás várható értéke kisebb, mint a varianciája, a negatív binomiális eloszlás várható értéke nagyobb, mint a varianciája, a Poisson-eloszlásnál pedig a két érték megegyezik. A három nevezetes eloszlás néhány tulajdonsága az alábbi táblázatban látható:

Eloszlás	Paraméterek	$P(X = k),\ k \in \mathbb{N}$	$E(X)$	$D^2(X)$
Binomiális	$n \in \mathbb{N}, p \in [0,1]$	$\binom{n}{k} p^k (1-p)^{n-k} \mathbb{1}_{0 \le k \le n}$	np	$np(1-p)$
Poisson	$\lambda \in (0, \infty)$	$\frac{\lambda^k e^{-\lambda}}{k!}$	λ	λ
Negatív binomiális	$r \in (0, \infty), p \in (0,1]$	$\frac{\Gamma(r+k)}{\Gamma(r)k!}(1-p)^k p^r$	$\frac{r(1-p)}{p}$	$\frac{r(1-p)}{p^2}$

2.1.1. Az egyéni kockázat modelljei

Az egyéni kockázat modelljében véges sok kár következhet be, mindegyik legfeljebb egyszer. Az egyes károk ebben a megközelítésben Bernoulli-eloszlásúak. Amennyiben feltesszük, hogy a károk azonos valószínűséggel következnek be, továbbá egymástól függetlenül - ezt gyakran feltehetjük -, a kárszám független azonos paraméterű Bernoulli-eloszlások összege lesz, azaz binomiális eloszlású. Ha csak a károk bekövetkezésének függetlenségét tesszük fel (a p_i $(1 \le i \le n)$ bekövetkezési valószínűségek különbözhetnek), akkor a károk bekövetkezése együttesen többdimenziós Bernoulli-eloszlású. Ha n-féle kár következhet be, X_i $(1 \le i \le n)$ az i-edik kár indikátora (értéke kármentesség esetén 0, a kár bekövetkezése esetén 1), akkor

$X = (X_1, \ldots, X_n)$ adja meg, hogy mely károk következnek be, X többdimenziós Bernoulli-eloszlású. Ha N jelöli a kárszámot, akkor annak a valószínűsége, hogy pontosan k $(0 \leq k \leq n)$ darab kár következik be:

$$P(N = k) = \sum_{\substack{x_1,\ldots,x_n \in \{0,1\}, \\ \text{ahol } x_1+\ldots+x_n=k}} \prod_{i=1}^{n} p_i^{x_i}(1-p_i)^{1-x_i}.$$

Amennyiben a károk bekövetkezései nem függetlenek egymástól, alkalmazhatnánk más eloszlásokat, azonban legtöbbször célravezetőbb a közöttük lévő kapcsolatot modellezni. Ha nem ismerjük az eloszlások közötti kapcsolatot, akkor érdemes valamilyen szisztematius sokkot feltételezni, ami mindegyik kockázattípust érinti. Ez jellemzően átmenetileg van jelen. Ezzel a feltételezéssel élve a modellünk még mindig binomiális vagy többdimenziós Bernoulli, azonban a kárvalószínűségek maguk is valószínűségi változók realizációi.

2.1.2. Az összetett kockázat modelljei

Az összetett kockázat modelljében szintén véges sok kártípus lehetséges, azonban ugyanolyan fajta károk többször is bekövetkezhetnek. Ennél a modellnél jellemzően kicsi annak a hatása, hogy egy szerződés megszűnik, ha elég nagy a biztosító portfóliója. A modell felépítéséhez szükséges definiálni a számlálófolyamatokat.

2.1.2.1. Definíció. Az $\{N(t), t \geq 0\}$ sztochasztikus folyamatot *számlálófolyamatnak* nevezzük, ha teljesülnek az alábbi tulajdonságok:

(1) $N(t) \in \mathbb{Z}$ minden $0 \leq t$-re,

(2) $N(t) \geq 0$ minden $0 \leq t$-re,

(3) $N(s) \leq N(t)$ minden $0 \leq s < t$-re.

A számlálófolyamatokkal adott események bekövetkezésének számát adhatjuk meg adott időintervallumban, azaz $N(t)$ megmutatja, hogy a $[0,t]$ intervallumon hányszor következett be az adott esemény. Ha $0 \leq s \leq t$, akkor $N(t) - N(s)$ megadja, hogy az $(s,t]$ intervallumban hányszor következett be az adott esemény. $N(t) - N(s)$-t *növekménynek* nevezzük [12, 14, 19, 21].

2.1.2.2. Definíció. Az $\{N(t), t \geq 0\}$ számlálófolyamatot *független növekményűnek* nevezzük, ha tetszőleges véges sok páronként diszjunkt balról nyílt, jobbról zárt intervallumon vett növekmények függetlenek, azaz tetszőleges $0 \leq t_1 < \ldots < t_n$-re a $(t_1, t_2], (t_2, t_3], \ldots, (t_{n-1}, t_n]$ intervallumokon vett $N(t_2) - N(t_1), N(t_3) - N(t_2), \ldots,$ $N(t_n) - N(t_{n-1})$ növekmények függetlenek [14, 19, 21].

2.1.2.3. Definíció. Az $\{N(t), t \geq 0\}$ számlálófolyamatot *stacionárius növekményűnek* nevezzük, ha tetszőleges balról nyílt, jobbról zárt intervallumon vett növekmény eloszlása kizárólag az intervallumtól hosszától függ, azaz tetszőleges $0 \leq s < t$-re és $h > 0$-ra, $N(t) - N(s)$ és $N(t+h) - N(s+h)$ azonos eloszlású. Amennyiben feltesszük, hogy $N(0) = 0$, akkor ez azt jelenti, hogy $N(t) - N(s)$ és $N(t-s)$ azonos eloszlású [14, 19, 21].

Az összetett kockázat modellezéséhez az egyik legjobban használható számlálófolyamat a Poisson-folyamat [14, 22].

2.1.2.4. Definíció. Legyen $\lambda > 0$ rögzített. Az $\{N(t), t \geq 0\}$ számlálófolyamatot λ paraméterű *stacionárius növekményű Poisson-folyamatnak* nevezzük, ha teljesülnek az alábbi tulajdonságok:

(1) $N(0) = 0$,

(2) $N(t)$ független növekményű és stacionárius növekéményű,

(3) tetszőleges $r > 0$-ra $N(r)$ $Poisson(\lambda r)$-eloszlású

[14, 19, 21, 22].

A stacionárius növekményű Poisson-folyamat bár kézenfekvő az összetett kockázat modellezésére, a valóságban gyakran sérülnek a használatához szükséges feltételezések. Ezt mutatják az alábbi példák:

1. A káresemények nem egymástól függetlenül következnek be. Ez történhet valamilyen közvetlen kapcsolat miatt, vagy közös sokkhatás esetén.

    - Lakástűz esetén, ha a biztosító egymáshoz közel élők közösségét biztosítja. Ekkor egyik lakásról átterjedhet a tűz a többire.
    - Egészségügyi területen, amikor egy járvány megfertőzi egy közösség egy tagját. Ezáltal megnő annak a kockázata, hogy mások is megbetegednek.
    - Terrorizmus esetén, ha például egy nagyobb szociális vagy politikai rendezvényen sokan érintettek.
    - Kibertámadás esetén, ha meggyengül egy egész számítógépes rendszer védelme.

2. A folyamat paramétere időben változhat. Például a paraméter értékét befolyásolhatják jövőbeli események, döntések.

    - Megváltozhat egy káresemény jövőbeli elbírálása, ha egy bírósági döntés precedenst teremt (például új törvényt hoznak) a későbbi hasonló károk megítéléséhez.

3. Több esemény csoportosan következik be.

    - Természeti vagy ember által okozott katasztrófák, mint például a terrorizmus, többféle kár együttes bekövetkezéséhez vezethetnek. Például

egy jégeső több ingatlankárt és mezőgazdasági káreseményt eredményezhet.

- A katasztrófák gyakoriságát befolyásoló külső tényezők hatással lehetnek arra, hogy azonos veszélyforrás miatt többféle esemény következik be egyidejűleg, mivel adottak hozzá a feltételek. Például meleg tengerek és gyakori hurrikánok, vagy tektonikai aktivitás és gyakori földrengések.

Láttuk, hogy nem mindig helytálló konstansnak feltételezni a Poisson-folyamat paraméterét, továbbá az aktuáriusi gyakorlat azt mutatja, hogy a Poisson-eloszlás alulbecsülheti a valódi varianciát, így gyakran használják a negatív binomiális eloszlást, amelynek a varianciája nagyobb, mint a várható értéke. Gyakran még abban az esetben is szükséges, hogy a variancia nagyobb legyen a várható értéknél, amikor a károk bekövetkezése egy adott időintervallumban Poisson-eloszlást követ. A probléma kezelhető a Poisson-folyamat általánosításával, azaz ha megengedjük, hogy a paramétere időben változhasson.

2.1.2.5. Definíció. Az $\{N(t), t \geq 0\}$ számlálófolyamatot $\lambda(t)$ intenzitású *Poisson-folyamatnak* nevezzük, ha teljesülnek az alábbi tulajdonságok:

(1) $N(0) = 0$,

(2) $N(t)$ független növekményű,

(3) $\lambda(t) \geq 0$ mérhető függvény, $\Lambda(t) = \int_0^t \lambda(u)\,du$ minden $t > 0$-ra véges,

(4) tetszőleges $0 \leq s < t$-re, $N(t) - N(s)$ $Poisson(R(s,t))$-eloszlású, ahol $R(s,t) = \int_s^t \lambda(u)\,du$

[14, 22].

Általában a $\lambda(t)$ intenzitás maga is egy sztochasztikus folyamat, ekkor $\Lambda(t) = \int_0^t \lambda(u)\,du$ is sztochasztikus folyamat, így jutunk el a Cox-folyamat fogalmához:

2.1.2.6. Definíció. Az $\{N(t), t \geq 0\}$ számlálófolyamatot $\lambda(t)$ intenzitású *Cox-folyamatnak* nevezzük, ha teljesülnek az alábbi tulajdonságok:

(1) $N(0) = 0$,

(2) $N(t)$ független növekményű,

(3) $\lambda(t)$ előrejelezhető sztochasztikus folyamat, $\lambda(t) \geq 0$ mérhető függvény, $\Lambda(t) = \int_0^t \lambda(u)\,du$ minden $t > 0$-ra véges,

(4) N λ-ra vett feltételes eloszlása $\lambda(t)$ intenzitású Poisson-folyamat

[4, 7, 11, 14, 13].

Cox-folyamatokra szokásos példa, amikor valamilyen külső tényező befolyásolja az intezitás időbeli változását, így maga az intenzitás is sztochasztikus folyamat. Ezzel modellezhető például, hogy mikor érkeznek a vásárlók egy boltba. Itt az intenzitás véletlenszerűen változik (akcióktól, leárazásoktól, árstoptól függően). Hasonlóan modellezhetünk földrengéseket is Cox-folyamatokkal. Itt az intenzitás függ például a szeizmikus aktivitástól, ami maga is egy sztochasztikus folyamat.

A Cox-folyamatok megoldják az alacsony variancia problémáját. Habár a mögöttes folyamat Poisson, illetve a kárszám Λ-ra vett feltételes eloszlása Poisson, a folyamat varianciája mégis nagyobb vagy egyenlő, mint a várható értéke. Ez az alábbi módon látható:

$$D^2(N) = E(D^2(N|\Lambda)) + D^2(E(N|\Lambda)) = E(E(N|\Lambda)) + D^2(E(N|\Lambda)) =$$

$$= E(N) + D^2(E(N|\Lambda)) \geq E(N).$$

Egyenlőség pontosan akkor áll fenn, ha $E(N|\Lambda)$ majdnem mindenütt konstans (rögzített t-re).

2.2. Kárnagyságok modellezése

A kárnagyságok modellezéséhez hagyományosan kézenfekvőnek tűnik a lognormális eloszlás használata. Lognormális eloszlású változó kizárólag pozitív értékeket vehet fel, továbbá többféle nagyságrendű kár is modellezhető a segítségével, így megfelelő választásnak bizonyulhat, azonban nagyobb adathalmazok vizsgálatával kiderülhet, hogy az eloszlás sem az átlagos, sem az extrém károkat nem modellezi kielégítő pontossággal.

Felelősségbiztosítások esetén a károk a legtöbb esetben jól modellezhetők általánosított Pareto-eloszlással (GPD). Az általánosított Pareto-eloszlás eloszlásfüggvénye:

$$F(x) = \begin{cases} 1 - \left(1 + \xi \frac{x-\mu}{\sigma}\right)^{-\frac{1}{\xi}}, & \text{ha } \xi \neq 0 \\ 1 - \exp\left(-\frac{x-\mu}{\sigma}\right), & \text{ha } \xi = 0, \end{cases}$$

ahol $\sigma > 0$, továbbá a függvény tartója $\xi \geq 0$ esetén $\{x : x \geq \mu\}$, míg $\xi < 0$ esetén $\{x : \mu \leq x \leq \mu - \frac{\sigma}{\xi}\}$. A $\xi = 0$ eset az eltolt exponenciális eloszláshoz tartozik, ez a másik esetből ξ-vel 0-hoz tartva adódik. A $\xi < 0$ eset az eltolt és átskálázott béta-eloszláshoz tartozik. A $\xi > 0$ eset az eltolt és átskálázott Pareto-eloszláshoz tartozik. A legtöbb esetben az eltolt és átskálázott Pareto-eloszlás megfelelő a modellalkotáshoz. Az általánosított Pareto-eloszlás jobban modellezi a nagy károkat, mint a lognormális, viszont a kis károk leírására a lognormális eloszlás jobban működik. Előfordul, hogy a két eloszlást "keverik": a kis károkat lognormális eloszlással, míg a nagyokat általánosított Pareto-eloszlással modellezik. A tapasztalat azt mutatja, hogy általában a lognormális eloszlás sem írja le jól a kisebb károkat, így célszerű más eloszlást választani. A Lomax-eloszlás megfelelő választásnak bizonyulhat kis károk modellezésére, míg az extrém károkat továbbra is általánosított Pareto-eloszlással írhatjuk le. A Lomax-eloszlás eloszlásfüggvénye:

$$F(x) = 1 - \left(1 + \frac{x}{\lambda}\right)^{-\alpha}, \text{ ha } x \geq 0,$$

ahol $\alpha > 0$, $\lambda > 0$. Vegyük észre, hogy a Lomax-eloszlás az általánosított Pareto-eloszlás speciális esete $\mu = 0$, $\sigma = \frac{\lambda}{\alpha}$, $\xi = \frac{1}{\alpha}$ paraméterekkel. Az eltolt Lomax-eloszlást (ha $\mu \neq 0$) II-es típusú Pareto-eloszlásnak hívják. Ezt egy másik általánosított Pareto-eloszlással kombinálva megfelelő modellhez juthatunk.

A nagy károk viselkedését jól megragadja az ún. duplázási arány, a segítségével leírható az eloszlás "farkának" a lecsengése:

$$DoublingRatio(x) = \frac{\bar{F}(2x)}{\bar{F}(x)},$$

ahol

$$\bar{F}(x) = 1 - F(x)$$

a túlélési függvény.

Exponenciális eloszlásra:

$$\bar{F}(x) = e^{-\lambda x} \quad (x \geq 0, \quad \lambda > 0),$$

így

$$DoublingRatio(x) = \frac{e^{-2\lambda x}}{e^{-\lambda x}} = e^{-\lambda x}.$$

Exponenciális eloszlás esetén a duplázási arány gyorsan 0-hoz tart. Ez is azt támasztja alá, hogy az exponenciális eloszlás nem megfelelően modellezi a nagy károkat.

Vizsgáljuk meg a Pareto-eloszlás duplázási arányát:

$$\bar{F}(x) = \left(\frac{\lambda}{x}\right)^{\alpha} \quad (x \geq \lambda, \quad \alpha > 0, \quad \lambda > 0),$$

így

$$DoublingRatio(x) = \frac{\left(\frac{\lambda}{2x}\right)^{\alpha}}{\left(\frac{\lambda}{x}\right)^{\alpha}} = 2^{-\alpha}.$$

Látható, hogy Pareto-eloszlás esetén a duplázási arány konstans, a túlélési függvény lecsengése hatványrendű. Ez azt mutatja, hogy a Pareto-eloszlás már alkalmasabb nagy károk modellezésére, mivel nagyobb valószínűséggel fognak nagy

károk előfordulni, mint exponenciális eloszlás esetén, azonban nagyon nagy károk még mindig kellően kis valószínűséggel fognak bekövetkezni.

Tekintsük az általánosított Pareto-eloszlást $\xi > 0$ esetén:

$$\bar{F}(x) = \left(1 + \xi \frac{x-\mu}{\sigma}\right)^{-\frac{1}{\xi}} \quad (x \geq \mu, \quad \sigma > 0, \quad \xi > 0),$$

így

$$DoublingRatio(x) = \frac{\left(1 + \xi \frac{2x-\mu}{\sigma}\right)^{-\frac{1}{\xi}}}{\left(1 + \xi \frac{x-\mu}{\sigma}\right)^{-\frac{1}{\xi}}} \to 2^{-\frac{1}{\xi}} \quad (x \to \infty).$$

Látható, hogy az általánosított Pareto-eloszlást $\xi > 0$ esetén aszimptotikusan ugyanúgy viselkedik, mint a Pareto-eloszlás, így ez is hasonlóan alkalmas nagy károk modellezésére.

A károk Pareto-típusú viselkedését a különböző kártípusok vizsgálatával lehet megérteni. Szorítkozzunk a testi sérüléses kártérítési igényekre. Ezek az alábbi komponensekből tevődnek össze:

- az orvosi ellátás és a gondozás költségei
- az elmaradt kereset miatti kompenzáció
- fizikai fájdalom és érzelmi megterhelés miatti kompenzáció
- életminőség-csökkenés, illetve maradandó rokkantság miatti kártérítés
- egyéb költségek (orvosi kezelésekhez kapcsolódó szállítási költségek, a lakás/jármű fogyatékosság miatt történő átalakításának költségei)

Amennyiben az ezekhez tartozó eloszlások közül legalább egynek a túlélési függvénye hatványrendben csökken (és nincs olyan, amelyiknek lassabban), akkor az illesztett eloszlásnak aszimptotikusan hatványrendben kell lecsengeni. A keresetkiesés mértéke egyenesen arányos a bevétellel, és Champernowne szerint az eloszlás

"farka" Pareto-típusú [6], így a modellhez választott eloszlásnak is hatványrendben kell lecsengenie. Ha több komponenshez tartozik olyan eloszlás, amelynek a "farka" hatványrendben csökken, akkor az fog dominálni, amelyik a leglassabban cseng le.

Egy másik példa a kárnagyságok viselkedésének illusztrálására a bírósági esetek költségeinek változása. A korábbi hasonló eseteknél kifizetett kártérítések összege viszonyítási alapként szolgál az újabb eseteknél megállapított kompenzáció mértékéhez. Újabb károk esetén a kifizetés mértékét a kártérítés referenciaértékéhez viszonyítják, annak valahányszorosa lesz az aktuális kompenzáció. Ez az eljárás exponenciális növekedést jelentene, azonban a gyakorlat azt mutatja, hogy idővel beáll egy ún. egyensúlyi állapot, és a növekedés üteme konstans lesz. Ez arra utal, hogy valamilyen mechanizmusok limitálják a növekedést. Egy ilyen limitáló faktor lehet a minimális (pozitív) kárnagyság (kifizetés) létezése. Ez lehet önrész bevezetése miatt, vagy szimplán azért, mert bizonyos összeghatár alatt nem foglalkoznak a károk bejelentésével. Egy másik tényező, amely csökkenti a növekedési ütemet, az, hogy bizonyos típusú károk idővel eltűnnek. Például bizonyos típusú károkért már nem jár kártérítés törvénymódosítások miatt, vagy a kár természetéből kifolyólag.

Ingatlankárok modellezésekor nem megfelelő módszer korábbi adatokra eloszlást illeszteni, mivel a jövőbeli kárnagyságok erősen függenek attól, hogy mekkora a lehetséges maximális kárnagyság (MPL), ez pedig a portfólió egyedi jellemzője, és értéke eltérhet a korábbi évek MPL-jeitől. Célszerű a kárnagyságot az MPL százalékában kifejezni, és ezekre görbét (eloszlásfüggvény, kitettségi függvény) illeszteni, így a görbék alakja nem függ az MPL-től. A Bernegger által javasolt görbék széles körben elterjedtek [3]. A Bernegger által aktuáriusi modellezésre bevezetett kétparaméteres függvények eredetileg a statisztikus mechanikában használatosak. A statisztikus mechanikában használt eloszlások sűrűségfüggvényei az alábbi alakú-

ak:

$$f(x) = \begin{cases} \frac{1}{Z} e^{-\frac{x}{kT}}, & \text{ha } x \geq 0, \quad \text{Maxwell–Boltzmann (MB)} \\ \frac{a}{ce^{\frac{x}{kT}} - 1}, & \text{ha } x \geq 0, \quad \text{Bose–Einstein (BE)} \\ \frac{a}{ce^{\frac{x}{kT}} + 1}, & \text{ha } x \geq 0, \quad \text{Fermi–Dirac (FD)} \end{cases}$$

Általánosabb alakban (ahogy Bernegger vizsgálta őket) a túlélési függvények az alábbi módon adhatók meg:

$$\bar{F}(x) = \begin{cases} \frac{1-b}{(g-1)b^{1-x} + (1-bg)}, & \text{ha } x < 1, \quad b > 0, \quad b \neq 1, \quad bg \neq 1, \quad g > 1 \\ b^x, & \text{ha } x < 1, \quad bg = 1, \quad g > 1 \\ \frac{1}{1+(g-1)x}, & \text{ha } x < 1, \quad b = 1, \quad g > 1 \\ 1, & \text{ha } x < 1, \quad g = 1 \text{ vagy } b = 0 \\ 0, & \text{ha } x \geq 1 \end{cases}$$

A $bg = 1$ eset a MB-eloszláshoz, a $bg > 1$ eset a BE-eloszláshoz, a $bg < 1$ eset a FD-eloszláshoz tartozik. A $b = 1$ esethez nem rendelhető fizikai tartalom. $\bar{F}(x) = 0$, ha $x \geq 1$ egy fizikai tartalommal szintén nem rendelkező módosítás, hogy a teljes kár ($X = 1$) valószínűsége pozitív legyen: $P(X = 1) = \frac{1}{g}$, $g \geq 1$. A továbbiakban a fentebb ismertetett eloszlást MBBEFD(b, g) eloszlásnak fogjuk nevezni. Bernegger nem állapított meg az aktuáriusi és a statisztikus mechanikai alkalmazás között mélyebb kapcsolatot, azonban érdekes módon, amennyiben a modellt tűzesetek által okozott károkra alkalmazhatjuk, megállapítható valamiféle kapcsolat. Ha különböző mechanizmusokat feltételezünk a tűz terjedéséről, visszakapjuk a MB, a BE és a FD esetet.

A tűzkárok szimulációjára ötletes megoldást kínál a gráfelméleti megközelítés [18]. A vizsgált ingatlant egy gráffal reprezentáljuk: a csúcsok helyiségeket (vagy valamilyen egységeket) szimbolizálnak. Két helyiség akkor van éllel összekötve, ha az egyik helyiségről a másikra (illetve a másikról az egyikre) közvetlenül átterjedhet a tűz. A legegyszerűbb modellben kezdetben minden csúcshoz az 1 értéket

rendeljük, majd ha egy helyiséget elér a tűz, a neki megfelelő csúcs értékét 0-ra változtatjuk. Minden helyiségre igaz, hogy vagy teljes benne a kár (1), vagy nem keletkezik benne kár (0). A tűz egy véletlenszerűen választott csúcsból indul el, ennek a csúcsnak az értékét 0-ra változtatjuk, továbbá minden olyan csúcs értékét is, amely elérhető ebből a csúcsból, mivel az általuk szimbolizált helyiségekre tud átterjedni a tűz. A kárnagyságot a 0 értékű csúcsok számával azonosítjuk. Ebben a modellben a teljes biztosított érték (TIV) a gráf csúcsainak száma, míg a lehetséges maximális kárnagyság (MPL) a gráf legnagyobb összefüggőségi komponensének mérete, mivel ha a tűz egy helyiségből elindul, csak az onnan elérhető (azaz az adott helyiséghez tartozó csúcs összefüggőségi komponenséhez tartozó helyiségeket érheti el). A modell finomítható azzal, hogy a csúcsokhoz más-más értékeket rendelhetünk, illetve ha részleges károk bekövetkezését is megengedjük.

Az eredeti modell továbbfejlesztésével kétféle újabb modellezési lehetőséget mutatunk be, a statikus és a dinamikus megközelítést. A statikus megközelítés szerint egy $G = (V, E, W)$ súlyozott gráfból indulunk ki, ahol V a csúcsok, E az élek, W pedig az élekhez tartozó súlyok halmaza. Minden $e \in E$ élhez tartozik egy $w_e \in W$ súly, ahol $w_e \in [0, 1]$. A tűzkárt úgy szimuláljuk, hogy véletlenszerűen választunk egy $v^* \in V$ csúcsot, ahonnan elindul a tűz. A következő lépésben minden $e \in E$ élt $1 - w_e$ valószínűséggel kitörlünk a gráfból, így egy G' gráfot kapunk. A kárnagyság a v^* G'-beli összefüggőségi komponensének a mérete. Ha a kárnagyságot elosztjuk az MPL értékével (G legnagyobb összefüggőségi komponensének mérete), akkor megkapjuk a kárarányt. A szimulációt sokszor elvégezve (közelítőleg) megkapjuk a kárarányok eloszlását. A szimulációt egyetlen ingatlanra is elvégezhetjük, illetve akár egy egész ingatlanportfólióra is. Ekkor viszont az ingatlanok közül is véletlenszerűen kell választani (néhányat), hogy melyiket/melyeket sújt tűzkár.

A dinamikus megközelítés szerint a tűz determinisztikus úton terjed, amelyet meghatároz a gráf összefüggősége, azonban a terjedési idők véletlenszerűek. Az

élekhez tartozó súlyok határozzák meg, hogy mennyi idő alatt terjed át a tűz egyik helyiségről a másikra, illetve például azt, hogy nyitva vagy zárva van-e az ajtó a két helyiség határán. Azt feltételezzük, hogy a tűz T ideig terjed, majd eloltják (elalszik). T exponenciális eloszlású:

$$F_T(t) = 1 - e^{-\lambda t}.$$

A statikus szimulációhoz hasonlóan most is kiválasztunk véletlenszerűen egy v^* csúcsot, ahonnan elindul a tűz, továbbá választunk egy $T = t^*$ időpontot is az exponenciális eloszlásból, hogy mennyi ideig terjed a tűz. Az, hogy mennyi időbe telik végighaladni egy élen, a szomszédos helyiségek határának típusától függ. Az élekhez megadunk n_{nyitva}, $n_{\text{zárva}}$, n_{fal} mennyiségeket, amelyek azt mutatják meg, hogy mennyi idő alatt terjed át a tűz az egyik helyiségről a másikra, ha köztük nyitva/zárva van az ajtó, illetve ha közös fal választja el őket egymástól. A szimulációhoz annak a valószínűségét is szükséges megadni, hogy nyitva van az ajtó két helyiség között. A statikus esethez hasonlóan ennél is elvégezhetjük a szimulációt csupán egyetlen ingatlanra, illetve akár egy egész ingatlanportfólióra is. Ekkor viszont itt is véletlenszerűen kell választani (néhányat) az ingatlanok közül, hogy melyiket/melyeket sújt tűzkár.

A továbbiakban bemutatjuk, hogy hogyan kombinálható a dinamikus megközelítés Bernegger elméletével. A feltételek változatlanok: a tűz elindul valamelyik csúcsból, terjed, majd elalszik. Amennyiben bizonyos időn belül nem fékezik meg a tüzet, teljes kár következik be. A tűz determinisztikus úton terjed, azonban véletlenszerű, hogy honnan indul el, illetve hogy mennyi időbe telik a tűzoltóknak/katasztrófavédelemnek eloltani a tüzet (megfékezni a terjedését). A modellben most egyetlen izolált ingatlanra szorítkozunk. Tegyük fel, hogy adott egy $\varphi(t)$ determinisztikus függvény, amely leírja, hogy adott terjedési idő esetén a kár

mekkora része következik be. A terjedési idő továbbra is exponenciális eloszlású:

$$F_T(t) = 1 - e^{-\lambda t}, \quad X = \varphi(T).$$

X azt mutatja meg, hogy a kár mekkora része következik be, ha T a terjedési idő. $\varphi(t)$-t nevezzük fejlődési függvénynek, legyen monoton növő, illetve legyen

$$\varphi(0) = 0; \text{ továbbá } \varphi(t) = 1, \text{ ha } t > t_{\max}; \text{ illetve } \varphi(t) < 1, \text{ ha } t < t_{\max}$$

valamely rögzített $t_{\max}$ maximális időre. Legyen a teljes kár bekövetkezésének a valószínűsége $P(X = 1) = \frac{1}{g}$, $g \geq 1$. Ekkor

$$P(T > t_{\max}) = e^{-\lambda t_{\max}} = e^{-\frac{t_{\max}}{E(T)}} = \frac{1}{g},$$

így

$$g = e^{\lambda t_{\max}} = e^{\frac{t_{\max}}{E(T)}}.$$

Ezzel megkaptuk a Bernegger-modell g paraméterét $t_{\max}$ és $E(T)$ függvényeként, így elegendő ismerni, hogy mennyi idő alatt semmisülne meg a tűztől a teljes épület, illetve, hogy várhatóan mennyi időbe telik a tűzoltóknak/katasztrófavédelemnek kiérni a helyszínre és eloltani a tüzet (megfékezni a terjedését). A modellt ez az értelmezés intuitívvá és természetessé teszi. Vezessük le az $x = \varphi(t)$ fejlődési függvényt a $T = t$ realizáció mellett az MBBEFD eloszlás túlélési függvényéből:

$$\bar{F}(x) = P(\varphi(T) > x) = P(T > \varphi^{-1}(x)) = e^{-\lambda \varphi^{-1}(x)},$$

azaz

$$t = \varphi^{-1}(x) = -\frac{\ln(\bar{F}(x))}{\lambda}.$$

$\bar{F}(x)$ definíciója alapján, bevezetve az $\alpha = bg - 1, \beta = 1 - b$ változókat kifejezhetjük x-et:

$$x = \varphi(t) = -\frac{\ln\left(\frac{\beta e^{\lambda t} + \alpha}{\beta + \alpha}\right)}{\ln(1 - \beta)}.$$

A $b = 1$ esethez a $\beta \to 0$ esetén adódó határérték tartozik. $1-\beta = b \geq 0$, $\beta e^{\lambda t}+\alpha \geq \beta + \alpha = b(g-1) \geq 0$. Ezzel kaptunk egy közös kifejezést, amely mindegyik esetet magában foglalja. Számoljuk ki $\varphi(t_{\max})$-ot $e^{\lambda t_{\max}} = g$ felhasználásával:

$$\varphi(t_{\max}) = -\frac{\ln\left(\frac{\beta e^{\lambda t_{\max}}+\alpha}{\beta+\alpha}\right)}{\ln(1-\beta)} = -\frac{\ln\left(\frac{\beta g+\alpha}{\beta+\alpha}\right)}{\ln(1-\beta)} = -\frac{\ln\left(\frac{1}{b}\right)}{\ln(b)} = 1.$$

Megvizsgálva a $0 < x < 1$ esetet, felírhatjuk a korábban megszabott határfeltételek alkalmazásával a fejlődési függvényt:

$$\varphi(t) = \min\left(1, -\frac{\ln\left(\frac{\beta e^{\lambda t}+\alpha}{\beta+\alpha}\right)}{\ln(1-\beta)}\right).$$

Ekkor az $X = \varphi(T)$ valószínűségi változó MBBEFD(b, g) eloszlású. Ez formálisan megegyezik a statisztikus mechanikában alkalmazott eloszlással, az átparaméterezett kifejezések használatát, illetve a megváltozott értelmezést csupán a modellünk természete indokolja.

Megmutatható, hogy α előjele alapján eldönthető, hogy a három lehetséges eset közül melyik áll fenn. $\alpha < 0$ a FD esetet adja, $\alpha = 0$ a MB esetet adja, míg $\alpha > 0$ a BE esetet adja. A három tartományban máshogy viselkedik a fejlődési függvény: a FD tartományban lassuló ütemben nő, a MB tartományban lineáris, míg a BE tartományban gyorsuló ütemben nő.

3. Mélytanulási módszerek

A mélytanulási módszerek neurális hálók használatával lehetővé teszik, hogy az adatokat több egymásra épülő réteg segítségével különböző absztrakciós szinteken ábrázoljuk. A módszer segítségével akár nagy adathalmazok (gyakran nem triviális) finomszerkezetéről is nyerhetünk információt. A mélytanulás előnye egyéb módszerekkel szemben az, hogy a különböző absztrakciós rétegek nem emberi elemzés eredményeként jönnek létre, hanem az adatból kinyerhető információkból egy általános célú tanulási folyamat segítségével. A mélytanulási módszerek jól teljesítenek nagy dimenziós adatok belső szerkezetének feltárásában, így használatukkal számos, hosszú ideig megoldatlan probléma, feladat végezhető el. Alkalmazhatóságuk kiterjed természettudományos és üzleti területekre is. A módszer felhasználható kép-, hang-, videó-, illetve beszédfeldolgozási eljárásokhoz, hatásos gyógyszermolekulák tervezéséhez, részecskegyorsítók adatainak elemzéséhez, agyi hálózatok rekonstruálásához, betegséget/rendellenességet okozó mutációk meghatározásához, természetes nyelvi feldolgozáshoz (NLP), illetve fordításhoz. Számunkra a legérdekesebb az aktuáriusi, pontosabban nem-életbiztosítási területen való alkalmazhatósága. Mélytanulási módszerek használatával jobb eredmények érhetők el, mint a kanonikusnak tekintett modellekkel, ugyanakkor az új módszer megtartja a mögöttes aktuáriusi modell szerkezetét, így megőrzi annak az előnyeit is. Ebben a fejezetben bemutatom a mélytanulási módszerek jellemző típusait, illetve azok általános sémáját. Az összefoglaló a [15, 23, 10] cikkeken alapszik.

3.1. Felügyelt tanulás és optimalizáció

A gépi tanulási módszerek egyik legelterjedtebb fajtája - amely mélytanulási módszereknél is használatos - a felügyelt tanulás (supervised learning). Tekintsük azt a problémát, hogy adott egy képekből álló nagy adathalmaz, mindegyik kép

valamilyen objektumot ábrázol (pl.: házat, autót, embert, állatot), ez az objektum lesz a kép címkéje, és egy olyan rendszert szeretnénk készíteni, ami képes a képeket az alapján csoportosítani, hogy milyen objektumot ábrázolnak.

A tanulási folyamat során a gép kap inputként egy képet és outputként egy pontokból (score) álló vektort generál. A vektor elemei azt mutatják meg, hogy a gép szerint az adott kép mennyire tartozik az adott kategóriába. Például a pontok legyenek a $[0, 1]$ intervallum elemei, a 0 pont azt jelenti, hogy az adott kép a gép szerint egyáltalán nem tartozik az adott kategóriába, míg az 1 pont, hogy igen. Az a célunk, hogy a vektor (lehetőleg egyértelmű) maximális eleme megadja azt a kategóriát, amelyikbe a kép tartozik. A képhez tartozó elméleti (pontos) vektor az lenne, amelynek az adott kép címkéjéhez tartozó eleme 1, a többi pedig 0. A gép által outputként adott vektor és az elméleti vektor távolságát (a hibát) egy célfüggvénnyel (hibafüggvénnyel) mérjük. A gép a célfüggvény értéke alapján módosítja a belső paramétereit a hiba csökkentése érdekében. A módosítható paraméterek (súlyok) valós számok, amelyek megadják, hogy adott képre milyen vektort adjon outputként a gép. Egy gyakorlatban használt mélytanulási rendszer több százmillió súllyal rendelkezhet, illetve a tanuláshoz használt adathalmaz is akár több százmillió képből is állhat.

A célfüggvény értékének csökkentéséhez (vagyis a hiba minimalizálásához) a gradiens ereszkedés (gradient descent) módszerét használhatjuk. Ehhez tekintsük a célfüggvény súlyok szerinti gradiensét, majd vonjuk le mindegyik súlyból a súlyhoz tartozó derivált lépésközzel (ez egy kis pozitív szám) szorzott értékét. Ezzel az eljárással eljuthatunk egy (esetleg lokális) minimumba, vagy közel juthatunk hozzá.

A gyakorlatban leggyakrabban egy ennél kifinomultabb módszert használnak, a sztochasztikus gradiens ereszkedés (stochastic gradient descent) módszerét [20, 1]. Ennél a módszernél az adathalmazból vesznek egy jellemzően kisebb elemszámú mintát, majd ezekre az inputokra kiszámítják az outputokat, illetve a hibákat.

Az ezekhez tartozó gradiensek átlagát tekintik, és azt felhasználva módosítják a súlyok értékeit. Ezek után újabb mintát vesznek az adathalmazból, és megismétlik újra a folyamatot. Az eljárást addig folytatják, amíg a célfüggvény értéke kellően meg nem közelíti az elérni kívánt (lokális) minimumot. Az eljárás azért sztochasztikus, mert a kis mintákra kiszámított átlagos gradiens értéke valójában az egész adathalmazhoz tartozó átlagos gradiens egy zajos becslése. Ezzel az eljárással gyorsan viszonylag pontos súlyokat kaphatunk (kifinomultabb technikákhoz képest is). A tanulási fázis után a gép teljesítményét egy (korábban nem használt) teszthalmazon mérik.

A gépi tanulás területén gyakran alkalmazott eljárás, hogy lineáris klasszifikáció alapján osztják két csoport közül valamelyikbe a vizsgált objektumot (two-class linear classifier). A bemeneti tulajdonságokat tartalmazó vektornak veszik a súlyvektorral vett skaláris szorzatát, majd amennyiben ez az érték meghalad egy küszöbértéket, az egyik osztályba, különben pedig a másik osztályba sorolják az adott objektumot [24]. Ezzel az eljárással azonban gyakran különböző típusú objektumokat is egy osztályba sorolunk, illetve azonos típusú objektumokat különböző osztályokba. Az eljárás javítható kernelek alkalmazásával, amelyek segítségével nemlineáris kapcsolatok is vizsgálhatók, illetve egyéb, úgynevezett feature extractorokkal, amelyek kinyerik a vizsgált objektum bizonyos fontos tulajdonságait. A mélytanulási módszerek nagy előnye, hogy az általános célú tanulási eljárással automatikusan kinyerik az adathalmaz lényeges, jó tulajdonságait.

Egy mélytanulási rendszer több, jellemzően 5-20 rétegből áll, (majdnem) mindegyik réteg tanulással fejlődik, és az inputból valamilyen, általában nemlineáris leképezés segítségével készít outputot. Az output réteget leszámítva mindegyik rétegnek a kimeneti értékei adják a következő réteg bemeneti értékeit. A nemlineáris rétegek lehetővé teszik az adathalmaz finomszerkezetének feltárását, így a rendszer érzékeny lesz az apró részletekre, például meg tud különböztetni egy

szamojéd kutyát egy fehér farkastól, azonban érzéketlen lesz az objektum típusát nem módosító változásokra, mint például a háttér, a pozíció, illetve a fényerő megváltozására.

A neurális hálók egymáshoz kapcsolódó rétegekből állnak, az input és az output rétegen kívüli rétegeket rejtett rétegeknek nevezik. Tekintsünk egy neurális hálót. Ahhoz, hogy megtaláljuk a hibafüggvény minimumát valamilyen, gradienst használó módszerrel, ki kell számolni a hibafüggvény súlyok szerinti deriváltjait. Az output rétegnél kiszámítjuk a hibafüggvény outputok szerinti deriváltjait. Ezek után a hibafüggvény outputok szerinti deriváltjaiból a láncszabály alkalmazásával megkapjuk a hibafüggvény inputok szerinti deriváltjait (a hibafüggvény adott output szerinti deriváltját az adott output adott input szerinti deriváltjával szorozva). Végül kiszámíthatjuk a súlyokhoz tartozó deriváltakat is a láncszabály segítségével (az adott inputhoz tartozó deriváltat megszorozzuk az adott input megfelelő súly szerinti deriváltjával). Ezek után a következő (rejtett) rétegnél is (sorban visszafele haladva) kiszámítjuk a hibafüggvény outputok szerinti deriváltjait, ezek az eggyel feljebbi réteghez tartozó inputok szerinti deriváltak súlyozott összegei. Innen már az eljárást a fentebb ismertett módon folytathatjuk a rétegeken visszafele haladva egészen az input rétegig. Ezt a módszert a hálóban visszafele történő haladás miatt backpropagationnek nevezik.

A mélytanulás alkalmazásakor gyakran feedforward neurális hálókat (feedforward neural network, FFN) használnak. Egy ilyen rendszerben a rétegek fix méretű bemeneteket alakítanak fix méretű kimenetekké. Mindegyik rétegnél kiszámolják a bemenetek súlyozott összegét, majd ezt az értéket átadják egy nemlineáris függvénynek. A nemlineáris függvények általában az alábbiak lehetnek: a ReLU (rectified linear unit), ez tulajdonképpen a pozitívrész-függvény, a tangens hiperbolikusz, illetve a logisztikus függvény. Sok rétegből álló neurális hálóknál jellemzően a ReLU függvény alkalmazásával tanulnak a leggyorsabban a rétegek.

Korábban azt gondolták, hogy a gradiens ereszkedés módszerével a folyamat "rossz" lokális minimumokba juthat, ahol a súlyok kismértékű változtatásával már nem csökkenthető az átlagos hiba, azonban későbbi elméleti és empirikus kutatások azt mutatják, hogy általában hasonló "minőségű" lokális minimumokba jutunk a kiindulási pont választásától függetlenül. További elemzések megállapították, hogy gyakran sok olyan nyeregpont található az optimalizációhoz tartozó felületen, amelyekben a Hesse-mátrix legtöbb sajátértéke pozitív, és csupán néhány negatív. Ezeken a helyeken megáll az algoritmus, azonban a nyeregpontokra is igaz, hogy általában hasonló hibaérték tartozik hozzájuk, így nem lényeges, hogy melyiket találjuk meg.

Előretörést hozott a feedforward mélytanulási rendszerek (deep feedforward network) fejlődésében a felügyelet nélküli tanulás (unsupervised learning) módszere, amelynek nagy előnye a felügyelt tanuláshoz képest, hogy nincs szüksége a tanuláshoz felcímkézett adatokra. A felügyelet nélküli tanulás során az input egy nagy adathalmaz, amely megfigyelésekből és azok tulajdonságaiból áll, de nem tartalmazza a kimenetként várt értékeket. A tanulási folyamat során a rendszer az adathalmaz belső szerkezetét tárja fel külső információ felhasználása nélkül [8]. Feedforward mélytanulási rendszerek esetén ez a módszer úgy alkalmazható, hogy a rétegek egymástól függetlenül felügyelet nélkül tanulnak, az egymásra épülő rétegek egyre bonyolultabb tulajdonságok felismerésére képesek, és az a cél, hogy minden réteg rekonstruálni tudja az alatta levő réteg által felismert egyszerűbb tulajdonságokat. Ezzel a módszerrel a neurális háló "előtanítható" (pre-training), és megfelelő kiindulási értékeket kaphatunk vele a súlyokra. Ezek után következhet a hagyományos backpropagation módszerrel történő optimalizáció. Ez a módszer különösen hasznos, ha kevés felcímkézett adat áll rendelkezésre. Az első jelentősebb alkalmazása a beszédfelismerés területén történt, segítségével a korábbi módszerekhez képest 10-20-szor gyorsabb tanulást értek el.

A feedforward mélytanulási rendszerek egy speciális típusa különösen könnyen tanítható és jól teljesít, főleg képfeldolgozás területén, ezek a konvolúciós neurális hálók (CNN). Az eredményes alkalmazhatóságuk négy kulcsfontosságú ötleten alapszik. Lokális kapcsolatokat használ, mivel az egymáshoz közeli területek (például pixelek) gyakran hasonló tulajdonságokkal rendelkeznek, így célszerű őket együtt vizsgálni, egy egységként kezelni. Ezekre az egységekre diszkrét konvolúciót alkalmaznak, majd valamilyen nemlineáris függvényt, például ReLU-t. Egy másik fontos ötlet, hogy több területen ugyanazokat a súlyokat használja. Ez azért fontos, mivel egy képen belül gyakran megjelennek ugyanolyan típusú egységek (például egy kutya két szeme), így ezeket a mintákat könnyebben felismeri a hálózat. A pooling fontos a dimenziócsökkentés, és ezáltal a túlillesztés elkerülése miatt. Egy összetartozó egység jellemzője általában jól megfogható a tulajdonságok maximumával vagy átlagával, ezt a jellemzőt szűri ki a max vagy az average pooling, illetve segítségével jobb invariánst kapunk, ami kisebb változásokra érzéketlenebb. Az utolsó fő gondolat az, hogy sok réteget használjunk. Kétszer-háromszor követi egymást váltakozva egy konvolúciós, egy nemlineáris és egy pooling réteg, majd ezek után több konvolúciós és teljes kapcsolatrendszerrel rendelkező réteg következik.

3.2. Szekvenciális adatok feldolgozása és kódoló-dekódoló rendszerek

Szekvenciális adatok feldolgozásához (ha az inputsorozat elemenként kerül feldolgozásra), például beszéd- és természetes nyelvi feldolgozáshoz, kiválóan használhatók a rekurrens neurális hálózatok (RNN). Egy ilyen neurális háló egyesével dolgozza fel az inputelemeket, és a rejtett egységeiben minden lépésben eltárol egy állapotvektort, ami implicit módon minden korábbi inputelemről tartalmaz információt. Tulajdonképpen az egészre tekinthetünk úgy is, hogy diszkrét idő-

közönként olvassuk be az inputelemeket, és mindig kapunk egy-egy outputot. A diszkrét időközönként kapott outputok megfeleltethetők egy mélytanulási rendszer neuronjai által adott outputoknak. Ezzel az azonosítással pedig alkalmazható a tanulásnál a backpropagation módszere.

A rekurrens neurális hálózatok nehezen tanulnak, mivel a backpropagation módszer alkalmazása gyakran problematikus: egy bizonyos idő elteltével a gradiensek eltűnhetnek, illetve elszállhatnak. Az eltűnő gradiensek megnehezítik, hogy a rendszer hosszú távon eltárolja a korábban kinyert információkat, míg az elszálló gradiensek a tanulási folyamatot instabillá téve megakadályozhatják a paraméterek konvergenciáját, így fals eredményeket adhat a modell [2].

A rekurrens neurális hálózatok fordításra is taníthatók. Tekintsünk egy angol mondatot, majd olvassuk be egyesével a mondat szavait egy angol kódoló hálózattal. Ez a neurális háló azt csinálja, hogy minden szó beolvasása után egy állapotvektort ad kimenetként, és így az utolsó állapotvektor a mondat által kifejezett gondolatot reprezentálja. Vegyünk ezután egy - az angol kódoló hálózattal együtt tanított - francia dekódoló hálózatot. Ez a dekóder kezdeti rejtett állapotként (vagy extra inputként) megkapja a kódoló hálózat által generált végső állapotvektort. A dekóder valamilyen valószínűség-eloszlásból kiválasztja a mondat francia fordításának első szavát. Miután kiválasztotta a rendszer az első szót, és ezt megkapta inputként a dekóder, a hálózat outputja a második szó valószínűség-eloszlása lesz. Ez a folyamat addig folytatódik, amíg ki nem választjuk a mondatot záró írásjelet. Tulajdonképpen a fordítási folyamat során francia szavakat generálunk az angol mondattól függő valószínűség-eloszlásokból. Ez a - látszólag naiv - megközelítés hamar versenyképes módszerré vált.

Nemcsak mondatokat fordíthatunk le ezzel a módszerrel egy másik nyelvre, hanem képek tartalmát is szavakba önthetjük. Ennél az eljárásnál a kódoló egy konvolúciós neurális háló, ami a pixelek által tartalmazott információt egy vektorba

tömöríti az utolsó rejtett rétegben; a dekóder pedig egy rekurrens neurális háló, ami az információt tartalmazó vektor alapján egy a kép tartalmát összefoglaló mondatot generál.

Elméleti és gyakorlati bizonyítékok is azt mutatják, hogy ezek a rendszerek nehezen tárolnak el hosszú ideig információt. A probléma egy lehetséges megoldása, hogy kibővítjük a hálózatot egy explicit memóriával. Ezt valósítják meg a hosszú rövidtávú memória (long short-term memory, LSTM) hálózatok. Ezekben a hálózatokban van egy speciális rejtett egység (memóriacella), ami azért felel, hogy hosszú távon megőrizze az inputok tartalmát. A memóriacella sok állapoton keresztül megőrzi a fontos információkat, így nem következik be adatvesztés az eltűnő gradiensek problémája miatt. Ezt úgy valósítja meg, hogy a memóriacella kapcsolatban áll önmagával (self-connection), ami azt jelenti, hogy 1-es súllyal változatlanul átadja az aktuális állapotát a következő állapotnak. Ezt az információátadást, illetve a bejutó új információt azonban úgynevezett kapuk (gates) szabályozzák, amelyek meghatározzák, hogy a régi információ mekkora hányadát tartsa meg a rendszer, illetve az új információ mekkora részét engedje be. Az LSTM hálózatok hatékonyabban teljesítenek, mint a hagyományos elődeik, különösen beszédfelismerési problémáknál, amikor az akusztikus forráshoz elkészítik az átiratot.

Az RNN és az LSTM hálózatok hátránya, hogy a hosszú szekvenciális adatok feldolgozása során nem lehet a folyamatot paralelizálni az adatok természetéből kifolyólag. Ezt a problémát oldja meg a transzformátor alkalmazása, amelynek a működése egy úgynevezett figyelmi mechanizmuson (attention mechanism) alapszik. A figyelmi mechanizmus lehetővé teszi, hogy a modell az inputsorozat különböző részeire figyeljen (helyezzen súlyt) annak függvényében, hogy az input mely részei fontosak az adott lépésben. Ezt a mechanizmust a figyelmi függvény (attention function) határozza meg, amely egy lekérdezéshez (query) és néhány kulcs-érték

párhoz (key-value pair) rendel valamilyen outputot. Az output az értékek súlyozott összegeként adódik, ahol a súlyokat egy a lekérdezésre és a neki megfelelő kulcsra alkalmazott kompatibilitási függvény (compatibility function) határozza meg. Gyakran alkalmazott figyelmi függvény a scaled dot-product attention, illetve a multi-head attention, amelynél több projekcióra (amelyeket lineáris leképezésekkel kapunk) alkalmazzuk ugyanazt a figyelmi függvényt, az eredményeket konkatenáljuk, majd erre alkalmazunk egy újabb projekciót.

A figyelmi mechanizmusok egyik fontos típusa az önfigyelem (self-attention). Az önfigyelem segítségével a gép az inputsorozat különböző részeire tud figyelni, így képes felismerni a szekvenciális input különböző részei közötti kapcsolatokat az egymástól való távolságuktól függetlenül. Ez a megközelítés segít elrugaszkodni az input hagyományos, lineáris feldolgozásától. A transzformátor működése teljesen az önfigyelmi mechanizmuson alapszik, ennek segítségével adja meg az input és az output reprezentációját szekvenciális RNN, illetve konvolúció használata nélkül.

A transzformátor szerkezetileg egy kódoló és egy dekódoló egységből áll, mindkét része önfigyelmi mechanizmust használó, illetve teljesen összekapcsolt rétegekből épül fel. A kódoló egység N darab azonos rétegből áll, egy rétegen belül pedig két alréteg található. Az első alréteg multi-head self-attention mechanizmust használ, míg a második alréteg egy pozíciónként teljesen összekapcsolt feedforward hálózat. Mindkét alrétegre reziduális kapcsolatokat alkalmazunk, majd a kapott eredményt normalizáljuk. Ez azt jelenti, hogy az alréteg outputjához hozzáadjuk az inputot, majd erre alkalmazunk egy normalizáló függvényt (az átlaggal és a tapasztalati szórással korrigálunk). A dekódoló egység szintén N darab azonos rétegből áll. A dekódoló egység rétegeinek a felépítése hasonlít a kódoló egység rétegeiére, a különbség annyi, hogy a két - korábban ismertetett - alréteg közé bekerül egy újabb alréteg, amely multi-head attention mechanizmussal hat a kódoló egységtől kapott outputra. Minden alrétegre reziduális kapcsolatokat alkalmazunk,

majd a kapott eredményt normalizáljuk - a kódoló egységnél ismertetett eljáráshoz hasonlóan. Ezenfelül úgy módosítjuk az önfigyelmi mechanizmust használó alréteget, hogy az csak az aktuális pozíciónál kisebb indexű pozíciókra tudjon figyelni, azaz egy eltolást, illetve maszkolást alkalmazunk a magasabb indexű pozíciókra, ezáltal az aktuális pozícióra tett előrejelzés csupán a korábbi, ismert pozíciók kimeneteitől függ. A dekóder outputjára még egy lineáris függvényt, majd egy softmax függvényt alkalmazunk.

A transzformátor egy táblázatos adatok feldolgozására alkalmas adaptációja - amely már aktuáriusi feladatok elvégzéséhez is használható - a feature tokenizer transformer (FTT). Az FTT rendszer egy feature tokenizer, több transzformátor, illetve egy előrejelző egységből áll. Az FTT inputként numerikus, illetve kategorikus adatokat kap, majd ezeket a feature tokenizer egy beágyazásmátrixszá alakítja. Numerikus, illetve kategorikus adatok esetén is az inputokat azonos dimenziós adatokká alakítjuk: numerikus adatok esetén egy vektorral szorozzuk (elemenkénti szorzással) az inputot, kategorikus adatok esetén pedig minden inputhoz (bemenettől függően) egy-egy előre meghatározott vektort rendelünk. A transzformáció után adódó vektorokhoz hozzáadunk egy-egy konstans vektort. Az így kapott vektorokat egy mátrixba rendezzük, így kapjuk meg a beágyazásmátrixot. A beágyazásmátrix elejére appendáljuk az úgynevezett [CLS] (klasszifikációs) token beágyazását. A [CLS] token az inputsorozatról tartalmaz információt, amit megőriz a transzformátorok által történő feldolgozás alatt is. Az így kapott mátrixot kapja meg az első transzformátor inputként. Ennek a transzformátornak az outputja lesz a következő transzformátor egység inputja és így tovább. Az utolsó transzformátor outputját dolgozza fel az előrejelző egység: az outputot először normalizálja, majd egy ReLU-t, ezután pedig egy lineáris függvényt alkalmaz rá, így kapjuk meg a végső outputot.

4. Mélytanulási módszerek a nem-életbiztosításban

A nem-életbiztosítás területén számos probléma megoldásához alkalmaznak általánosított lineáris modelleket (GLM). Ezek közé tartozik a kárgyakoriság, illetve a kárnagyság előrejelzése Poisson, illetve Gamma GLM-mel, továbbá a törlés, illetve a potenciális ügyfelek bevonzásának a modellezése logisztikus regresszióval. A neurális hálókat használó módszerek rendszerint GLM alapú modellekre építenek, így megtartják a hagyományos aktuáriusi modellek előnyeit, ugyanakkor javítanak a segítségükkel kapott előrejelzések pontosságán, mivel kihasználják a mélytanulási módszerek előnyeit. A biztosítók által előszeretettel alkalmazott GLM alapú modellek kiválóan kombinálhatók a korábban bemutatott transzformátor, jelesül FTT alapú rendszerekkel. Ebben a fejezetben példákat mutatok a mélytanulási módszerek aktuáriusi területen való használatára: bemutatom néhány módszer elméleti hátterét, illetve megvizsgálom az alkalmazhatóságukat valós adatokon. Pythonban implementálom a tárgyalt modelleket, majd összehasonlítom a kapott eredményeket. Az összefoglaló, illetve a reprodukció alapjául szolgáló Python kód az [5] cikken alapszik.

4.1. Aktuáriusi modellek

Számos modell kiindulópontja az általánosított lineáris modell (GLM) [16], így elengedhetlen a megismerése. Tegyük fel, hogy numerikus változóink vannak (vagy ha nem, akkor már numerikussá alakítottuk őket). A modell alapötlete az, hogy a várható érték egy transzformáltja a prediktorok lineáris függvénye. A modell az alábbi formulával írható le:

$$\hat{y} = g^{-1}(\beta_0 + \langle \boldsymbol{\beta}, \boldsymbol{x} \rangle),$$

ahol $\langle ., . \rangle$ a skaláris szorzatot jelöli, β_0 a tengelymetszet, $\boldsymbol{\beta} \in \mathbb{R}^k$ a regressziós paraméterekből álló vektor, $\boldsymbol{x} \in \mathbb{R}^k$ a tulajdonságokból álló vektor, g az úgyneve-

zett kapocsfüggvény, amely szigorúan monoton nő, továbbá $\hat{y}$ az eredményváltozó. Kapocsfüggvényként gyakran használják a logaritmusfüggvényt, például a Gamma GLM-nél, illetve a Poisson GLM-nél is.

A Poisson-regressziónál kitettségekkel is súlyozzák az eredményváltozót, így az alábbi formula adódik:

$$\hat{y}_i = \exp\left(\beta_0 + \langle \boldsymbol{\beta}, \boldsymbol{x_i} \rangle\right) v_i,$$

ahol v_i az i-edik megfigyeléshez tartozó kitettség.

A lineáris tagokat gyakran spline-okkal helyettesítik, így bonyolultabb összefüggések is modellezhetők. $s_j(x_j)$ jelöli az x_j-hez tartozó spline tagot. Ezzel eljutunk az általánosított additív modellhez (GAM):

$$\hat{y} = g^{-1}\left(\beta_0 + \sum_{j=1}^{k} \beta_j s_j(x_j)\right).$$

A feedforward neurális hálók a GLM általánosításának tekinthetők, így aktuáriusi területen is alkalmazhatók. Legyen $d \in \mathbb{N}$ a rejtett rétegek száma, az m-edik réteg (ahol $m \in \{1, \ldots, d\}$) q_m darab neuronból áll, $w_0^{(d+1)} \in \mathbb{R}$, $\boldsymbol{w}^{(d+1)} \in \mathbb{R}^{q_d}$ az utolsó rejtett réteg kimenetéhez tartozó súlyok és h az aktivációs függvény. A modell az alábbi formulával írható le:

$$\hat{y} = h\left(w_0^{(d+1)} + \left\langle \boldsymbol{w}^{(d+1)}, \left(z^{(d)} \circ \ldots \circ z^{(1)}\right)(\boldsymbol{x}) \right\rangle\right),$$

ahol a rétegek közötti kapcsolatokat az alábbi függvényekkel írjuk le:

$$z^{(m)} : \mathbb{R}^{q_{m-1}} \to \mathbb{R}^{q_m}$$

$$\boldsymbol{x} \mapsto z^{(m)}(\boldsymbol{x}) = \left(z_1^{(m)}(\boldsymbol{x}), \ldots, z_{q_m}^{(m)}(\boldsymbol{x})\right)^T,$$

ahol az m-edik réteg neuronjai által adott előrejelzés-vektor egy eleme így írható le:

$$z_j^{(m)}(\boldsymbol{x}) = \phi_m\left(w_{0,j}^{(m)} + \left\langle \boldsymbol{w}_j^{(m)}, \boldsymbol{x} \right\rangle\right).$$

Itt $\phi_m : \mathbb{R} \to \mathbb{R}$ a megfelelő aktivációs függvény, $w_{0,j}^{(m)} \in \mathbb{R}$ a megfelelő eltolásparaméter, $\boldsymbol{w}_j^{(m)} \in \mathbb{R}^{q_{m-1}}$ pedig a megfelelő súlyvektor.

Poisson-regresszió esetén az aktivációs függvényt jellemzően exponenciálisnak választják, a neurális hálók kimeneteit pedig kitettségekkel súlyozzák, az i-edik kimenetet v_i-vel:

$$\hat{y}_i = \exp\left(w_0^{(d+1)} + \left\langle \boldsymbol{w}^{(d+1)}, \left(z^{(d)} \circ \cdots \circ z^{(1)}\right)(\boldsymbol{x}_i)\right\rangle\right) v_i.$$

A neurális hálók paramétereit a sztochasztikus gradiens ereszkedés módszerének különböző változataival határozzák meg, és a túlillesztés elkerülése érdekében különféle korai megállási kritériumokat használnak.

A feedforward neurális hálók és a GLM ötvözésével kapjuk a kombinált aktuáriusi neurális hálót (CANN). A modell a GLM-nél, illetve a neurális hálónál kapott utolsó (a függvénnyel való ráhatás előtti) bemenetet összegzi, majd erre alkalmazza az aktivációs függvényt:

$$\hat{y} = g^{-1}(\eta_{\text{GLM}} + \eta_{\text{FNN}}),$$

ahol

$$\eta_{\text{GLM}} = \beta_0 + \langle \boldsymbol{\beta}, \boldsymbol{x} \rangle,$$

továbbá

$$\eta_{\text{FNN}} = w_0^{(d+1)} + \left\langle \boldsymbol{w}^{(d+1)}, \left(z^{(d)} \circ \ldots \circ z^{(1)}\right)(\boldsymbol{x})\right\rangle.$$

Fontos hangsúlyozni, hogy az aktivációs függvény bemenetét adó η_{GLM} és η_{FNN} tagokat tulajdonképpen fix, 1-es súlyokkal kombináljuk, azaz ezeket a súlyokat nem optimalizáljuk a tanulási folyamat során.

A tanulási folyamathoz a neurális háló megkapja kiindulási paraméterekként a GLM (már korábban illesztett) paramétereit. A neurális háló utolsó rejtett rétegéhez tartozó súlyokat ($w_0^{(d+1)}$-et és $\boldsymbol{w}^{(d+1)}$-et) pedig 0-ra inicializáljuk, így a kezdeti

súlyokkal nem befolyásoljuk a tanulási folyamatot. A GLM súlyait általában rögzítjük, azokat a neurális háló nem módosíthatja. Tulajdonképpen a GLM adja a CANN modell alapját, a neurális háló pedig kiegészíti, finomítja azt. Mivel a neurális háló a hibát tanulja meg, reziduális modellnek tekinthető.

Poisson-regresszió esetén úgy is eljárhatunk, hogy a modell alapjául egy feedforward neurális háló szolgál, és ennek a kimenetét súlyozzuk a v_{GLM} kitettséggel, aminek az értékét egy GLM segítségével kapjuk meg:

$$\hat{y}_i = \exp\left(w_0^{(d+1)} + \left\langle \boldsymbol{w}^{(d+1)}, \left(z^{(d)} \circ \cdots \circ z^{(1)}\right)(\boldsymbol{x}_i)\right\rangle\right) v_{\mathrm{GLM}}(\boldsymbol{x}_i).$$

Itt a v_{GLM} kitettség egy Poisson GLM eredményeként adódik:

$$v_{\mathrm{GLM}}(\boldsymbol{x}_i) = \exp\left(\beta_0 + \langle \boldsymbol{\beta}, \boldsymbol{x_i}\rangle\right) v_i,$$

ahol v_i az i-edik megfigyeléshez tartozó (szokásos) kitettség.

A LocalGLMnet modell a GLM módosításával adódik. Ez a rendszer tulajdonképpen egy olyan GLM, amelynek az együtthatóit egy feedforward neurális háló kimeneteként kapjuk meg:

$$\hat{y} = g^{-1}(\beta_0 + \langle \boldsymbol{\beta}(\boldsymbol{x}), \boldsymbol{x}\rangle),$$

ahol $\boldsymbol{\beta}(\boldsymbol{x})$ egy olyan d darab rejtett rétegből álló feedforward neurális háló outputja, amely bemenetének és kimenetének a dimenziója megegyezik, k-val egyenlő:

$$\boldsymbol{\beta} : \mathbb{R}^k \to \mathbb{R}^k$$

$$\boldsymbol{\beta}(\boldsymbol{x}) = \left(z^{(d)} \circ \ldots \circ z^{(1)}\right)(\boldsymbol{x}).$$

A modell neve onnan ered, hogy $\boldsymbol{x}$ egy kis környezetében $\boldsymbol{\beta}(\boldsymbol{x})$ egy konstans $\boldsymbol{\beta}$-val közelíthető, és így lokálisan GLM-nek tekinthető. Amennyiben rendelkezésre állnak egy GLM regresszióval kapott együtthatói, akkor ezeket használhatjuk a neurális háló kiindulási paramétereiként. Ennek az inicializációnak az előnye,

hogy egy lényegében helyes modellből indulunk ki, és a neurális háló ezt fejleszti tovább. A modell szerkezetéből adódóan külön-külön jól megfigyelhető az egyes tulajdonságok kimenetre gyakorolt hatása. Ez kifejezetten hasznos az aktuáriusi területen, mivel fontos megérteni, hogy az egyes változók hogyan befolyásolják az eredményt.

Poisson-regressziónál a GLM értelemszerű módosításával a várt formula adódik:

$$\hat{y}_i = \exp\left(\beta_0 + \langle \boldsymbol{\beta}(\boldsymbol{x_i}), \boldsymbol{x_i} \rangle\right) v_i,$$

ahol v_i az i-edik megfigyeléshez tartozó kitettség.

Aktuáriusi modellekhez használhatjuk a feature tokenizer transformert, a következőkben ennek a felépítését mutatom be részletesen. Legyen $\mathbb{X} = \mathbb{X}_1 \times \cdots \times \mathbb{X}_k$ az állapottér, ennek az eleme a k tulajdonságot tartalmazó $\boldsymbol{x} = (x_1, \ldots, x_k)^T$ vektor, amely numerikus, illetve kategorikus adatokat is tartalmazhat. Amennyiben a j-edik tulajdonság (ahol $j \in \{1, \ldots, k\}$) numerikus, akkor $\mathbb{X}_j = \mathbb{R}$, ha pedig kategorikus, akkor $\mathbb{X}_j$ az x_j tulajdonság lehetséges értékeiből áll. A tulajdonságokat fix dimenziójú vektorokká transzformáljuk, legyen d_{emb} ez a közös beágyazási dimenzió. Ekkor a feature tokenizer által végrehajtott beágyazás az alábbi módon írható le a j-edik tulajdonságnál:

$$\text{FT}_j(x_j) : \mathbb{X}_j \to \mathbb{R}^{1 \times d_{\text{emb}}}.$$

Amennyiben a j-edik bemenet kategorikus, és $|\mathbb{X}_j| = S_j$ a j-edik tulajdonság lehetséges értékeinek száma, akkor a beágyazás az alábbi képlettel írható le:

$$\text{FT}_j(x_j) = e_j^T W_j^{(\text{cat})} + b_j,$$

ahol e_j az adott tulajdonsághoz tartozó indikátorvektor, $W_j^{(\text{cat})} \in \mathbb{R}^{S_j \times d_{\text{emb}}}$ az a mátrix, amely tartalmazza a j-edik tulajdonság összes lehetséges értékéhez tartozó beágyazási értékeket, $b_j \in \mathbb{R}^{1 \times d_{\text{emb}}}$ pedig a megfelelő konstans vektor. Amennyiben a j-edik bemenet numerikus, a beágyazást egy feedforward neurális háló végzi,

amelynek a kimeneti rétege d_{emb} dimenziós. A beágyazás ekkor az alábbi formulával adható meg:

$$\text{FT}_j(x_j) = x_j W_j^{(\text{num})} + b_j,$$

ahol $W_j^{(\text{num})} \in \mathbb{R}^{1\times d_{\text{emb}}}$, $b_j \in \mathbb{R}^{1\times d_{\text{emb}}}$ pedig a megfelelő konstans vektor. A tokenizációval kapott vektorokat egy mátrixba rendezzük, amelynek az elejére appendáljuk a random inicializált tanítható súlyokból álló [CLS] $\in \mathbb{R}^{1\times d_{\text{emb}}}$ tokent:

$$\text{Stack}_{\text{Input}}(\boldsymbol{x}) = \begin{bmatrix} \text{CLS} \\ \text{FT}_1(x_1) \\ \vdots \\ \text{FT}_k(x_k) \end{bmatrix} \in \mathbb{R}^{(k+1)\times d_{\text{emb}}}.$$

Ezt a mátrixot kapja meg inputként az első transzformátor réteg. Az utolsó transzformátor réteg outputja az alábbi formulával írható le:

$$\text{Stack}_{\text{Output}}(\boldsymbol{x}) = \left(T_t^{\text{Block}} \circ \cdots \circ T_1^{\text{Block}}\right)(\text{Stack}_{\text{Input}}(\boldsymbol{x})),$$

ahol az l-edik ($l \in \{1,\ldots,t\}$) transzformátor hatása egy

$$T_l^{\text{Block}} : \mathbb{R}^{(k+1)\times d_{\text{emb}}} \to \mathbb{R}^{(k+1)\times d_{\text{emb}}}$$

leképezéssel adható meg. A $\text{Stack}_{\text{Output}}(\boldsymbol{x}) \in \mathbb{R}^{(k+1)\times d_{\text{emb}}}$ mátrix [CLS] tokenhez tartozó (itt az első) $\text{Stack}_{\text{Output}}(\boldsymbol{x})_1 \in \mathbb{R}^{1\times d_{\text{emb}}}$ sorát dolgozza fel az előrejelző egység, azaz először normalizálja azt, erre elemenkénti ReLU aktivációt alkalmaz, majd egy sűrű utolsó réteg következik a h aktivációs függvénnyel. Ezek alapján a végső output az alábbi alakú:

$$\hat{y} = h\left(w_0 + \left\langle \boldsymbol{w}, \text{ReLU}\left(\text{Norm}_{\text{Layer}}\left(\text{Stack}_{\text{Output}}(\boldsymbol{x})_1\right)\right)^T \right\rangle\right),$$

ahol $w_0 \in \mathbb{R}$ és $\boldsymbol{w} \in \mathbb{R}^{d_{\text{emb}}}$.

Poisson-regressziónál az alábbi formulával írható le a módszer:

$$\hat{y}_i = \exp\left(w_0 + \left\langle \boldsymbol{w}, \text{ReLU}\left(\text{Norm}_{\text{Layer}}\left(\text{Stack}_{\text{Output}}(\boldsymbol{x}_i)_1\right)\right)^T \right\rangle\right) v_i,$$

ahol v_i az i-edik megfigyeléshez tartozó kitettség.

Az FTT alapú modell is általánosítható a CANN mintájára, így jutunk a kombinált aktuáriusi FTT (CAFTT) módszeréhez. A CAFTT-t a CANN-hez hasonlóan építjük fel azzal a különbséggel, hogy itt feedforward neurális háló helyett FTT-t használunk:

$$\hat{y} = g^{-1}(\eta_{\text{GLM}} + \eta_{\text{FTT}}),$$

ahol

$$\eta_{\text{GLM}} = \beta_0 + \langle \boldsymbol{\beta}, \boldsymbol{x} \rangle,$$

továbbá

$$\eta_{\text{FTT}} = w_0 + \left\langle \boldsymbol{w}, \text{ReLU}\left(\text{Norm}_{\text{Layer}}\left(\text{Stack}_{\text{Output}}(\boldsymbol{x})_1\right)\right)^T \right\rangle.$$

Ebben az esetben is az aktivációs függvény bemenetét adó η_{GLM} és η_{FTT} tagokat fix, 1-es súlyokkal kombináljuk, azaz ezeket a súlyokat nem optimalizáljuk a tanulási folyamat során.

A CANN-hez hasonlóan megkapja az FTT kiindulási paraméterekként a GLM (már korábban illesztett) paramétereit a tanulási folyamathoz. Az előrejelző réteghez tartozó súlyokat (w_0-t és $\boldsymbol{w}$-t) pedig 0-ra inicializáljuk, így a kezdeti súlyokkal nem befolyásoljuk a tanulási folyamatot. A GLM súlyait általában rögzítjük, azokat az FTT nem módosíthatja. Ez eseben is a GLM adja a modell alapját, az FTT pedig kiegészíti, finomítja azt. Mivel az FTT a hibát tanulja meg, reziduális modellnek tekinthető.

Poisson-regresszió esetén a CANN-hez hasonló módon kapjuk meg a modellt:

$$\hat{y}_i = \exp\left(w_0 + \left\langle \boldsymbol{w}, \text{ReLU}\left(\text{Norm}_{\text{Layer}}\left(\text{Stack}_{\text{Output}}(\boldsymbol{x}_i)_1\right)\right)^T \right\rangle\right) v_{\text{GLM}}(\boldsymbol{x}_i).$$

Itt a v_{GLM} kitettség egy Poisson GLM eredményeként adódik:

$$v_{\text{GLM}}(\boldsymbol{x}_i) = \exp\left(\beta_0 + \langle \boldsymbol{\beta}, \boldsymbol{x_i} \rangle\right) v_i,$$

ahol v_i az i-edik megfigyeléshez tartozó (szokásos) kitettség.

FTT alkalmazásával tovább általánosítható a LocalGLMnet modell, így jutunk a LocalGLMftt-hez. Az előrejelzés a LocalGLMnet-nél látottakhoz hasonlóan írható le:

$$\hat{y} = g^{-1}(\beta_0 + \langle \boldsymbol{\beta}^{\mathrm{FTT}}(\boldsymbol{x}), \boldsymbol{x} \rangle),$$

ahol $\boldsymbol{\beta}^{\mathrm{FTT}}(\boldsymbol{x})$ egy olyan FTT outputja, amely bemenetének és kimenetének a dimenziója megegyezik, k-val egyenlő:

$$\boldsymbol{\beta}^{\mathrm{FTT}} : \mathbb{R}^k \to \mathbb{R}^k$$

$$\boldsymbol{\beta}^{\mathrm{FTT}}(\boldsymbol{x}) = z\left(\mathrm{ReLU}\left(\mathrm{Norm}_{\mathrm{Layer}}\left(\mathrm{Stack}_{\mathrm{Output}}(\boldsymbol{x})_1\right)\right)^T\right),$$

ahol $z : \mathbb{R}^{d_{\mathrm{emb}}} \to \mathbb{R}^k$ annak a sűrű rétegnek a hatását írja le, amely az FTT k dimenziós outputját adja eredményül. A modell a LocalGLMnet-éihez hasonló előnyös tulajdonságokkal rendelkezik.

Poisson-regressziónál a LocalGLMnet modellnél látott formula értelemszerűen módosított verziója adódik:

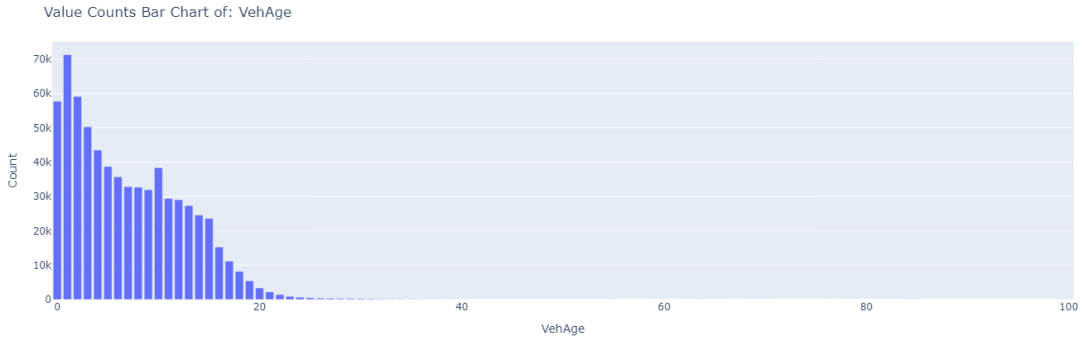
$$\hat{y}_i = \exp\left(\beta_0 + \langle \boldsymbol{\beta}^{\mathrm{FTT}}(\boldsymbol{x_i}), \boldsymbol{x_i} \rangle\right) v_i,$$

ahol v_i az i-edik megfigyeléshez tartozó kitettség.

4.2. A modellek implementálása és az eredmények értelmezése

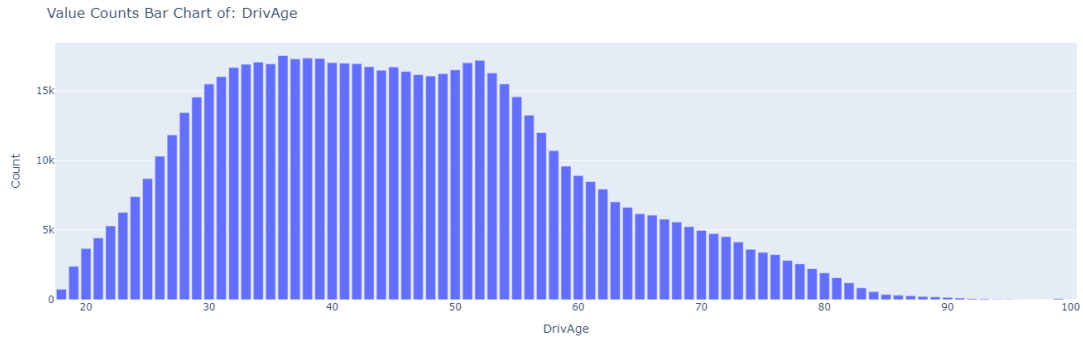
A fentebb ismertetett modelleket francia gépjármű-felelősségbiztosításhoz tartozó kárgyakorisági adatokon implementáltam (French motor third-party liability, MTPL) [9], ezt az adathalmazt gyakran használják aktuáriusi területen. Az adatokat R-ben töltöttem le, majd Pythonban implementáltam a modelleket. Az adathalmaz (adattisztítás előtt) 678 013 megfigyelésből áll, minden megfigyelésnek egyedi azonosítója van, tartozik hozzá egy eredményváltozó (kárgyakoriság),

egy kitettség érték, illetve 9 magyarázó változó. A magyarázó változók közül 2 kategorikus, 6 numerikus és 1 Boolean típusú. Az adattisztítási és -előkészítési lépéseket szintén az [5] cikkben ismertetett módon végeztem el. A numerikus változókat átskáláztam, míg a kategorikus adatokra one-hot encodingot alkalmaztam (kivéve a transzformátoron alapuló modelleknél, mivel azoknál kategorikus beágyazást alkalmaztam one-hot encoding helyett). Az adatok 90%-át a tanításhoz, míg a maradék 10%-át validációhoz használtam, miként ezen adathalmaz használata esetén szokás. Az alábbiakban három hisztogrammal szemléltetem az adathalmaz tulajdonságait:



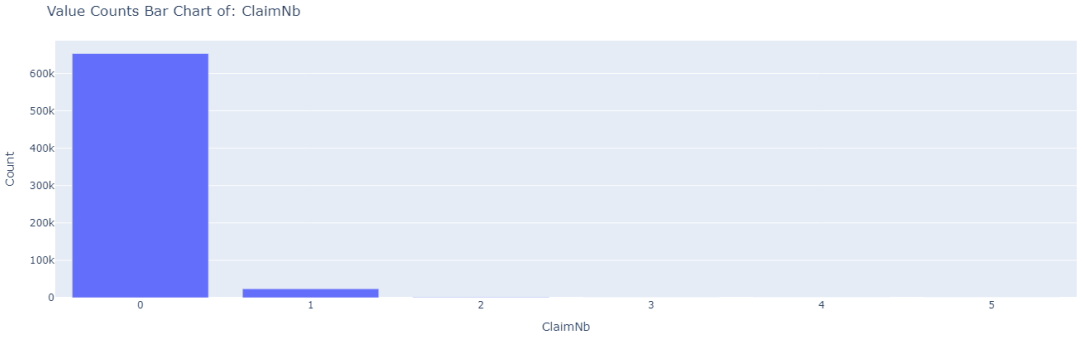
1. ábra. Szerződésszám a jármű korának a függvényében

Az 1. ábrán látható, hogy 1 éves autókra kötötték a legtöbb szerződést, majd néhány outliertől eltekintve jellemzően idősebb autókra egyre kevesebb szerződést kötöttek.



2. ábra. Szerződésszám a sofőr korának a függvényében

A 2. ábrán látható, hogy 31 és 53 éves kor között jellemzően hasonló a szerződésszám, fiatalabb sofőr esetén pedig növekvő, míg idősebb sofőr esetén csökkenő tendenciát mutat.



3. ábra. Szerződésszám a kárszám függvényében

A 3. ábrán látható, hogy a legtöbb szerződő kármentes, egy kárt csupán a szerződők 3%-a okozott, a magasabb kárszámok pedig elenyészőek.

A modelleket az [5] cikk alapján implementáltam Pythonban a 3.11.9-es verziót használva. A modellek összehasonlításához a Poisson-veszteségfüggvényt használtam. Kárgyakoriságokkal dolgoztam, így a Poisson GLM-eknél logaritmusfüggvény a kapocsfüggvény. Mindegyik modelltípus esetén 15 modellt illesztettem különböző

seedekkel, majd ezeknek az eredményét átlagoltam.

Háromféle GLM-et illesztettem: az elsőnél a szokásos előkészítési módszert alkalmaztam (a numerikus adatokat intervallumokra bontottam (binning), míg a kategorikus adatokra dummy változókat vezettem be), a másodiknál már transzformáltam is a numerikus adatokat (logaritmikus és polinomiális tagok), míg a harmadiknál a különböző változók kölcsönhatását leíró tagokat is bevezettem.

Kétféle neurális hálót tanítottam. Az FNN_{OHE}-nél one-hot encodingot alkalmaztam a bemenetre, míg az FNN_{EMB}-nél a kategorikus változókra kétdimenziós beágyazást alkalmaztam. A modellek három-három rejtett rétegből állnak, melyek dimenziója (20, 15, 10). Az utolsó réteget leszámítva minden rétegnél tangens hiperbolikusz aktivációs függvényt alkalmaztam (ez előnyös választásnak bizonyul, mivel korlátos és páratlan függvény), míg az utolsó rétegnél az aktivációs függvény exponenciális.

A CANN modellhez a harmadik típusú GLM-et, illetve az FNN_{EMB} modellt használtam.

A LocalGLMnet modellhez egy az első GLM-hez, illetve az FNN_{OHE}-hez hasonló modellt használtam.

Az FTT modellnél 3 transzformátor blokkot használtam 8 attention headdel. Az FFN aktivációs függvénye ReGLU (ez a ReLU-hoz hasonlóan teljesít).

A CAFTT az FTT és a CANN modell értelemszerű ötvözésével kapható meg.

A LocalGLMftt modellt pedig a LocalGLMnet modellből kapjuk értelemszerűen FTT alkalmazásával.

A kapott eredményeket az alábbi táblázatban foglalom össze:

Model	**Train Loss**	**Test Loss**
Homogeneous Model	0.2518	0.2578
GLM_1	0.2405	0.2462
GLM_2	0.2404	0.2460
GLM_3	0.2403	0.2460
FNN_OHE	0.2370	0.2444
FNN_EMB	0.2368	0.2440
CANN	0.2365	0.2442
LocalGLMnet	0.2368	0.2446
FTT	0.2376	0.2448
CAFTT	0.2362	0.2434
LocalGLMftt	0.2366	0.2440

1. táblázat. A különféle modellekre adódó tanulási és validációs veszteségek

A veszteségek alapján jól látható, hogy a neurális hálót használó modellek jobban teljesítenek a GLM alapú modelleknél. Azt is megfigyelhetjük, hogy a CAFTT és a LocalGLMftt transzformátor alapú aktuáriusi modellek jobb eredményeket adnak, mint az FNN alapú párjaik, a CANN, illetve a LocalGLMnet. Ennek ellenére megállapíthatjuk, hogy a jelentősen hosszabb tanulási idő ellenére a transzformátor alapú modellek nem teljesítenek szignifikánsan jobban, mint az FNN alapú társaik.

Hivatkozások

[1] Shun-ichi Amari. Backpropagation and stochastic gradient descent method. *Neurocomputing*, 5(4-5):185–196, 1993.

[2] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

[3] Stefan Bernegger. The Swiss Re Exposure Curves and the MBBEFD Distribution Class1. *ASTIN Bulletin: The Journal of the IAA*, 27(1):99–111, 1997.

[4] Gérard Biau, Benoît Cadre, Quentin Paris, et al. Cox process learning. *Preprint*, 2013.

[5] Alexej Brauer. Enhancing actuarial non-life pricing models via transformers. *European Actuarial Journal*, pages 1–22, 2024.

[6] David G Champernowne. A model of income distribution. *The Economic Journal*, 63(250):318–351, 1953.

[7] David R Cox. Some statistical methods connected with series of events. *Journal of the Royal Statistical Society: Series B (Methodological)*, 17(2):129–157, 1955.

[8] Happiness Ugochi Dike, Yimin Zhou, Kranthi Kumar Deveerasetty, and Qingtian Wu. Unsupervised learning based on artificial neural network: A review. In *2018 IEEE International Conference on Cyborg and Bionic Systems (CBS)*, pages 322–327. IEEE, 2018.

[9] Christophe Dutang and Arthur Charpentier. *CASdatasets: Insurance Datasets*, 2018. R package version 1.0-8.

[10] Yury Gorishniy, Ivan Rubachev, Valentin Khrulkov, and Artem Babenko. Revisiting deep learning models for tabular data. *Advances in neural information processing systems*, 34:18932–18943, 2021.

[11] Geoffrey Grimmett and David Stirzaker. *Probability and Random Processes*. Oxford University Press, 2020.

[12] James J Higgins and Sallie Keller-McNulty. *Concepts in Probability and Stochastic Modeling*. Duxbury Press, 1995.

[13] Jean Jacod and Albert N Shiryaev. Limit Theorems for Stochastic Processes. 1989.

[14] Olav Kallenberg and Olav Kallenberg. *Foundations of modern probability*, volume 2. Springer, 1997.

[15] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[16] John Ashworth Nelder and Robert WM Wedderburn. Generalized linear models. *Journal of the Royal Statistical Society Series A: Statistics in Society*, 135(3):370–384, 1972.

[17] Pietro Parodi, Derek Thrumble, Peter Watson, Zhongmei Ji, Alex Wang, Ishita Bhatia, Joseph Lees, Sophia Mealy, Rushabh Shah, Param Dharamshi, et al. Loss modelling from first principles. In *CAS E-Forum*, 2024.

[18] Pietro Parodi and Peter Watson. Property graphs–a statistical model for fire and explosion losses based on graph theory. *ASTIN Bulletin: The Journal of the IAA*, 49(2):263–297, 2019.

[19] Hossein Pishro-Nik. *Introduction to Probability, Statistics, and Random Processes*. Kappa Research, LLC Blue Bell, PA, USA, 2014.

[20] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

[21] Sheldon M Ross. *Stochastic Processes*. John Wiley & Sons, 1995.

[22] Sheldon M Ross. *Introduction to Probability Models*. Academic press, 2014.

[23] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[24] Guo-Xun Yuan, Chia-Hua Ho, and Chih-Jen Lin. Recent advances of large-scale linear classification. *Proceedings of the IEEE*, 100(9):2584–2603, 2012.