Budapesti Műszaki és Gazdaságtudományi Egyetem

Közlekedésmérnöki és Járműmérnöki Kar

Anyagmozgatási és Logisztikai Rendszerek Tanszék

Tóth Gergő
RD7H5L

A neurális hálók logisztikai alkalmazhatósága

Tudományos Diákköri Konferencia

Budapest, 2017.

Absztrakt

Szerző: Tóth Gergő, RD7H5L

Azonosító:

Képzés: Logisztikai mérnök MSc

Tanszék: BME-KJK, Anyagmozgatási és Logisztikai Rendszerek Tanszék

Konzulens: Dr. Bóna Krisztián

A mesterséges intelligenciákat már számos, az ökonometriához szorosan köthető szakterületen sikerrel alkalmazták. A gyorsuló világunkban tapasztalható trendek alapján az látszik kirajzolódni, hogy mind a gazdasági és hétköznapi életben egyre inkább nagyobb szerephez jutnak a különféle neurális hálózatok. Ennek a dolgozatnak a célja feltárni a neurális hálózatok alkalmazhatóságát a keresletelőrejelző rendszerek tekintetében.

Az első fejezetben egy kellően átfogó képet kívánok alkotni a teljesség igénye nélkül az olyan mesterséges intelligencia architektúrákról, amelyek potenciálisan sikerrel alkalmazhatóak kereslettervező rendszerek készítéséhez, vagy pedig azok támogatására. Ebben a fejezetben továbbá igyekszem az olyan könyvtárakra is kitérni, amelyek szerves részét képezték munkámnak a programozás során.

A második fejezetben a hagyományos kereslettervező eszközrendszer kerül bemutatásra, kitérve a szükséges lépesekre, amelyek felmerülnek egy ilyen szisztéma felállításánál. A „state of the art” jellegű bemutatást követően pedig annak lehetőségét vizsgálom meg, hogy a neurális hálóknak milyen szerep juthat a kereslet előrejelzésben.

A harmadik és egyben utolsó fejezetben, ezt követően az általam lefolytatott kísérlettervet részletezem illetve ennek eredményeit. Az elemzések nyomán a lehetőségekhez mérten próbálok következtetéseket levonni a neurálishálók működéséről a paraméterek függvényében. Végül a további kutatási irányokat jelölöm ki a jövőre való tekintettel.

Kulcsszavak: neurális hálók, mesterséges intelligenciák, kereslettervezés, előrejelzés, identifikáció, adattisztítás, finomtervezés, Python, Tensorflow, Keras, visszacsatolt hálók, LSTM, ADAM, ARIMA, SARIMA, hibamutatók, korrelogram, autokorreláció, parciálisautokorreláció, tanítás, hibamutatók, MLP, perceptron, ökonometria, epoch, batch, neuron, tanulási ráta, statisztika, kísérlet

Tartalomjegyzék

Bevezetés

Napjaink társadalmát valósággal elárasztják az adatok, az életünk szerves részét képező technológiai eszközökön keresztül. Becslések alapján naponta több mint 2.5 trillió bájtnyi adatot generál a világ lakossága.[1] Az információs kor hajnalát követően szemünk előtt formálódik meg egy olyan szép új világ képe, ahol teljesen új perspektívák nyílnak meg az emberek előtt a számítógépes technológiák területén. A lehetőségek végtelen tárháza egyre inkább arra ösztökéli a világ úttörő tudósait, hogy egyre több energiát invesztáljanak a biológián alapuló számítástudományba.

A biológián alapuló számítástudomány alapjait a XX. század közepén kezdték el lefektetni. A számítógépes hardverek terén elért robbanásszerű fejlődés egyre komolyabb algoritmusok implementálását tették lehetővé. Jórészt a biológiai kutatások eredményeképpen merült fel az a gondolat, hogy a természetben előforduló jelenségek és rendszerek mintájára algoritmusokat konstruáljanak. A 70-es és 80-as évek tudományos eredményei olyan eljárásoknak alapozták meg a jövőjét, mint a neurális hálók vagy éppenséggel a genetikus algoritmusok. Ezek a módszerek a természetből ellesett minták, példák alapján nyert tapasztalatok útján valósítják meg a rájuk ruházott feladatok kivitelezését. Azon módszertanok közül, amelyek a természet mechanizmusait másolják, talán az egyik legfigyelemreméltóbb a neurális hálózatok tudománya. A neurális számítástechnika mára önálló tudománnyá vált, amely szilárd elméleti alapokkal, egyre szélesebb alkalmazási körrel és egyre több alkalmazási tapasztalattal rendelkezik.

A neurális hálózatok olyan számítási feladatok megoldására létrejött párhuzamos feldolgozást végző, adaptív eszközök, melyek eredete a biológiai rendszerektől származtatható. Az idegrendszer tanulmányozása, az idegsejt (neuron) felépítésének, illetve működésének valamilyen szinten való megismerése indította el azt a gondolkodást, hogy kíséreljünk meg az élő szervezetekben létező, bonyolult rendszerek mintájára létrehozni számító rendszereket. A biológiai, "természetes" neurális hálózatoknál nagyszámú, hasonló vagy azonos felépítésű, egymással összeköttetésben lévő építőelemekből, idegsejtekből felépülő hálózatok a legkülönfélébb feladatok ellátására bizonyulnak alkalmasnak.[2]

Egy tárgy, vagy egy arc felismerése korántsem jelent megerőltető feladatot az ember számára, viszont azt nehéz megmondani, hogy mi alapján történik a felismerés, kihívás lenne a folyamat pontos lépéseit definiálni és algoritmizálni olyan formában, hogy azt utána könnyen reprodukálni is lehessen. A biológia neurális rendszerek titkának a nyitja a tanulási és adaptációs képességünkben rejlik. Az előbb említett igencsak figyelemre méltó kvalitásaik, olyan feladatok elvégzésre is alkalmassá teszik ezeket a rendszereket, amelyek hagyományos értelemben véve nem algoritmizálhatóak.

E feladatok egy részénél - ilyenek pl. komplex ipari, gazdasági vagy pénzügyi folyamatok időbeli viselkedésének előrejelzése - a megoldás nehézsége általában abból ered, hogy nem rendelkezünk azzal a tudással, amely az algoritmikus megoldáshoz szükséges lenne. Alapvetően kétféle tudásra lenne szükségünk: egyrészt a folyamatok mögött meghúzódó fizikai, közgazdasági, stb. törvényszerűségeket kellene ismernünk, másrészt a törvényszerűségek ismeretében is csak akkor tudnánk a feladatot megoldani, ha a folyamatokat

létrehozó rendszerek pillanatnyi állapotát és e rendszerekre ható összes vagy legalább minden lényeges hatást, a környezeti feltételeket is ismernénk. A feladatokról ugyanakkor más formában - adatokban megtestesülve - rendelkezésünkre áll tudás, amelyet ha fel tudunk használni, a feladat valamilyen megoldásához eljuthatunk. Egy adott problémáról mindig véges számú adatunk lehet, továbbá az adatok által hordozott tudás sohasem teljes. Mégis a természetes neurális hálók képesek az adatokból nyert ismeretek általánosítására, olyan szituációkban is jó választ adnak, melyek az adatok között nem találhatók meg. Egyes mesterséges neurális hálók is rendelkeznek ezzel a képességgel. Ez azt jelenti, hogy hiányos, esetleg pontatlan ismereteket hordozó, legtöbbször zajos adatokból is kinyerhető általános tudás.[2]

A logisztikával és az ökonometriával foglalkozó szakemberek egyik legnagyobb kihívása a gazdasági és keresleti igényfolyamatok jövőbeli alakulásának megjóslása. Az előbb említett jelenségek leírása idősorokkal történik. Az idősorok jövőbeli alakulását alapvetően komplex statisztikai eszközrendszer alkalmazásával lehet megmondani. Az első előrejelzési modellek a 40-es és 50-es években születtek meg. A legnagyobb áttörést a kereslettervezés területén Box és Jenkins érte el az 1970-es években az ARIMA és SARIMA modellek megalkotásával, amelyek mind a mai napig bezáróan a leggyakrabban alkalmazott előrejelzési eljárásoknak számítanak. Ezeknek a modelleknek a megbízhatóságát és hatékonyságát számos kutató igazolta mind a rövid, közép és hosszú távú forecast-ok esetében. Azonban a mesterséges intelligenciák megjelenésével egy új lehetőség nyílik az idősorok jövőbeli alakulásának prognózisára.[3]

A fent említett komplex matematikai modellekre nyújthat egy igen vonzó alternatívát a neurális hálókkal történő előrejelzés. A neurális hálózatok tudománya egy fejlődésben lévő tudomány. Az újabban megélénkült kutatások eredményeképpen most már nemcsak azt sikerült bemutatni, hogy a neurális hálózatok komplex feladatok megoldásában igen hasznosnak bizonyulnak, hanem az ehhez szükséges elméleti alapokat - vagy legalábbis számos elméleti alaperedményt - is sikerült kidolgozni. Bizonyított tény pl., hogy megfelelő felépítésű neurális hálózatok képesek csaknem tetszőleges nemlineáris leképezések tetszőleges pontosságú approximációjára, vagyis, hogy a neurális hálózat univerzális approximátornak tekinthető. E képességük révén a hálózatok alkalmasak nemlineáris (statikus vagy dinamikus) rendszerek modellezésére. Ugyancsak bizonyítható, hogy megfelelő tanulási algoritmussal, adott felépítésű hálózat alkalmassá tehető adatokban meglévő hasonlóságok felismerésére, az adatokban rejtve meglévő ismeretek kinyerésére, tehát elméletileg előrejelzési modellek is készíthetőek velük.[2]

Ennek a dolgozatnak a célja, hogy feltárja mélységében neurális hálók működésének hátterét, hogy megismerje a különböző struktúrával rendelkező mesterséges intelligenciák tulajdonságait és az ezekhez köthető algoritmusok koncepcióit. A részletes háttérfeltárást követően további cél neurális hálók előrejelzési alkalmazhatóságának vizsgálata, valamint a paraméterek modellre gyakorolt hatásainak kísérleti úton történő feltárása. A tesztek lefolytatását és elemzését követően cél a mesterséges intelligencián alapuló modellek hatékonyságának összevetése az általánosan használt ökonometriai alapokon nyugvó előrejelzési eszközökkel való összehasonlítás és az összehasonlítás eredményeinek elemzése.

1. A neurális hálók áttekintése

Az ember a múlt század közepén felfedezte a természet zsenialitását és megpróbálta azt legjobb tehetsége szerint lemásolni. A mesterséges intelligenciák területén tett felfedezéseknek köszönhetően ma már a masszív adathalmazokban rejlő szabályosságokat nem feltétlenül humán erőforrásoknak kell komplex statisztikai eszközrendszer segítségével felderíteni, hanem ezeket a feladatokat mesterséges intelligenciák is el tudják végezni.

Neurális hálózatnak nevezzük azt a hardver vagy szoftver megvalósítású párhuzamos, elosztott működésre képes információ feldolgozó eszközt, amely:

- azonos, vagy hasonló típusú - általában nagyszámú - lokális feldolgozást végző műveleti elem, neuron (*processing element, neuron*) többnyire rendezett topológiájú, nagymértékben összekapcsolt rendszeréből áll,
- rendelkezik tanulási algoritmussal (*learning algorithm*), mely általában minta alapján való tanulást jelent, és amely az információfeldolgozás módját határozza meg,
- rendelkezik a megtanult információ felhasználását lehetővé tevő információ előhívási, vagy röviden előhívási algoritmussal (*recall algorithm*).[2]

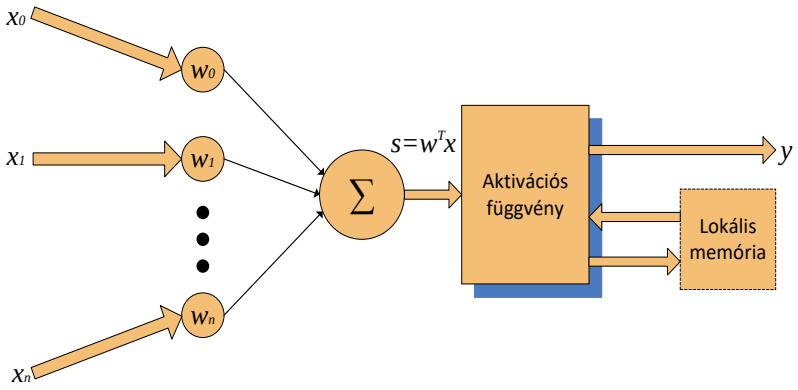
A neurális hálózatok működésénél tipikusan két fázist különböztethetünk meg. Az első fázis, melyet tanulási fázisnak nevezünk, a hálózat kialakítására szolgál, melynek során a hálózatba valamilyen módon beépítjük, eltároljuk a rendelkezésre álló mintákban rejtve meglévő információt. Eredményként egy információ-feldolgozó rendszert kapunk, melynek használatára általában a második fázisban, az előhívási fázisban kerül sor. A két fázis a legtöbb esetben időben szétválik. A tanulási fázis rendszerint lassú, hosszú iterációkat, tranzienseket, esetleg sikertelen tanulási szakaszokat is hordoz. Ezzel szemben az előhívási fázis tipikusan gyors feldolgozást jelent. Azonban ahhoz, hogy tüzetes képet kapjunk a neurális hálózatok felépítéséről, működéséről, valamint azok kereslet előrejelzés terén való alkalmazhatóságáról alaposan fel kell tárni a témához köthető irodalmat.

1.1. A perceptron modellje

Egy neuron vagy műveleti elem, processzáló elem (processing element) egy több-bemenetű, egykimenetű eszköz, amely a bemenetek és a kimenet között általában valamilyen nemlineáris leképezést valósít meg. Egy neuron rendelkezhet lokális memóriával is, amelyben akár bemeneti, akár kimeneti értékeket vagy a működés előéletére vonatkozó állapotinformációt tárolhat. A bemeneti vagy a bemeneti- és a tárolt értékekből az aktuális kimeneti értéket egy tipikusan nemlineáris függvény alkalmazásával hozza létre, melyet aktiváló vagy aktivációs függvénynek nevezünk. A „neuroforecasting” szempontjából ez azért fontos, mivel a bonyolultabb rendszerek alapját is a perceptron modellje alkotja.[2]

A műveleti elemek legegyszerűbb és egyben legelterjedtebb változata az egyenrangú bemenetekkel rendelkező memória nélküli neuron, melynek tipikus felépítése hasonló az első ábrán láthatóéhoz, azonban blokkvázlata nem tartalmazza a szaggatott vonallal jelölt lokális memória elemet A bemutatott neuron esetén az x_i skalár bemenetek w_i ($i=0,1,...,N$) súlyozással kerülnek összegzésre, majd a súlyozott összeg egy f nemlineáris elemre kerül. Szokás az első

ábrán látható neuronok esetén a bemeneti jelek súlyozott összegét ingernek (*excitation*), míg a kimeneti jelet válasznak (*activation*) nevezni. Ezek az elnevezések az egyes területeken ma is erősen hivatkozott biológiai analógiára utalnak. Egyes neuronhálóknál a nemlineáris transzfer függvény elmarad, így lineáris neuronokról is beszélhetünk. Az ilyen lineáris súlyozott összegzőt megvalósító neuront önmagában viszonylag ritkán használják, legtöbbször egy nagyobb hálózat kimeneteinek előállítására szolgál. Vannak azonban olyan nemellenőrzött tanítású hálók, amelyek lineáris neuronokból épülnek fel. Ezeknél a hálóknál a speciális tulajdonságot a súlyok kialakítása, a speciális tanulási eljárás biztosítja.[2]



1. ábra *A perceptron elvi felépítése*

A w_i a háló tanításának eredményeképpen kapjuk meg. A hálózat összegző pontján a bemenetek lineáris kombinációját kapjuk meg az alábbi formula szerint:

$$s = \sum_{i=0}^{N} w_i x_i = w^T x$$

Ahol:

- s: az aktivációs függvény bemenete, más néven az inger,
- w_i: az i-edik bemenet súlya,
- x_i: az i-edik bemenet értéke.

Az első ábrán bemutatott, összegzővel sorba kapcsolt aktivációs függvényt használó egyszerű felépítésű neuront, lokális memória nélkül perceptron néven Frank Rosenblatt, illetve adeline néven Bemard Widrow javasolta. A két neuron-modell felépítése megegyezik, eltérés csupán az alkalmazott tanítási eljárásokban van. A hatékonyabb tanítás miatt az adeline nagyobb népszerűségre tett szert, ezért sok szakirodalomban és cikkben használatos az adeline elnevezés. A perceptron, alapvető strukturális tulajdonságainak köszönhetően lehetővé teszi, hogy építőköveként szolgáljon a komplexebb modelleknek, akárcsak a természetben, az elemi idegsejtek kapcsolódásához hasonlóan, komplex virtuális hálókat lehet felépíteni. Az ilyen módon felépülő modellek működését és alkalmazhatósági területét nagymértékben befolyásolja az alkalmazott aktivációs függvény.[4]

1.3. Az aktivációs függvények

A neurális hálókban leggyakrabban használt aktivációs függvények a szigmoid jellegűek, amely alatt sokszor a logisztikus függvényt érti a szakirodalom. A logisztikus függvény mellett még számos más nemlinearitást szoktak alkalmazni, esetenként akár többfélét is egy adott feladat megvalósításában. Gyakorlatilag a modellezés során szinte bármilyen aktivációs függvényt használhatnánk feltéve, hogy differenciálható, amely a tanítási algoritmusok sajátossága miatt szükséges. Sokszor javasolt a lineáris aktivációs függvény alkalmazása, azonban ez önmagában sokszor nem elég a komplexebb feladatok megoldásánál a fennálló nem lineáris összefüggések megléte miatt, ezért csak a kimenő, vagy a bemenő oldalon szoktak lineáris transzfer függvényt használni. A téma áttekintése azért fontos, hogy tudjuk milyen függvénnyel érdemes próbálkozni a kereslettervezésre használható mesterséges intelligenciák esetében.[1]

A logisztikus aktivációs függvény sok szakértő véleménye szerint áll a legközelebb ahhoz, hogy az agyi neuronok működését másolja. A függvény karakterisztikája, gyakorlatilag lehetővé teszi annak a folyamatnak az imitálását, amikor egy neuron működésbe lép. A bemenet meghatározza, hogy működésbe lép-e az adott elem, illetve működés esetén a válaszjel erősségét is meghatározza. Azonban a logisztikus függvény alkalmazásának is vannak korlátai. Problémát okozhat, ha bemeneti értékek között nagy mennyiségben fordulnak elő negatív értékek, mivel ekkor a kimeneti értékek rendkívül közel lennének a nullához, ami azt eredményezné, hogy a háló tanítási folyamata rendkívül lassú lenne, és ezzel párhuzamosan nőne az esélye annak, hogy a háló súlyai bent ragadnak egy lokális minimumban. A logisztikus függvény képletét az alábbi egyenlet mutatja be:[1]

$$\varphi(s) = \frac{1}{1 + e^{-s}}$$

Ahol:

- s: az aktivációs függvény bemenete, más néven az inger,
- $\varphi(s)$: a perceptron ingerre adott válasza.

Ahogyan korábban is említésre került a neurális hálók egyik legelső alkalmazási területe a lineáris szeparáció volt. Ha ennek kontextusában szeretnénk értelmezni a logisztikus függvényt, mint aktivációs függvényt, az eredmény gyakorlatilag annak a valószínűségét adja meg, hogy egy adott bemenet mekkora valószínűséggel tartozik egy adott csoportba, mivel a függvény értékkészlet {0,1} intervallumba esik. Más szemszögből vizsgálva az eredményeket úgy is lehet értelmezni, mint hogy az adott bemenet mekkora súllyal van ráhatással a prediktált értékre. A második ábrán a logisztikus függvény is látható. Az ábrán láthatjuk a függvény tipikus „S" jellegű karakterisztikáját. A függvény, ha végtelenbe tartunk s-sel (s→∞), mivel e^{-s} értékei egyre nagyobb számok esetén egyre kisebb értékeket vesznek fel, a végtelenben megközelíti az 1-et. A függvény dinamikája pontosan fordítottan működik abban az esetben, ha egyre kisebb értékeket helyettesítünk be és -∞ esetén eléri a 0 értéket. Az, hogy végső soron a bemeneti jelre milyen választ ad a rendszer, az inputok és az ezekhez tartozó súlyok lineáris kombinációjaként előálló inger határozza meg. [1]

Bináris kimenet esetén a megjósolt valószínűséget az alábbiak szerint lehet a kívánt formára áttranszformálni:

$$\hat{y} = \begin{cases} 1, & ha: s \geq 0.5 \\ 0, & egyébként \end{cases}$$

Annak érdekében, hogy a logisztikus függvény esetében említett lokális minimumba ragadás veszélyét minimalizálni lehessen, a szakirodalom a hiperbolikus tangens függvényt javasolja, főleg a többrétegű teljesen összekapcsolt hálók esetében, a rejtett rétegek neuronjainak esetében. A hiperbolikus tangens függvény egyenletét az alábbi egyenlet mutatja be: [1]

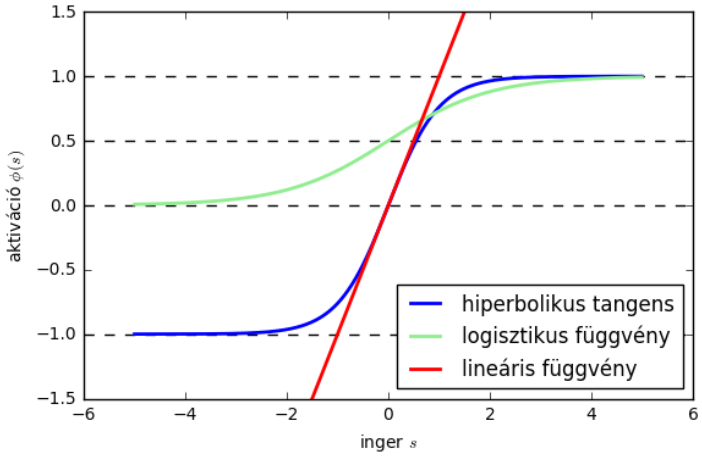
$$\varphi_{tanh}(s) = \frac{e^s - e^{-s}}{e^s + e^{-s}}$$

Ahol:

- s: az aktivációs függvény bemenete, más néven az inger,
- $\varphi_{tanh}(s)$: a perceptron ingerre adott válasza hiperbolikus tangens aktivációs függvény esetében.

A hiperbolikus tangens rendkívül nagy előnye a logisztikus függvénnyel szemben, hogy értékkészlete sokkal szélesebb spektrummal bír. A függvény értékei a (-1,1) nyitott intervallumban bármilyen értéket felvehetnek. Egy mintafelismerésről szóló oxfordi tanulmány szerint, ennek következtében a tanítási folyamat konvergenciája nagymértékben javítható. A hiperbolikus tangens függvényét az alábbi második ábra mutatja be. A diagramon jól látható, hogy a függvény által felvett értékek a nulla köré centralizálódnak. Összehasonlítva a logisztikus függvénnyel, egy sokkal meredekebb függvénnyel van dolgunk a hiperbolikus tangens esetében. A szakirodalom alapvetően csak komplex hálók esetében javasolja az implementálást. [1] Mivel a kereslet előrejelzési problémák esetében is hasonlóan komplex problémákkal van dolgunk, így a hiperbolikus tangens függvényt az általunk vizsgált architektúrák esetében is alkalmaztuk. Tapasztalataink a mi esetünkben is igazolták, hogy a hiperbolikus tangens a kereslet előrejelzési problémáknál is hatékonyan alkalmazható.

Az aktivációs függvények közül a legegyszerűbb a lineáris függvény. Bár önmagában komplex problémák megoldására alkalmatlan, több rétegű hálók esetében, ahol a köztes rétegekben szigmoid jellegű aktivációs függvények kerülnek alkalmazásra, ott a lineáris aktivációs függvény rendkívül hasznosnak bizonyul. Az előbbi kombináció teljesülése esetén egy rendkívül erős regressziós modell hozható létre. A lineáris aktivációs függvényt az alábbi második ábra mutatja be.[1]



2. ábra *Aktivációs függvények*

1.4. A neurális hálók architektúrája

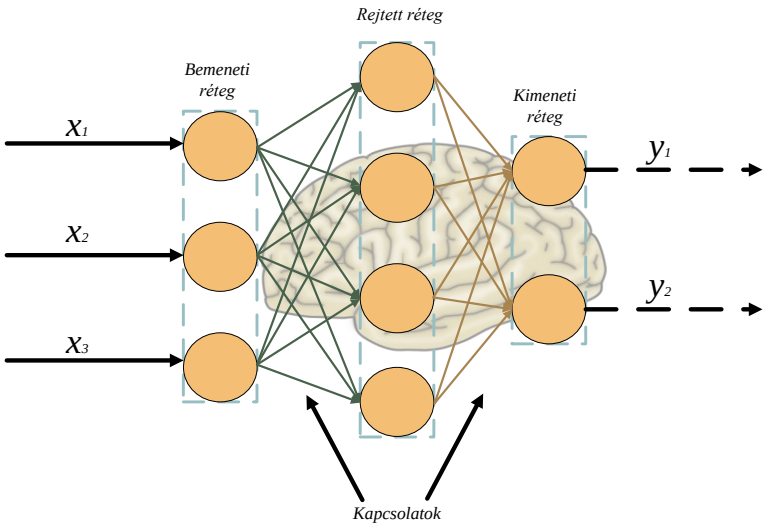
A neurális hálózat topológiáján a neuronok összeköttetési rendszerét és a hálózat bemeneteinek és kimeneteinek helyét értjük. A hálózat topológiáját általában irányított gráffal reprezentáljuk. A gráf csomópontjai a neuronoknak felelnek meg, míg a kapcsolatokat a kimenetek és bemenetek között a gráf élei reprezentálják, a neuron-bemenettől a kimenet felé irányítva. (Az élekhez értelemszerűen hozzárendelhetjük a megfelelő w_{ij} bemeneti súlytényezőket.[2]

A gráf egyes csomópontjai rendszerint nincsenek az összes többi csomóponttal kapcsolatban, csupán a csomópontok egy részhalmazával. Ez legtöbbször lehetőséget nyújt arra, hogy a gráf csomópontjainak - a neuronoknak - halmazát diszjunkt részhalmazokra bontsuk. Ennek megfelelően háromféle neuront különböztethetünk meg:

- bemeneti neuronok, melyek típusukban is különböznek a többi neurontól (egybemenetű, egykimenetű, buffer jellegű neuronok, melyeknek jelfeldolgozó, processzáló feladatuk nincs), bemenetük a hálózat bemenete, kimenetük más neuronok meghajtására szolgál,
- kimeneti neuronok, melyek kimenete a környezet felé továbbítja a kívánt információt, típusukra nézve nem feltétlenül különböznek a többi neurontól,
- rejtett neuronok (*hidden neurons*), melyek mind bemeneteikkel, mind kimenetükkel kizárólag más neuronokhoz kapcsolódnak.[2]

A neuronokat sok esetben rétegekbe (*layers*) szervezzük, ahol egy rétegbe hasonló típusú neuronok tartoznak. Az egy rétegbe tartozó neuronokra még az is jellemző, hogy kapcsolataik is hasonlók. Az azonos rétegbe tartozó neuronok mindegyikének bemenetei a teljes hálózat bemenetei, vagy egy másik réteg neuronjainak kimeneteihez kapcsolódnak, kimenetei pedig ugyancsak egy másik réteg neuronjainak bemeneteit képezik, vagy a teljes hálózat kimeneteit alkotják. Ennek megfelelően beszélhetünk bemeneti rétegről (*input layer*), rejtett rétegekről (*hidden layers*) és kimeneti rétegről (*output layer*). A bemeneti réteg, amely buffer jellegű neuronokból épül fel, információfeldolgozást nem végez, feladata csupán a háló bemeneteinek a következő réteg bemeneteihez való eljuttatása. Ennek megfelelően egy rétegekbe szervezett háló legalább két réteggel, egy bemeneti és egy kimeneti réteggel kell hogy rendelkezzen. E két réteg között elvben tetszőleges számú rejtett réteg helyezkedhet el. A szakirodalomban a

rétegszám definíciója sokszor nem egyértelmű. Egyes szerzők a rendszerint lineáris, bemeneti buffer réteget is beleszámítják a rétegszámba, mások csak a rejtett- és kimeneti rétegeket, tehát csak azokat a rétegeket, melyek információfeldolgozást is végeznek. Így az egyértelműség kedvéért az a szokás terjedt el, hogy az aktív, jelfeldolgozást is végző (processzáló) rétegek számát adják meg a hálózat jellemzőjeként A neuronhálókat az egyes neuronok közötti összeköttetési rendszer alapján két fő csoportba sorolhatjuk: beszélhetünk előrecsatolt hálózatokról (*feedforward networks*) és visszacsatolt hálózatokról (*recurrent networks*). Visszacsatoltnak nevezünk egy neurális hálót, ha a topológiáját reprezentáló irányított gráf tartalmaz hurkot, egyébként a neurális háló előrecsatolt. A következő harmadik ábra egy egyszerű egy rejtett réteget tartalmazó előrecsatolt neuron hálót mutat be.[2]



3. ábra *Egy rejtett réteget tartalmazó neurális háló*

A visszacsatolt hálózatoknál beszélhetünk globális és lokális visszacsatolásról. Globális visszacsatolásnál a hálózat kimenetét csatoljuk vissza a bemenetére. A lokális visszacsatolásokat három csoportba sorolhatjuk:

- elemi visszacsatolásról (*recurrent connections*) beszélünk, ha egy réteg egy neuronjának kimenete közvetlenül egyik saját bemenetére van visszacsatolva,
- laterális (*lateral connections, intra-layer connections*) visszacsatolásoknál, valamely rétegek neuronjainak kimenetei ugyanazon réteg neuronjainak bemeneteire kapcsolódnak, de nem értjük ide a neuron önmagára való visszacsatolását,
- a rétegek közötti visszacsatolások (*inter-layer connections*) több réteget tartalmazó hurkot hoznak létre a gráfon.[2]

A neuronokat sok esetben rétegekbe (*layers*) szervezzük, ahol egy rétegbe hasonló típusú neuronok tartoznak. Az egy rétegbe tartozó neuronokra még az is jellemző, hogy kapcsolataik is hasonlók. Az azonos rétegbe tartozó neuronok mindegyikének bemenetei a teljes hálózat bemenetei, vagy egy másik réteg neuronjainak kimeneteihez kapcsolódnak, kimenetei pedig ugyancsak egy másik réteg neuronjainak bemeneteit képezik, vagy a teljes hálózat kimeneteit alkotják. Ennek megfelelően beszélhetünk bemeneti rétegről (*input layer*), rejtett rétegekről (*hidden layerek*) és kimeneti rétegről (*output layer*). A bemeneti réteg, amely buffer jellegű

neuronokból épül fel, információfeldolgozást nem végez, feladata csupán a háló bemeneteinek a következő réteg bemeneteihez való eljuttatása. Ennek megfelelően egy rétegekbe szervezett háló legalább két réteggel, egy bemeneti és egy kimeneti réteggel kell rendelkezzen. E két réteg között elvben tetszőleges számú rejtett réteg helyezkedhet el. A szakirodalomban a rétegszám definíciója sokszor nem egyértelmű.

1.5. A neurális hálók approximációs képessége

A feladatok jelentős része viszont valamilyen dinamikát mutat: vagy maga a vizsgált folyamat, amelyet optimalizálunk, vagy melynek az eredményeit osztályozzuk, stb. esetleg a környezet vagy zaj változik az időben, vagy a folyamat pillanatnyi értéke nem csupán az adott bemeneti jeltől, hanem az előzményektől is függ. Ilyen feladattal találkozunk akkor is, ha egy időben változó értéksorozat valahány elmúlt értékének, egy idővariáns folyamat régebbi mintavételi értékeinek ismeretében a következő értéket vagy értékeket kell előre jelezni, jósolni, a későbbi mintavételi értékekre becslést adni. Ezeket a feladatokat idősor előrejelzési feladatoknak (*time series prediction*) szokás nevezni, és a legkülönbözőbb alkalmazási területeken, pl. komplex ipari rendszerek működésének, gazdasági és pénzügyi folyamatok viselkedésének, természeti jelenségek lefolyásának, időbeli alakulásának, stb. előrejelzésénél jelennek meg.[2]

Természetesen ilyen esetekben a feldolgozó algoritmusnak vagy modellező eljárásnak is a megfelelő időfüggést kell mutatnia. Ezt neurális hálók alkalmazása esetén legtöbbször úgy érjük el, hogy a statikus feladatok megoldására kialakított (rendszerint nemlineáris) neurális hálót kiegészítjük dinamikus (rendszerint lineáris) komponensekkel. Több szinten történhet a dinamikus elemek felhasználása, pl. maga a neuron is dinamikussá tehető, szűrők alkalmazásával. Dinamikus feladatok megoldásánál ugyanakkor gyakrabban alkalmaznak olyan megoldást, amikor a tisztán statikus hálót kiegészítik dinamikus komponensekkel.

A neurális hálók alapjait az alábbi matematikai tételek képezik:

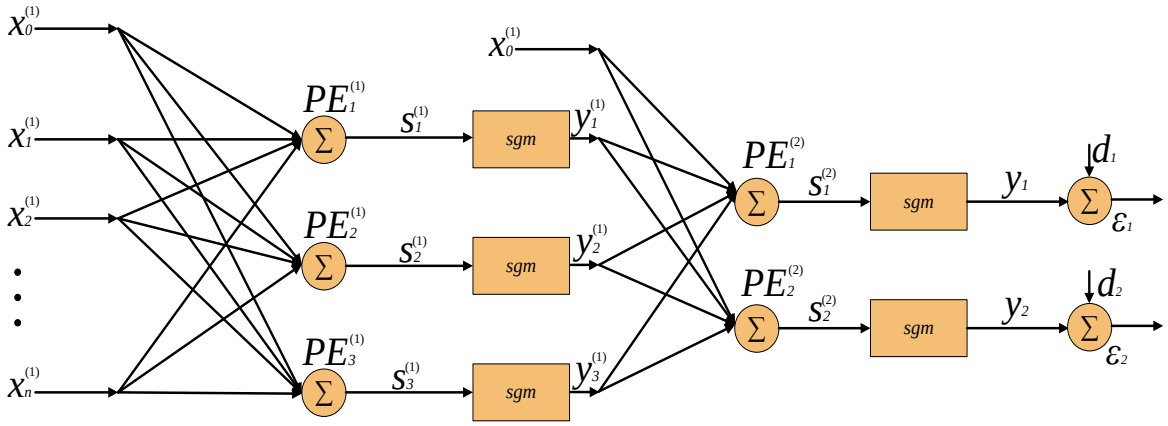
- Kolmogorov tétel,
- Sprecher tétel,
- Hecht tétel,
- Funahashi tétel.

Ezek a tételek kimondják, hogy a komplex rendszerek dekomponálhatók egyváltozós függvényekké és kimondják, hogy egy rejtett réteg is elegendő ahhoz, hogy egy folytonos függvény tetszőleges pontossággal közelíteni tudjunk, illetve, hogy a megfelelő approximációs képességet nemlineáris aktivációs függvénnyel érhetjük el.[2]

1.6. A több rétegű perceptron

A többrétegű perceptron, vagy más néven multi-layer perceptron, röviden pedig MLP a gyakorlati feladatok megoldásánál talán a leggyakrabban alkalmazott struktúra. Az MLP rétegekbe szervezett neuronokból áll ahol annak biztosítására, hogy a hálózat kimenete a súlyok folytonos, differenciálható függvény legyen, a neuronok differenciálható kimeneti aktivációs függvénnyel rendelkeznek. A többrétegű hálózatok felépítése a következő negyedik ábrán követhető. Az ábra egy két aktív réteget tartalmazó hálózatot mutat, amelyben az első aktív

rétegben, a rejtett rétegben három, a második rétegben, jelen esetben a kimeneti rétegben két processzáló elem található. A hálózat tehát egy többrétegű előrecsatolt hálózat. A fentebb említett matematikai tételek értelmében tudjuk, hogy a neuronhálók, a súlyvektorok megfelelő kialakítását követően, bármilyen folytonos nemlineáris függvény tetszőleges pontosságú approximációjára képesek.[2]



4. ábra *Egy komplex MLP modellje*

A háló bemutatásához vezessük be a következő jelöléseket: egy *L* információ-feldolgozást végző rétegből álló MLP - nél az egyes rétegeket felső indexszel ($l = 1, 2, \ldots, L$) különböztetjük meg. A rétegen belüli processzáló elemekhez (*PE*) az *i* indexet rendeljük, míg *j* a PE bemeneteit megkülönböztető index jelölésére szolgál, ez lesz egyben a megfelelő súlyvektor komponenseinek indexe is. (Pl. PE_i^l' az *l*-edik réteg *i*-edik processzáló elemét, neuronját jelöli, w_i^l ennek a neuronnak a súlyvektorát, míg w_{ij}^l ugyanezen súlyvektornak az j - edik komponensét jelöli, ennek megfelelően w^l az *l*-edik réteg összes neuronja szerint a súlyokat reprezentáló mátrix.) A neuronok bemeneteire kerülő jeleket *x*-szel, a kimeneti jeleket *y*-nal jelöljük, szintén alkalmazva a réteg és a rétegen belüli indexeket. Így pl.: x_{ij}^l az *l*-edik neuronréteg *i*-edik processzáló elemének *j*-edik bemenetére kerülő jelet adja meg. A kimeneteknél megkülönböztetjük a súlyozott összegző kimenetét (lineáris kimenet) a neuron nemlineáris kimenetétől. A lineáris kimenetet az eddigi jelöléseknél megfelelően *s*-sel jelöljük. Egy MLP tetszőleges számú rejtett réteget használhat, és mint ahogy korábban arról már volt szó ahhoz, hogy univerzális approximátor képességgel rendelkezzen, legalább egy szigmoid aktivációs függvénnyel rendelkező rejtett réteget kell tartalmaznia.[2]

1.7. A neurális hálók tanítása

A tanulási képesség lehetővé teheti azt is, hogy egy eddig megfelelő viselkedésű rendszer a változó körülményekhez való alkalmazkodás céljából módosítsa a viselkedését, adaptálódjon. Az adaptív, tanuló rendszerek alapvető jellemzője tehát, hogy nem rögzített képességekkel rendelkeznek, amelyek egy adott feladat ellátására teszik őket alkalmassá, hanem képességeiket fejleszteni tudják, továbbá alkalmazkodni tudnak a változó körülményekhez, környezethez.[2]

Neurális hálózatokban a tanulás alábbi főbb formáival találkozhatunk:

- tanulás tanítóval, ellenőrzött, vagy felügyelt tanulás,
- tanulás tanító nélkül, nemellenőrzött vagy felügyelet nélküli tanulás,

Ellenőrzött tanulásnál (*supervised learning*) a hálózat összetartozó be- és kimeneti értékei, tanító mintapont párok állnak rendelkezésre. A tanítás ezeken az ismert összerendelt mintapárokon alapul. A háló feladata, hogy megtanulja a mintapont párok által reprezentált bemenet-kimenet leképezést. Mivel ismertek a kívánt válaszok (a bemenetekhez tartozó kimenetek), a hálózat tényleges válasza minden esetben közvetlenül összehasonlítható a kívánt válasszal. Az összehasonlítás eredménye - a tényleges és a kívánt válasz különbsége - felhasználható a hálózat olyan módosítására, hogy a tényleges válaszok a kívánt válaszokkal minél inkább megegyezzenek és a hálózat tényleges viselkedése és a kívánt viselkedés közötti eltérés csökkenjen. Az ellenőrzött tanulás általában iteratív eljárással, a környezetből származó ismeret fokozatos felhasználásával, a tanító mintapontok egyenkénti, esetenként többszöri, ismételt felhasználásával történik. Amennyiben a mesterséges intelligenciákat akár közvetlenül az előrejelzés elvégzésére, vagy akár a folyamat támogatására szeretnénk felhasználni úgy mindenképpen, kell ezzel a tanítási módszerrel foglalkozni.[2]

Nemellenőrzött tanulásnál (*unsupervised learning*) nem állnak rendelkezésünkre adott bemenetekhez tartozó kívánt válaszok. A hálózatnak a bemenetek és a kimenetek alapján kell valamilyen viselkedést kialakítania; a környezetből azonban nincs semmiféle visszajelzés, ami a hálózat viselkedésének helyességére utalna. A nemellenőrzött tanulású hálózatoknak azt kell felderíteniük, hogy van-e a hálózat bemenetére kerülő adatokban, jelekben valami hasonlóság, van-e az adatok között korreláció, kialakíthatók-e a bemeneti adatok között kategóriák, csoportok. Itt tehát a háló által megvalósítandó leképezés pontosan nem definiálható. Ennek ellenére a hálózat képes önmaga módosítására. Ez a kutatás szempontjából azért releváns, mivel komplex több dimenziós bemenet esetén fel lehet használni ezt a módszert főkomponens analízisre, ahogy azt majd láthatjuk a későbbiekben.[2]

A modell-illesztési feladat általánosan, tehát a kereslettervezés esetében is, a következő formában fogalmazható meg: Adott egy rendszer, amely bemenetei és kimenetei között valamilyen *g(.)* leképezést valósít meg. A lékepezés nem ismert, viszont rendelkezésre áll egy $\{z_i\}_{j=1}^{l} = \{x_i, d_i\}_{j=1}^{l}$ mintapont készlet, amelynél az összetartozó bemeneti és kimeneti értékek közötti kapcsolatot az ismeretlen leképezés adja meg. A mintakészlet elemei a mintapontok általában többdimenziós vektorok, melyek a mintatérben helyezkednek el.

A tanulási feladat az, hogy egy *f(x,w)* paraméteres függvényhalmazból, ahol w∈W a függvenyhalmaz paramétere és W a paramétertér, válasszunk egy függvényt olyan módon, hogy adott X_i bemenetre a kiválasztott függvény $y_i = f(x_i, w)$ válasza valamilyen értelemben a lehető legjobban közelítse az ismeretlen leképezés azonos bemenetre adott d_i válaszát. A d_i válaszokat kívánt válaszoknak (*desired response*) nevezzük. A függvényhalmaz szabad paramétereit tartalmazó w vektort a W paramétertér megfelelő pontjaként a rendelkezésre álló mintapontok alapján kell megválasztani vagy meghatározni. Az $\{x_i, d_i\}_{j=1}^{l}$ mintapont készletét, tanító készletnek, vagy tanító mintakészletnek (*training set*) nevezzük, azt az eljárást pedig, melynek során a tanító mintakészlet felhasználásával az *f(x,w)* leképezést kialakítjuk,

tanulásnak. A tanuló eljárás célja tehát egy valamilyen értelemben optimális *w** paraméter szett meghatározása a tanító mintakészlet alapján.[2]

A legjobb approximációt biztosító függvény meghatározásához az approximáció minőségét jellemző mértékre van szükség. Ez a mérték egy hibafüggvény (*error function*) felhasználásával határozható meg, amit költségfüggvénynek (*cost function*) vagy veszteségfüggvénynek (*loss function*) is szokás nevezni. A veszteségfüggvény, ami a továbbiakban *J(d,f(x,w))*-vel lesz jelölni mind a háló válaszának, *f(x,w)* -nek, mind a rendelkezésre álló *d(x)* kívánt válasznak a függvénye: leggyakrabban az approximáció $\varepsilon(x,w)$ hibájának a függvénye, ahol a hibát a kívánt válasz és a tényleges válasz különbségeként definiálhatjuk:[2]

$$\varepsilon(x,w) = d(x) - f(x,w)$$

A tanuló rendszer konstrukciója véges számú tanító mintapont alapján történik. A tanuló eljárás célja azonban általában nem az, hogy a véges számú tanítópontban kapjunk megfelelő válaszokat, hanem egy olyan leképezés megtanulása, amely a tanítópontok által reprezentált bemenet-kimenet kapcsolatot is megadja. A tanuló rendszer tehát általánosítóképességgel (*generalization capability*) kell, hogy rendelkezzen.[2]

A tanuló rendszer minősítésére olyan mértékre van szükségünk, amely nem csupán azt mondja meg, hogy a tanuló rendszer a tanító mintapontokat milyen pontosan tanulta meg, hanem a rendszer általánosítóképességéről is mond valamit. A veszteségfüggvény adott bemenet mellett minősíti a tanuló rendszert. A minősítésre azonban alkalmasabb mérték, ha a veszteség összes pontra vonatkozó átlagos értékét, a veszteség várható értékét, az úgynevezett. kockázatot (*risk*) határozzuk meg, ahol a kockázat a veszteségfüggvényhez rendelt számérték.[2]

$$R(w) = E_{x,d}\{J(d, f(x,w))\}$$

A kockázat meghatározásánál a várható értéket a minták *p(x,d)* együttes sűrűségfüggvénye szerint kell vennünk. A kockázat a paramétervektor függvénye, amit szokás kritériumfüggvénynek (*criterion function*) is nevezni és *C(w)*-vel jelölni. A regressziós feladatoknál az egyik leggyakoribb veszteségfüggvény a hiba négyzetes függvénye, amit következőképpen lehet matematikai formába önteni:[2]

$$J\big(d, f(x,w)\big) = \big(d(x) - f(x,w)\big)^2$$

Más veszteségfüggvények alkalmazása is szokásos. Ilyen függvény lehet pl. az abszolútérték függvény vagy ennek kissé módosított változata az ún. ε érzéketlenségi sávval rendelkező abszolútérték függvény. Egyes speciális alkalmazásokban szerepet kaphatnak a különböző abszolútérték-hatvány függvények, és ezek lineáris kombinációi. Az átlagos négyzetes hiba vizsgálata több hasznos következtetésre ad lehetőséget. Feltételezve, hogy megfigyelési zaj nulla várható értékű, továbbá, hogy a zaj és a *g(x)* leképezés korrelálatlan és hogy *d(x)-g(x)=n*, vagyis kimeneti megfigyelési zaj [2]

Matematikailag bebizonyítható, hogy kockázat minimumát biztosító w* paramétervektor egyben a tanuló rendszer és a regressziós függvény válaszainak átlagos négyzetes eltérését is minimalizálja. A feltételezett zajmodell mellett tehát a zajos megfigyelések ellenére a kockázatminimalizálás a regressziós függvény átlagos négyzetes értelemben vett legjobb

közelítését eredményezi. Másfelől, ha a tanulás célját úgy fogalmazzuk meg, hogy azt az *f(x,w)* -t keressük, amely adott *x* mellett a lehető legjobban közelíti a megfelelő *d* kívánt választ, akkor a megoldást az *MSE* minimalizálása biztosítja.[2]

Az, hogy a neurális hálók esetében a legideálisabb veszteségfüggvénynek az *MSE* mondható egy különösen szerencsés véletlen, hiszen a kereslettervezésben a legfontosabb mutatók közé sorolandó a szóban forgó indikátor. Az *MSE* felbontható torzítás négyzetére és a variancia összegére. A torzítás (*bias*) azt mutatja meg, hogy a tanuló rendszer leképezésének várható értéke mennyiben tér el a regressziós függvénytől, míg a variancia a leképezésnek a saját várható értékétől való átlagos négyzetes eltérését adja meg. A torzítás arra ad választ, hogy az adott függvényosztály, amit a háló típusa és architektúrája rögzít mennyire alkalmas a kérdéses approximációs feladat megoldására, a variancia pedig a véges számú tanítópont következménye: azt adja meg, hogy a véges számú tanítópont felhasználásával konstruált háló válasza, *f(x,w)* mennyire érzékeny arra, hogy egy adott problémából származó aktuális tanítóhalmaz épp milyen mintapontokból áll. Azonos négyzetes hiba előállhat nagyobb torzítás és kisebb variancia vagy kisebb torzítás és nagyobb variancia mellett; a két eset eltérő jellegű approximációt jelent. A szabad paraméterek számának növelése és így a háló által megvalósított függvényosztály komplexitásának növelése a torzítás csökkentését eredményezi. Minél több a szabad paraméter, annál komplexebb leképezés megvalósítására lehet alkalmas a neuronháló. Elegendően nagyszámú szabad paraméter mellett a háló válasza a tanítópontokban akár tetszőlegesen kis hibájú is lehet, miközben a tanítópontok között a válasz a valódi leképezéstől jelentős mértékben eltérhet. Ezt az esetet, amikor a torzítás kicsi, de a variancia nagy túlilleszkedésnek (*overfitting*) nevezik. A variancia csökkentése elsősorban a felhasznált tanítópontok számának a növelésével, pontosabban a szabad paraméterek számának és a tanítópontok számának megfelelő összehangolásával lehetséges. A torzítás-variancia dilemma tehát azt a kérdést veti fel, hogy hogyan viszonyuljon egymáshoz a kétféle hibakomponens. A kérdés szorosan összefügg a háló típusának és méretének, komplexitásának a megválasztásával, de kapcsolatban van a tanítópontok számával és magával az alkalmazott tanuló eljárással is. Az átlagos négyzetes hiba ilyen felbontása tehát azáltal, hogy segít a hiba értelmezésében, fontos szerepet tölt be a hálók megfelelő konstrukciójában.[2]

Amennyiben a tapasztalati kockázatot a tanulásnál felhasznált mintapontokra számítjuk, a háló általánosítóképességéről általában semmit sem mondhatunk. A már említett túl illeszkedés miatt ugyanis előfordulhat, hogy a háló a tanítópontokat nagy pontossággal megtanulta, miközben a tanítópontoktól eltérő pontoknál a hiba nagyon nagy is lehet. A háló általánosító-képességének jellemzésére olyan mintapontokra kell meghatároznunk a tapasztalati hibát, melyeket a tanításnál nem használtunk fel. Ennek érdekében egy külön kiértékelő (*validation*) mintakészletet kell használnunk, melynek mintái ugyancsak a megoldandó feladatból származnak, de amelyek függetlenek a tanító készlettől. A kiértékelő mintakészletet sokszor a tanítás folyamata alatt, a tanulás előrehaladtának értékelésére használjuk. Így annak ellenére, hogy ezen minták nem tartoznak a tanításnál közvetlenül felhasznált mintapontok közé, közvetve részei a tanuló eljárásnak. A megtanított háló minősítésére ekkor egy ettől is független mintakészlet, a tesztkészlet szolgálhat. A tesztkészlet szintén az adott problémából származó összetartozó be-kimeneti mintapárok halmaza. A tesztkészlet mintapontjait azonban a tanításnál sem közvetlenül, sem közvetve nem használjuk fel. A tesztkészlet a megtanított háló

végső értékelésére, a háló általánosítóképességének a becslésére használható. A rendelkezésre álló mintapontokat egy problémánál ezért három részre, tanító-, értékelő- és tesztelő készletre kell bontanunk. Kellően nagyszámú adat mellet ez nem okoz nehézséget. Ha azonban kevés adat áll rendelkezésünkre három diszjunkt készletre bontás azt eredményezheti, hogy az egyes részfeladatok elvégzéséhez túl kevés mintapontunk marad. Így, ha viszonylag sok pontot használunk fel a tanításra és kevés marad a megtanított hálózat minősítésére, nem lehetünk biztosak a megtanított hálózat képességeit illetően. Ezzel szemben, ha a minősítésre hagyunk több mintapontot és ezért kevés tanító pontunk lesz, csak azt konstatálhatjuk gyengén tanulta meg a feladatot.

1.8. Paraméterbecslés adaptív momentum becsléssel

A gradiens módszerek hatékony működéséhez a kritériumfelületről bizonyos tulajdonságokat fel kell tételezni. A kedvező konvergencia-sebességet általában csak kvadratikus hibafelület mellett biztosítják, továbbá a minimumhely elérése csak akkor biztos, ha a felületen nincsenek lokális minimumok, vagyis a hibafüggvény felülete unimodális. A neurális hálózatokra tipikusan a nem kvadratikus hibafelületek, a lokális minimumok fennállásának lehetősége és általában a konvergencia szempontjából kedvezőtlen tulajdonságok a jellemzőek. Ez a megállapítás az általunk vizsgált kereslet előrejelzési probléma esetében is fokozottan érvényes. A lokális minimumhelyekben való bennragadás elkerülésére sztochasztikus gradiens eljárásokat dolgoztak ki, amelyek fő jellemzője, hogy a kritériumfelületen valamilyen kisvalószínűséggel a felfelé mozgást is megengedik, lehetővé téve a lokális minimumból való kiszabadulást. A sztochasztikus gradiens eljárások célja, hogy a tanulási folyamat során véletlen zaj felhasználásával lehetővé tegyük, hogy a hálózat a lokális minimumból kikerüljön. A gradiens alapú eljárások lényege, hogy a kimeneten keletkező hibát a háló súlyai szerint deriváljuk, majd a deriválás láncszabálya szerint visszaterjesztjük a mélyebb rétegek felé. A súlyokat pedig a hiba alapján a tanulási ráta függvényében módosítjuk. [2]

A közel múlt leghatékonyabb sztochasztikus gradiens alapú eljárásának az *Adam* bizonyult a kísérletek alapján. A módszert 2015-ben publikálták, ötvözi a korábban leghatékonyabbnak vélt módszerek tulajdonságait, kihasználva azok minden előnyét. A név egy akroníma, amelynek jelentése „Adaptív momentum becslés”. Az algoritmus előnye, hogy végrehajtása kevés memóriát igényel. Minden egyes neuronra önálló tanulási rátát határoz meg a célfüggvény gradiensének első és másodrendű, vagyis centrális momentuma alapján. Az eljárás során a momentumokat a várható értékkel illetve a szórással becsüljük, továbbá az algoritmus az esetleges torzító hatásokat is korrigálja. Az előbb említett tulajdonságoknak köszönhetően az algoritmus immunissá válik a tanítás során a gradiens léptékének megváltozására. A lépésköz hiperparaméternek köszönhetően rögzített lépésközökkel végzi az optimum megkeresését. Az algoritmus pszeudo kódját az alábbiak szerint lehet leírni: [5]

1. lépés: m_0=0 elsőrendű momentum vektor inicializálása,
2. lépés: v_0=0 másodrendű momentum vektor inicializálása,
3. lépés t=0 iteráció számláló inicializálása,
4. lépés: iteráció,

while w_t nem konvergál **do**

- $t = t + 1$
- $g_t = \nabla_0 f_t(w_{t-1})$
- $m_t = \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$
- $v_t = \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$
- $\hat{m}_t = m_t / (1 - \beta_1^t)$
- $\hat{v}_t = v_t / (1 - \beta_2^t)$
- $w_t = w_{t-1} - \alpha \cdot \hat{m}_t / (\sqrt{\hat{v}_t} + \epsilon)$

end while
return θ_t

Ahol:

- m_t:az elsőrendű momentum vektor a t-edik időpillanatban,
- v_t: a másodrendű momentum vektor a t-edik időpillanatban,
- t: az iteráció számláló a tanítás során,
- g_t:a gradienseket tartalmazó vektor a t-edik időpillanatban,
- β_1, β_2: az exponenciális lecsengés ráták a momentum becsléseknek,
- w_t:a neuronháló adott rétegében lévő súlyokat tartalmazó vektor,
- $\hat{v}_t$:a torzítás korrekció utáni másodrendű momentum vektor,
- $\hat{m}_t$: a torzítás korrekció utáni elsőrendű momentum vektor.

Az algoritmus lényegében a hibavisszaterjesztéses algoritmuson alapjaiból indul ki, csak a súlyok módosításának eljárása alakul másképpen. Legyen a $f_t(w_t)$ egy zajos differenciálható függvény w_t súlyok szerint. A cél ennek a függvénynek, a várható értékének a minimalizálása. A függvény sztochasztikus jellege a véletlenszerű batch-ek kiértékeléséből, és a célfüggvényben megbújó zajból származik. Az algoritmus a súlyok megváltoztatásához a gradiensek exponenciálisan mozgóátlagát számítja ki a $\beta_{1,2} \in [0,1)$ paraméterek segítségével, ezek a paraméterek a mozgóátlagok hatásának lecsengését szabályozzák. A mozgóátlagok lényegében becslést adnak a gradiensek elsőrendű és másodrendű momentumaira. Mivel azonban ezeket a vektorokat null-vektorokként inicializáljuk, meg van annak az esélye, hogy a nulla felé torzulnak a súlyok, azonban ezt lehet korrigálni.[5]

Az átláthatóság rovására az algoritmus hatékonysága növelhető, ha a számítási hurokban az utolsó három műveletet az alábbi kettővel helyettesítjük:

$$\alpha_t = \alpha \cdot \sqrt{1 - \beta_2^t} / (1 - \beta_1^t)$$

$$w_t = w_{t-1} - \alpha_t \cdot m_t / (\sqrt{v_t} + \hat{\epsilon})$$

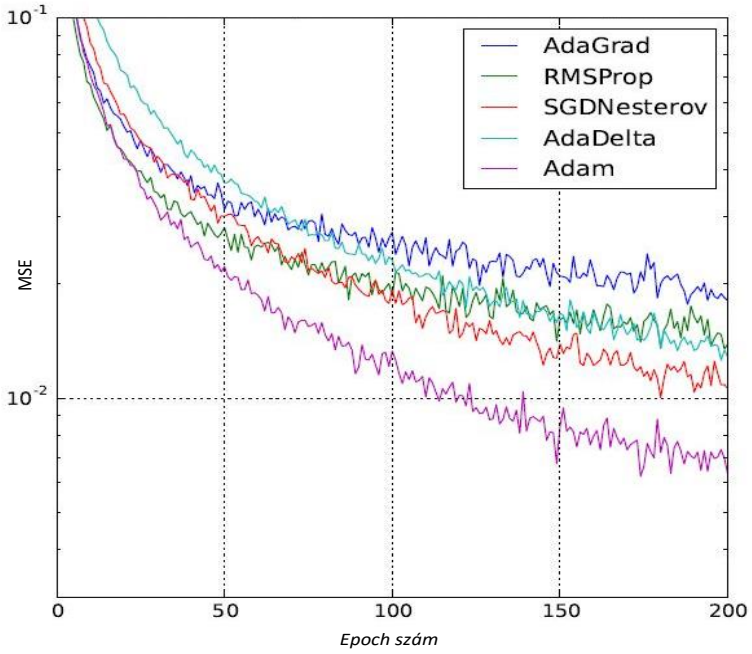
A hatékonyság érdekében a módszertan alkalmazása során a módosított algoritmussal dolgozok annak érdekében, hogy minél jobb hatásfokkal, minél jobb eredményeket érjek el.[5]

Az ideális paraméter beállítások a szerző kísérletei alapján az alábbiak:[5]

- β_1:0.001

- β_2:0.9
- α:0.001
- ϵ:10^{-8}

A módszer hatékonyságát a többi népszerű módszertannal összevetve a következő ötödik ábra mutatja be az MNIST teszt adatsoron.[5]



5. ábra *A tanító algoritmusok hatékonyságának összehasonlítása*

A szerző több kísérletet elvégzett több adatsoron és minden esetben az Adam bizonyult a leghatékonyabbnak. A módszertannak léteznek továbbfejlesztett változatai is, azonban a munkáim során az alap algoritmusnál maradok, mivel ez is eléggé hatékonynak bizonyul a fenti eredmények alapján.[5]

1.9. A neurális hálók konstrukciójának általános kérdései

A neurális hálók gyakorlati alkalmazásánál számos kérdés merül fel. A legkardinálisabb kérdések a hálózatok konstruálásával, illetve ezek tanításával kapcsolatosak. A hálózat méretének megválasztása a rétegek számának és az egyes rétegeken belül a neuronok számának meghatározását jelenti. A rétegek számának növelése három, esetleg még több tanítható réteg alkalmazása azonban megkönnyítheti a feladat megoldását, illetve rétegenként kevesebb processzáló elem felhasználása is elegendőnek bizonyulhat. Az eredmények alapján csak felső becsléseket lehet arra vonatkozóan adni, hogy milyen architektúrával lehet már biztosan jó eredményt produkálni. Azonban fontos kiemelni, hogy a túl sok processzáló elem alkalmazása könnyen túltanulást eredményezhet.[2]

A hálók tanításánál a tanítópontokat általában többször, ismételten felhasználjuk, és a tanítást addig folytatjuk, amíg az eljárás vagy magától le nem áll, vagy megfelelően kis hibát el nem érünk. Bár elvileg a tanító eljárások egy minimumpont elérésekor önmaguktól leállnak, a pillanatnyi gradiens alapú eljárásoknál semmi sem garantálja, hogy nulla gradiens egyáltalán

elő fog fordulni, így legtöbbször a tanulás leállításáról külön gondoskodni kell. Általában előre nem határozható meg a hálózat hibájának alakulása a tanító lépések függvényében, így azt sem lehet megmondani, hogy megfelelő kis hiba eléréséhez hány tanító lépés szükséges. A hálózat "jóságának" kiértékelése, minősítése amúgy is számos elméleti és gyakorlati problémát vet fel. Nem csupán a tanító lépések, hanem a tanító pontok számának meghatározása is kérdéses.[2]

További lehetőségeket jelent, hogy a súlymódosításokat pontonként, vagy kötegelt (*batch*) eljárás szerint, csak a teljes tanító készlet felhasználása után végezzük. Ezek az eljárások a súlykorrekció gyakoriságában térnek el. Az első esetben minden tanító pont alkalmazásánál meghatározzuk a kimeneti hibát és az ebből számítható korrekciós értékkel lépésenként el is végezzük a súlymódosítást. A batch eljárásnál a tanító készlet egy meghatározott részhalmazának összes mintáját egyenként egymás után adva a hálózatra, minden esetben kiszámítjuk a hibát, de az egyes tanító pontokhoz tartozó hibával nem végzünk korrekciót, hanem az egyedi korrekciókat összegezzük és az akkumulált módosítást végezzük csak el. Az első esetben tehát a súlymódosítások száma megegyezik a tanító pontok számának és a tanítási ciklusok számának szorzatával, míg a batch eljárásnál csak ciklusonként korrigálunk, ahol egy ciklus a tanító készlet részhalmazának egyszeri felhasználását jelenti.[2]

Túltanítás (*overtraining*) akkor lép fel, ha a tanító készlet mintáira már nagyon kis hibájú válaszokat kapunk, miközben a kiértékelő készletre egyre nagyobb hibával válaszol a hálózat. Ez azért következhet be, mert a hálózat válaszai túlzottan illeszkednek a véges számú tanító pont által megszabott leképezéshez, a tanító pontokban akár tetszőlegesen kis hiba is biztosítható miközben a közbenső pontokra adott válaszok jelentősen eltérhetnek a megfelelő kívánt válaszoktól. Annak érdekében, hogy a háló jóságáról képet kapjunk a hálót a teszt adatsorokon szoktuk letesztelni annak érdekében, hogy a tanuló adatsortól függetlenül tudjunk a viselkedésről képet alkotni.[2]

További nagyon fontos paraméter tanulási algoritmus tanulási rátájának megválasztása, amely meghatározza, hogy az egyes tanítási iterációs lépéseket követően a súlyok mekkora értékben kerüljenek módosításra a visszaterjesztett hiba alapján. A legfrissebb eredmények azt a sejtést engedik következtetni, hogy a tanulási rátát legcélszerűbb a tanítási folyamat során változtatni, azonban kőbe vésett szabályok ebben az esetben sincsenek.

A fent tárgyalt probléma körök komoly megválaszolandó kérdéseket vetítenek előre a munkámat illetően, legyen szó akár a kereslettervezés folyamatának támogatásáról, akár komplex előrejelző rendszerek készítéséről, hiszen a mesterséges intelligenciák ezen a téren való alkalmazása relatív friss tudománynak számit, ebből kifolyólag nincsenek általános érvényességű szabályok. Célom a továbbiakban feltárni irodalom kutatása, majd a későbbiekben kísérletek segítségével, hogy mik a legjobb módszertani megközelítések a különböző kereslettervezési feladatok esetében, amennyiben neurális hálókkal kívánjuk ezeket megoldani.

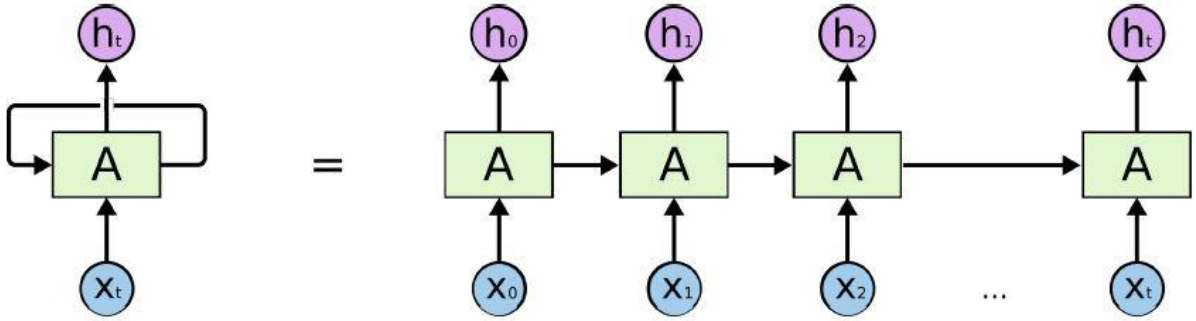
1.10. Visszacsatolt neurálishálózatok

Ahogyan az korábban már említésre került, egyes hálózatok topológiájában találkozhatunk visszacsatolással. Ez a visszacsatolás lehet lokális vagy globális jellegű. Az alapvető ötlet a

visszacsatolás mögött, hogy a hurok által felruházhatjuk a mesterséges intelligenciát egy olyan képességgel, hogy a korábban betáplált információk fényében mondja, adja meg a következő predikciót. [6]

A globális visszacsatolásnak számos formája lehetséges. A háló kimenetét visszacsatolhatjuk a bemenetre, valamelyik rejtett réteg, vagy akár több rejtett réteg kimenetét visszacsatolhatjuk a bemenetre, vagy akár az előbbiekben elhangzottak tetszőleges kombinációját is lehet alkalmazni. A visszacsatolásnak köszönhetően dinamikusan vezérelt hálózatokról beszélhetünk. A huroknak köszönhetően lehetővé válik az állapot reprezentáció, amelynek köszönhetően kiválóan alkalmassá válik nemlineáris predikciók elkészítésére, kommunikációs csatornák adaptív kiegyenlítésére, beszéd felismerésre, valamint előrejelzési feladatok elvégzésére. A globális hurok jelentette előnyös tulajdonságoknak köszönhetően a visszacsatolt hálózatok sok esetben alkalmasabbak az előzőekben elhangzott feladatok maradéktalan elvégzésére, mint a más egyéb módszerek, például szűrők segítségével dinamikussá tett neurális hálók, mi több a hurok potenciálisan csökkentik a szükséges memória mennyiségét.[7]

Az első pillantásra igencsak bonyolultnak hangozhat a visszacsatolással bíró neurálishálók működése, azonban jobban átgondolva az architektúrát, úgy tekinthetünk a hurokra, amennyiben egy háló vagy egy réteg kimenetét csatoljuk vissza önmaga bemenetére, mintha ugyanazon réteg vagy háló számtalan másolatáról lenne szó, egymás után sorban, ahol az első rész az utána következő számára is küld bemenetet, a kimenet előállításán túlmenően. Ezt szemléletesebben az alábbi hatodik ábra mutatja be.



6. ábra *A visszacsatolás időbeli kigörgetése*

A láncszerű szerkezet felfedi, hogy valójában ideális konstrukció a szekvenciák és listák feldolgozására, ennél fogva alkalmassá tehető kereslet előrejelző modellek kreálására. Az elmúlt évek kutatásai számos alkalmazhatóságra és előnyös tulajdonságra derítettek fényt a visszacsatolt neurálishálók kapcsán.

1.11. A hosszú távú függések megtanulásának kérdésköre

A visszacsatolt neurális hálók felhasználhatók szekvenciális bemenetek feldolgozására és szekvenciális kimenetek előállítására, ezek közé tartoznak a felismerő típusú feladatok, a hangfeldolgozás, valamint az előrejelzési feladatok. Azonban az olyan feladatok kivitelezése során, amelyeknél időbeli függőségek állnak fenn, a tanítás nehézségekben ütközhet.[7]

A fentebb említett feladatok esetében olyan hálózatokra van szükség, amelyek eltárolják és frissítik az összefüggő információkat. A visszacsatolt hálózatok azért különösen alkalmasak

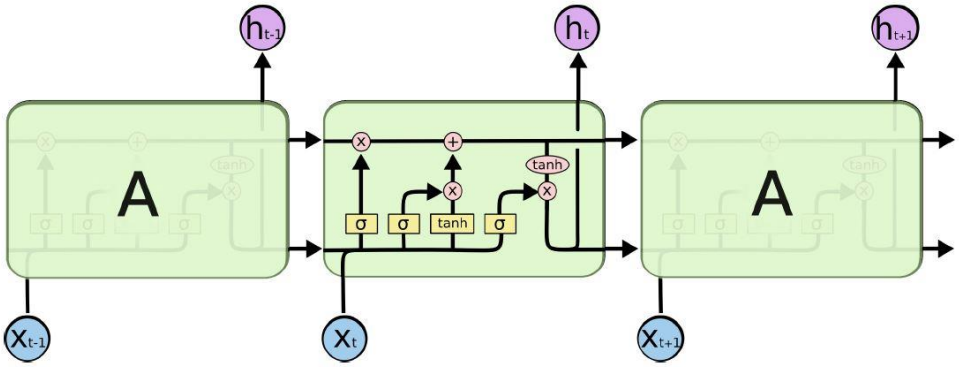
erre, mivel belső állapotuk leírható. A háló hurkai lehetővé teszik az információ tárolását a korábbi bemenetekről egy bizonyos ideig, ami a súlyoktól és a bemenetektől függ. Egy szekvenciális bemenettel bíró háló felhasználható arra, hogy szekvenciális kimenetet adjon figyelembe véve egyéb környezeti információkat. A kauzális rendszerek esetén hosszú távú függésekről beszélünk, amennyiben a következő meghatározandó t időpillanatban felvett értékre egy sokkal korábbi τ időpillanatot jellemző mennyiség is ráhatással van, vagy $t \gg \tau$. Bár sok esetben a visszacsatolt hálózatok könnyen jobb teljesítményt adnak adott feladat elvégzése során, azonban a fentebb vázolt hosszú távú függések esetén a tanításuk meglehetősen nehézkes lehet. A kísérletek azt mutatják, hogy a visszacsatolt hálózatok paraméterei ilyen esetekben úgy alakulnak, hogy azok csak a rövidtávú függéseket veszik figyelembe. Ennek okát a hibavisszaterjesztéses algoritmusban állapították meg a kutatók, ebből kifolyólag a tanításra más alternatívákat javasoltak, mint például a szimulált hűtés, vagy például az idősúlyozott áll-Newton optimalizáció.[8]

A kutatások azt mutatják, hogy bár alap esetben a visszacsatolt hálózatok jobban teljesítenek, a hosszú távú függések esetében a gradiens alapú tanító eljárások nem alkalmasak ezek kimutatására, megtanítására. Amennyiben feltesszük, hogy a belső állapotokra vonatkozó információt hiperbolikus attraktorokban kerül tárolásra, a hurokkal ellátott rendszer vagy nem lesz robosztus a zajokra, vagy pedig nem lesz hatékonyan tanítható gradiens alapú eljárásokkal. Bár a lefolytatott kísérletek nem zárják ki annak a lehetőségét, hogy meg lehessen tanítani egy visszacsatolt hálót egy adott feladat elvégzésére, azonban kétségtelen, hogy a hagyományos gradiens alapú eljárások hatékonysága nagymértékben romlik, bár ez javítható a súlyok kezdeti helyes megválasztásával.[8]

1.12. Az LSTM hálók

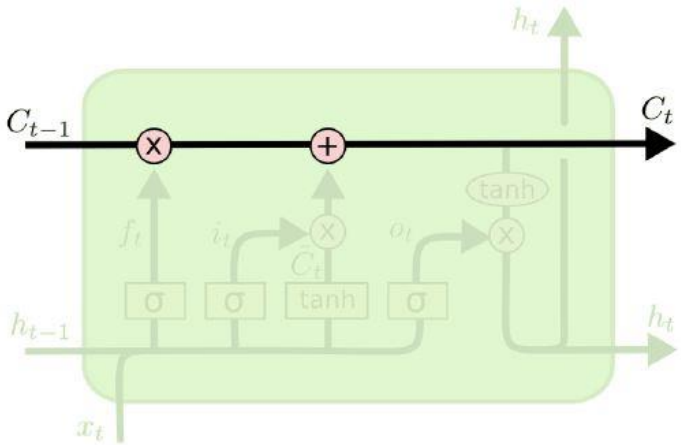
Az LSTM (*Long- Short Term Memory*) hálók a visszacsatolt neurális hálók egy speciális változata. Napjainkban tapasztalt népszerűsége elsősorban annak köszönhető, hogy széleskörűen alkalmazható, valamint a fentebb tárgyalt hosszú távú függések problémáját eliminálja szerkezetének köszönhetően. A hálókat többek között ennek a problémának a kiküszöbölésére hozták létre és alap tulajdonsága, hogy az információt hosszú időre eltárolja. Szerkezetileg hasonló felépítéssel bír, mint az egyszerű visszacsatolt hálók. Azonban elemeit tekintve nem hagyományos neuronokból épül fel, hanem blokkszerű elemekből. Minden egyes blokk önmagába vissza van csatolva. Az LSTM szerkezetét az alábbi hetedik ábra mutatja be.[6]

A blokkon belül és a blokkok között a nyilak a vektoriális kapcsolatokat reprezentálják. A vektorokkal elvégzett műveletek elemenként történnek. Egy adott LSTM blokk három logisztikus és egy hiperbolikus tangensfüggvényt tartalmaz. Az ábrán az *x*-ek a bemeneti vektorokat reprezentálják, míg *h*-val az egyes modulok kimeneti vektorait ábrázoljuk.



7. ábra *Az LSTM blokk vázlata*

Amint az ábrán láthatjuk, lényegében a neuronhálók tipikus építőkövét, az elemi neuront memória egységekkel helyettesítjük. Maguk a blokkok pedig tartalmaznak egy tényleges neuront, valamint kapukat. A kulcs aspektusa az LSTM hálóknak az úgynevezett cella állapot. A kapuk segítségével módosíthatunk a cella állapoton. A cella állapot rendelkezik a korábbi bemenetekről információval. A módosított cella állapotot a kalkulált kimenettel együtt visszacsatoljuk. A cella állapotot a blokkon belül a következő nyolcadik ábrán láthatjuk.[6]



8. ábra *Az LSTM cella belső állapota kiemelve*

Az LSTM hálók egy egyedi tulajdonsága, hogy fel vannak ruházva a felejtés képességével, megnézi h_{t-1} kimenetet, valamint az x_t bemenet és ez alapján kalkulál a cella állapot minden értékére egy számot a [0,1] intervallumon és elemenként összeszorozza ezeket. A következő lépésben azt kell eldönteni, hogy milyen új információt adunk a cella állapothoz. Először egy logisztikus függvénnyel eldöntjük, hogy mely értékek kerülnek módosításra, ezután egy hiperbolikus tangens függvény segítségével új értékeket generálunk a megfelelő helyekre, ezeket kombinálva végül pedig felfrissítjük a cella állapotot. Egyenletek formájában az alábbiak szerint lehet leírni. A h_{t-1} és x_t vektorokat összefűzzük.[6]

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t], +b_f)$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t], +b_i)$$

$$\tilde{C}_t = tanh(W_C \cdot [h_{t-1}, x_t], +b_C)$$

Ahol:

- W: adott elemhez tartozó súlyok,
- h_{t-1}:a (t-1)-edik időpillanat kimenete,
- x_t: a t-edik időpillanat bemenete,
- b: a torzítás korrekciója,
- f_t: a felejtő kapu válasza,
- i_t: a bemeneti kapu válasza,
- $\tilde{C}_t$: a választó kapu válasza.

Miután kiszámításra kerültek a fenti értékek, vagyis elemenként beszorozzuk f_t értékeivel, majd ezt követően hozzáadjuk i_t és $\tilde{C}_t$ elemenkénti szorzatait pontonként. Ezt követően az új cella állapotot egy hiperbolikus tangens függvénnyel megszűrve, valamint a közvetlen bemenetet feldolgozó neuron válaszát elemenként összeszorozva megkapjuk a blokk kimenetét. Egyenletes formában pedig az alábbiak szerint lehet leírni. [9]

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t], + b_o)$$

$$h_t = o_t \cdot \tanh(C_t)$$

Az LSTM hálók számos előnyös tulajdonsággal rendelkeznek. Architektúrájuk lehetővé teszi a hosszú távú függések megtanulását, valamint robosztus a zajjal szemben. A rejtett Markov modellekkel szemben nem igényel apriori információt az állapotok véges számáról, így elméletben végtelen számú állapot számmal is dolgozhat. Az LSTM hálók rendkívül jó általánosítóképességgel bírnak. A háló mindazonáltal képes a jól elkülönülő jelenségek megkülönböztetésére és megtanulására anélkül, hogy rövidtávú függések jelentkeznének a tanuló minták alapján. Az LSTM általában gyorsan tanítható és a paraméterek előzetes megválasztására általában nincs szükség. A visszacsatolásnál negatív súlyokat generálva nagyobb hangsúlyt tud fektetni a közelmúlt eseményeire és képes megtanulni a lecsengési rátákat.[10]

A fent felsorolt tulajdonságok az LSTM hálókat kiválóan alkalmassá teszik keresleti idősorok megtanulására. Hiszen ahhoz, hogy egy zajos adatsort jól megtanuljon, jó általánosítóképességgel kell, hogy bírjon. Továbbá komplex ökonometriai adatokban egyértelműen megtalálhatók hosszú és rövidtávú függések egyaránt. Ki kell még emelni, hogy azáltal, hogy a háló képes a jól elkülönülő jelenségek megkülönböztetésére, még inkább válik potenciálisan alkalmazható eszközzé a kereslettervezés területén, hiszen az idősor komponenseit (alapjel, trend, szezon, zaj) jól kell tudni azonosítani

Számos előnye ellenére azonban az LSTM-nek is akadnak korlátai. Az egyik ilyen akadály, amit nehezen tud leküzdeni az ”erősen késleltetet kizárólagos-vagy” problémák, ahol is a cél két jól elkülöníthető bemenet XOR-jának kiszámítása. Ennek oka, hogy az előző eredmény eltárolása nem segít a következő kimenet meghatározásában. A moduláris felépítés jóval komplexebb számítási igénnyel bír, mivel a módosítható súlyparaméterek száma is megnövekszik. Vannak feladatok továbbá, amelyek egyszerű súly tippeléssel könnyen megoldhatóak, mint például 500-bites paritás probléma, azonban LSTM hálókkal ezek jóval bonyolultabb feladatnak ígérkeznek. Ha egy esemény bekövetkezésénél számít, hogy

kilencvenkilenc vagy száz lépéssel ezelőtt történt, akkor a hálót számlálóval kell ellátni. Azonban ha mondjuk a harmadik és tizenegyedik lépést kell megkülönböztetni, az nem okoz a hálónak problémát.[10]

1.13. A Keras

A Keras alapvetően olyan önálló Python alapú könyvtár, amely háttér platformként számos numerikus könyvtárat tud használni, köztük a legnépszerűbb Google által fejlesztett Tensorflow-t és az akadémiai gondozásban lévő Theano-t is. A program egy olyan interfészként funkcionál, amely lehetővé teszi az end-to-end jellegű feladat megoldást a komplexebb kódolási igények nélkül. A programnak meg van a maga számítási gráf leképző módszere, azonban a számításokat nem önállóan, hanem csak a háttér könyvtárak segítségével tudja elvégezni. Ennek a szerkezetnek nagy előnye, hogy a struktúra definiálások során nagyban megkönnyíti a programozó dolgát, azonban nem enged mélyebb betekintést a végrehajtandó mechanizmusok mögé és így csak feketedoboz jellegű modelleket lehet benne létrehozni. A modellből bár ki lehet nyerni az egyes rétegek súlyait, illetve kimeneteit, a struktúrára nincs a programozónak mélyebb ráhatása, csupán az építőkomponensek típusát tudja változtatni egy relatív szűk tartományban, illetve ezek egymásután való következésének sorrendjét, valamint az egyes rétegeket alkotó komponensek legfontosabb paramétereit.[11]

A szerzők célja elsődlegesen egy olyan platform létrehozása volt, amely a nagy általánosságban használt, legalapvetőbb elemekből felépíthető modellek leprogramozását nagymértékben megkönnyíti. Ezzel a megközelítéssel a szoftver az egyszerűbb klasszifikációs, regressziós és képfeldolgozó munkák esetében igen nagy népszerűségnek örvend. Azonban a komplexebb gráf struktúrák létrehozásában komoly korlátokat tud támasztani a fejlesztők előtt.[11]

Mivel alapvetően a számítások a háttér könyvtárak segítségével megy végbe, a hosszabb tanítási folyamatok esetében arra kell számítani, hogy a tanulási folyamat nagymértékben el fog húzódni. Ennek oka abban keresendő, hogy a háttér program és az interfész közötti kommunikáció sok számítási kapacitást igényel és ebből kifolyólag drámain lelassítja a folyamatok végrehajtását.

Bár kétségtelen, hogy a modellek létrehozása során az alsóbb szintű programcsomagok jóval nagyobb szabadságot és mozgásteret biztosítanak, a Keras moduláris felépítése minimalista stílusra való törekvése, könnyű kezelhetősége és bővíthetősége kiválóan alkalmassá teszi a mesterséges intelligenciák bemutatására és megismertetésére a kezdő felhasználók számára.

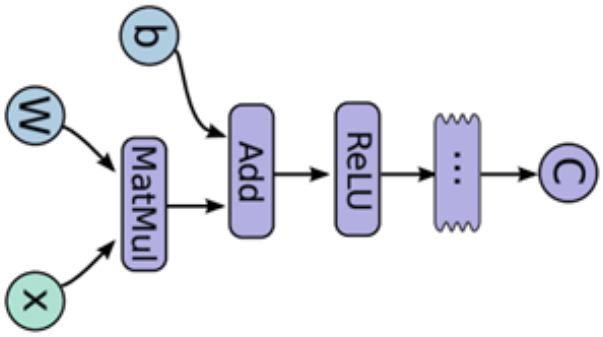
További hátrány, hogy a Keras egymást követő frissítései nem teremtik meg a kontinuitást, mivel eltérő verziókon implementálva ugyanazt a modell struktúrát, jelentősen eltérő eredményt kapunk, amely a véletlen szám generátorból adódó eltérésekkel nem magyarázható és fekete doboz jellegéből adódóan nem is kaphatunk rá komolyabb magyarázatot.

Véleményem és az előzetes kísérletek alapján, amelyet a harmadik fejezetben bővebben taglalok, a Keras nem engedi meg azt az alkotói szabadságot, amely ahhoz szükséges, hogy számos modell struktúrát kipróbálhassak, ezért néhány kísérletet követően a Tensorflow-ra helyezem át a hangsúlyt a különböző struktúrák vizsgálatánál, valamint az időtakarékosságot szem előtt tartva.

1.14. A Tensorflow

A Tensorflow egy olyan könyvtár, amely Python és C++ programnyelveken lehet használni mesterséges intelligenciák programozására. A programcsomag egy alacsony kódoltság szintű interfész felületet hoz létre bonyolult számítási algoritmusok elvégzésére. A rendszer kellő rugalmasságot biztosít a felhasználónak az előre definiált objektumaival és megkönnyíti a komplex hálók implementálását. A könyvtár alapvetően képes processzoron, illetve videókártyán is számításokat elvégezni, azonban a nagyobb rendszerek esetében mindenképpen preferált a GPU elemen való futtatás. A Tensorflow első változata még 2015-ben jelent meg, azóta viszont népszerűsége robbanásszerűen nő, köszönhetően a lehetőségek széles tárházának, amit kínál.[12]

A programnyelv lehetővé teszi, hogy a definiált számításokat irányított gráfként lehessen értelmezni. A gráf struktúrája pedig az adatfolyam irányát reprezentálja a számítások során. A kódolás eredményeként az alábbi struktúrához hasonló számítási gráfokat tudunk definiálni. Az kilencedik ábrán egy egyszerű perceptron látható lényegében, ahol b a torzítás W a súlyokat és x pedig a bemeneteket ábrázolja. [12]



9. ábra *Perceptron struktúra Tensorflowban*

A struktúrát az alábbi kódolással lehet meghívni:

```
import tensorflow as tf

b = tf.Variable(tf.zeros([100]))
W = tf.Variable(tf.random_uniform([784,100],-1,1))
x = tf.placeholder(name="x")
relu = tf.nn.relu(tf.matmul(W, x) + b)
C = [...]
```

A Variable segítségével tudjuk definiálni, a gráf olyan változóit, amelyeket huzamosabb ideig tárolunk és a tanítási folyamat során a hiba visszaterjesztési algoritmus segítségével folyamatosan frissítünk. A placeholder változók arra szolgálnak, hogy egy iteráció erejéig, legyen szó akár a tanításról akár a háló alkalmazásról, a bemeneti adatokat tárolja és gráfba juttassa ezeket. A számítások, illetve a könyvtár alapját a tenzorok képzik. A tenzor egy olyan több dimenziós mátrixként értelmezhető, amelyek struktúrájuknak köszönhetően biztosítják GPU memória legjobb kihasználását.[13]

A Tensorflow-t alapvetően minden operációs rendszeren lehet használni, C++ vagy akár Python programnyelven is. Bár kompatibilitása hivatalosan nem hardver függő fontos megjegyezni, hogy a videókártya alapú futtatása csak az Nvidia videókártyákon támogatott, azon belül is

leginkább a Pascal és Tesla típusú kártyákat javasolják, amelyeket külön erre a célra fejlesztettek. Természetesen a könnyebben hozzáférhető átlagosabb Nvidia GPU-kal is kompatibilis program.[12]

Ahhoz, hogy a Tensorflow-t a GPU-n futtatni lehessen, több kiegészítő szoftver kell telepíteni, amelyek alkalmassá teszik videokártya meghajtót a komplex műveletek elvégzésére. Az ilyen programok közé tartozik CUDA kiegészülve Cudnn könyvtárakkal. Azonban az előzőek telepítése csak akkor kivitelezhető, ha mindezeket megelőzően telepítettük a Microsoft Visual Studio 2015-ös változatát.[13]

A Tensorflow azon túlmenően, hogy a lehetővé teszi a teljes gráf kontrolját és a számítások átláthatóságat azáltal, hogy a programozónak magának kell ezeket implementálnia az előre definiált objektumok segítségével, a Tensorboard kiterjesztés segítségével betekintést enged a gráf struktúrájába és ezáltal könnyebben megérthetővé válik a modell működése.

2. A neurális hálók lehetséges alkalmazása a kereslet előrejelzésben

Napjaink piaca igencsak szürke képet mutat a vállalati problémák keresztmetszeti tervezésének terén. A „sales and operation planning” koncepciói még a legtöbb mamut vállalat működésében sem tölt be jelentős szerepet. A szegregált tervezési területek alkalmanként más-más motivációkkal bírnak. A szeparált vállalati részlegek saját indikátoraik alapján dolgoznak egyéni célkitűzéseiknek elérése érdekében. A rossz vállalati mutatószámrendszer és célkritérium rendszer azt eredményezi, hogy a vállalati pareto optimumot nem lehet megtalálni. Éppen emiatt a cél egy olyan vállalati indikátor halmaz felépítése, amely támogató szerepet tud biztosítani, az egyes vállalati divíziók terveinek iterálásában. Egy helyesen felállított monitorozó rendszer lehetővé teszi a rendszer szintű optimum megkeresését. Azonban ahhoz, hogy ez kivitelezhető legyen elengedhetetlen, a vállalati produktum megfelelő megbízhatósági színvonalon történő előrejelzése és csak ezután lehetséges a megfelelő erőforrás gazdálkodás megtervezése.

2.1. A kereslet előrejelzés hagyományos módszerei

A kereslettervezés folyamatát számos tényező befolyásolja, a hektikus kereslet, jelentős szezonhatás, az igényt befolyásoló külső és belső hatások, valamint nehezíti a feladatot, hogy a termékeket differenciáltan kell kezelni. A szakirodalomban fogalmak alapján fontos leszögezni, hogy a kereslettervezés nem egyenlő az előrejelzés készítéssel. A kereslettervezés, vagy idegen szavakkal élve a „demand planning” egy rendkívül összetett, komplex feladat, amely az alábbi szegmensekre bontható fel:

- adatelőkészítés,
- identifikáció,
- előrejelzés,
- finomtervezés.

Az adatok előkészítése akár a teljes tervezési folyamat felét is kiteheti. Az első fázis keretében körültekintő információgyűjtést kell végrehajtani, amellyel a további szakaszok elvégzését meg lehet alapozni. A tervezésnek, ennek a szakaszában minden adott iparág specifikus és keresletet befolyásoló tényezőt be kell gyűjteni annak érdekében, hogy az adatokat megfelelő formátumra lehessen hozni, mivel rossz adatra nem létezik megfelelő eljárás. Továbbá, ebben a fázisban történik az adatok megfelelő formátumra hozása, vagyis az adatok tisztítása. Az identifikációs fázisban az adatsorokban megbújó jelenségeket kell azonosítani, hogy a megfelelő módszert kiválaszthassuk az előrejelzési modell felállításához. Az előrejelzés szakaszában az érvényes matematikai modellt kell felállítani a megfelelő paraméterek segítségével. Ezt követően a tervezési időhorizonttal összhangban kell a modell segítségével a megfelelő jövőbeli adatokat előállítani. Az előrejelzést a finomtervezés követi. A finomtervezés során a keresletet befolyásoló tényezők hatásait statisztikai eszközök segítségével rászuperponáljuk az előrejelzett adatokra. A tudományos keretek között folyó munkálatok általában a modell felállításával és tesztelésével véget érnek, azonban a vállalati keretek között elengedhetetlen a felállított modellek nyomon követése, valamint minősítése mutatószámok segítségével.

2.1.1. Adatelőkészítés

Az adatok előkészítése során lényegében strukturálatlan adatokból egy rendezett és rendszerezett adatbázist építünk fel, amely a kiindulási alapját képzi a további műveleteknek. A tervezésnek, ennek a fázisában kerül sor az adatok aggregálására is. Az adatokat több dimenzió mentén lehet kumulálni. Az előrejelzés céljától függően az adatokat lehet terület szerint, időszeletek szerint, értékesítési csatornák szerint, valamint termékcsoportonként aggregálni, illetve ezek tetszőleges kombinációja is felmerülhet. Az előzőekben említettekre azért van szükség, mivel így felszínre lehet hozni az adatokban megbújó sajátosságokat, ki lehet simítani az idősort ezzel csökkentve a zaj hatását és felerősíteni az adatsorra jellemző jelenségeket. További fontos előnye, hogy az esetleges sporadikus attribútumok szintén eltüntethetők ezáltal, ez azért fontos, mivel a sporadikus adatsorok előrejelzése jelentős nehézségek elé állítják a kereslettervező szakembereket.[14]

Az adatok kumulációja mellett rendkívül fontos az adatok osztályozása. A vállalati eredményességre nem egyforma hatással vannak a szortiment egyes elemei, ennél fogva célszerű súlyokat rendelni az egyes cikkekhez a szortiment analitikai vizsgálatok alapján. Az osztályozáson túlmenően a kereslettervezési adatbázisnak szerves részét kell, hogy képezze minden olyan számszerűsíthető és nem számszerűsíthető adat, amely befolyást gyakorolhat a kereslet időbeli alakulására, valamint a kereslettervezés pontosságára. Ezek az információk lehetnek külső vagy belső információk, valamint múltra és jövőre vonatkozó adatok egyaránt lehetnek. Továbbá az adatok között lehetnek számszerűsíthető adatok, valamint kevésbé számszerűsíthető adatok. A szóban forgó adatokat részletesen az alábbi első táblázat tartalmazza.[14]

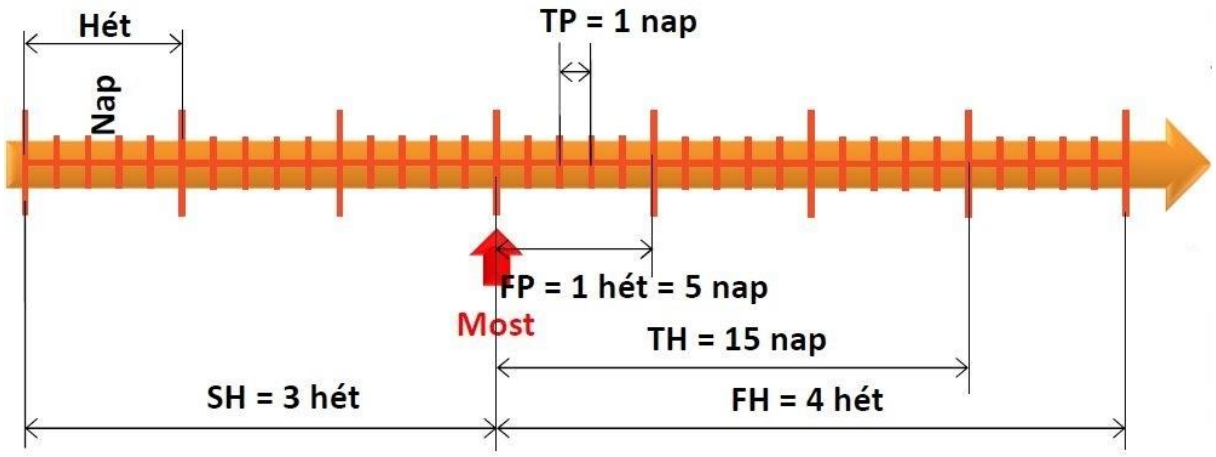
1. táblázat *A keresletet és a kereslettervezést befolyásoló információk*

	Múltra vonatkozó információ		Jövőre vonatkozó információ	
	Külső információ	Belső információ	Külső információ	Belső információ
számszerű-síthető	környezeti hatások	tény megrendelések	beruházások	várható megrendelések
	uralkodó trendek	tény értékesítések	piaci mechanizmusok	tervezett értékesítések
	múltbéli befektetések	aktivitások	környezeti hatások	tervezett aktivitások
	időjárás alakulása	tervezési pontosság	időjárás előrejelzés	modell várható pontossága
	...	tervezett értékesítések	...	...
nem számszerű-síthető	piaci hatások	termék specifikus jellemzők	piaci hatások	termék specifikus jellemzők
	információ a konkurenciáról	technikai állomány	információ a konkurenciáról	technikai állomány
	gazdasági hatások	"know-how"	gazdasági hatások	"know-how"
	politikai döntések	partnerek jellemzői	szóbeszéd	partnerek jellemzői

Az információ gyűjtésen túl az adatok előkészítésénél talán az egyik legfontosabb az adatok tisztítása. Az adatokban lehetnek hibák, úgynevezett „peakek”, negatív és pozitív kiugrások egyaránt. Ezektől elsősorban két okból kell megtisztítani az adatot, az identifikációs fázisban megnehezítené az adatsori sajátosságok azonosítását, ami rossz modell meghatározáshoz vezetne, továbbá modelltől függetlenül az előrejelzési adatokat torzítaná. Az adattisztítást jellemzően szűrők segítségével végzik el. A leggyakoribb alkalmazott szűrők az alul- és felülszűrők.[14]

Az előkészítési fázis egyik fontos eleme az időhorizontok kezelése. Az adatokat fontos olyan időformátumúvá alakítani, hogy az összhangban legyen a tervezési időhorizontokkal. Az

igénytervezési időhorizont az igénytervezési időperiódus és készlettervezési időperiódus egész számú többszöröse. A igénytervezési időperiódusnak továbbá összhangban kell lennie a mintavételezési időhorizonttal is. A készlettervezési időhorizont pedig egész számú többszöröse a készlettervezési időperiódusnak. A készlettervezési időperiódust a rendelésütemezési politika határozza meg, megadja azt az időtartamot, amely két újrarendelés között eltelik. Az igénytervezési időperiódus pedig megadja azt az időtartamot, amelyre az előrejelzési modell megállapítja az igény mértékét. A készlettervezési időhorizont azt az időtartamot adja meg, amelyre az erőforrásainkat előre megtervezzük. Az igénytervezési időhorizont pedig megadja, azt az időtávlatot, amelyre a keresletet előre jelezzük. Az időhorizontok egymáshoz való viszonyát az alábbi tizedik ábra mutatja be. A készlettervezési időperiódust TP-vel, az igénytervezési időperiódust FP.-vel, a keresletttervezési időhorizontot TH-val, az igénytervezési időhorizontot FH-val, a mintavételezési időhorizontot pedig SH-val jelöltem az ábrán.[14]



10. ábra *Az időhorizontok egymáshoz való viszonya*

Az adatelőkészítés lényegében a számtalan forrásból származó strukturálatlan adat rendezett keretek közé terelését és feldolgozását jelenti, amelynek főbb lépéseit az alábbiakban lehet összefoglalni:

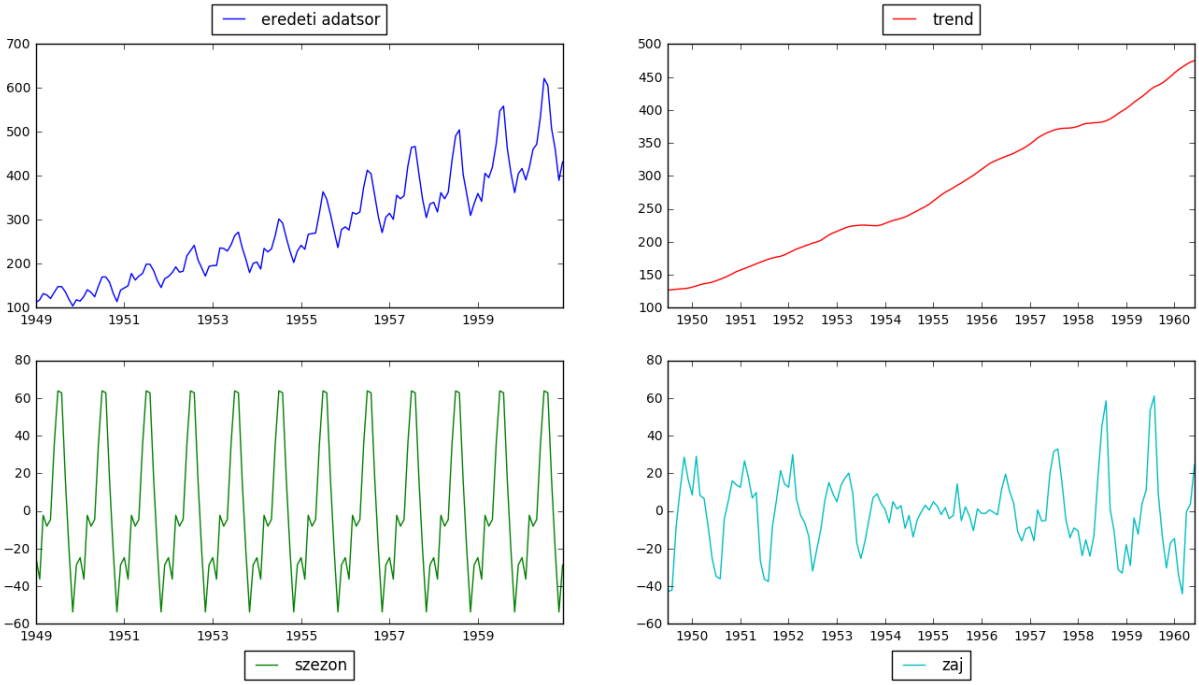
- bemeneti adatok összegyűjtése,
- aggregációs szint meghatározása,
- a tervezési időhorizont meghatározása,
- a termékek osztályozása,
- az adatok vizsgálatra alkalmas állapotba hozása.

Az adatokat célszerűen, könnyen elérhető formában, vállalat irányítási rendszerrel összekötött adatbázisban kell tárolni.[14]

2.1.2. Identifikáció

Az identifikáció a leghangsúlyosabb eleme a kereslettervezés folyamatának. Ennek segítségével lehet azonosítani az előrejelzésre alkalmas modellek összességét. Cél az elemzett adatsor stacioner állapotának felszínre hozása, ezáltal előhozva az adatsorban rejlő olyan

sajátosságokat, mint például a szezon és trend hatások. A stacionaritás azt jelenti, hogy egy sztochasztikus folyamat együttes eloszlása az időbeli eltolásra nézve invariáns, vagyis közömbös. A keresleti adatsorok alapvetően négy fő tényezőre oszthatóak a keresleti alapjelre, a trendre, a szezonhatásokra, valamint a zajra. Az alapjel azt az igény mennyiséget írja le, amely akkor is elkelne a piacon a termékből, ha nem létezne semmilyen befolyásoló tényező. A trend pedig a termék életciklusából adódik, attól függően, hogy felfutó vagy kifutó termékről van szó. A szezonhatás pedig a terméksajátossága, amely szabályos időperiódusonkénti keresletfelfutásban, majd visszaesésben nyilvánul meg. Az alábbi tizenegyedik ábra egy átlagos keresleti adatsort lehet látni kiemelve az idősorban megbúvó sajátosságokat.[14]



11. ábra *Egy tipikus keresleti idősor összetevőire bontva*

Az identifikáció lefolytatásához a bemeneteket az adatelőkészítés során felépített tudásbázis szolgáltatja. Az adattranszformálást a tisztított adatsorokon hajtjuk végre. A transzformáció célja, hogy a következő adatsorokat megkapjuk:

- az eredeti idősor első normál differenciája (D0d1),
- az eredeti idősor második normál differenciája (D0d2),
- az eredeti idősor első szezonális differenciája (D1d0),
- az eredeti idősor első szezonális differenciája, illetve ennek első normál differenciája (D1d1),
- az eredeti idősor első szezonális differenciája, illetve ennek második normál differenciája (D1d2).

Az eredeti idősort D0d0-al jelöljük, amelynek a hossza *n* és a transzformált idősorokat az alábbiak szerint állítjuk elő, a képletekben az *s* a szezonhatárt jelenti:[14]

- D0d1 adatsor: i=2…n:

$$D0d1(i) = D0d0(i) - D0d0(i-1)$$

- D0d2 adatsor:i=3…n:

$$D0d2(i) = D0d1(i) - D0d1(i-1)$$

- D1d0 adatsor: i=s+1…n:

$$D1d0(i) = D0d0(i) - D0d0(i-s)$$

- D1d1 adatsor i=s+2…n:

$$D1d1(i) = D1d0(i) - D1d0(i-1)$$

- D1d2 adatsor:s+3…n:

$$D1d2(i) = D1d1(i) - D1d1(i-1)$$

A szezonhatárok megállapításában jellemzően arra tudunk támaszkodni, hogyha megbújik az adatsorban ilyen típusú jelenség, akkor az évenként ismétlődik, így ennek megfelelően kell az s paraméter nagyságát megállapítani annak függvényében, hogy hetenkénti vagy pedig havi bontásban dolgozunk. A heti bontásnál problémát jelenthetnek az év végi tört hetek, ilyenkor jellemzően 52 hetes évi bontásban gondolkoznak a szakemberek. A szakirodalom azt javasolja, hogy a differenciált adatsorok elkészítésénél szezonális differenciálás esetében egynél mélyebbre, illetve normál differenciálás esetében kettőnél mélyebbre ne menjünk. Az így kapott adatsorokon végezzük el a korrelációs vizsgálatokat. Sajnálatos módon az idősorok transzformációja adatvesztéssel jár, ami a korrelációs vizsgálatok megbízhatóság csökkentheti kevés adat esetén. A tesztek során autokorrelációs és parciális autokorrelációs függvényeket alkalmazunk.[14]

Egy idősorban az adatok időbeli összefüggését autokorrelációval lehet vizsgálni. Az autokorrelációs és parciális autokorrelációs függvények értékeivel lehet felmérni, hogy az idősor adott elemére milyen mélységben van hatással a múltja, azaz a korábbi időpontbeli változók közül mennyi van szignifikáns korrelációban vele. Ezek az értékek is, mint a korrelációs koefficiens -1 és 1 közötti értékek lehetnek. A 0-hoz közeliek a lineáris kapcsolat hiányát mutatják, míg az abszolút értékben 1-hez közeliek erős függvényszerű kapcsolatot jelentenek. Ha az egyik adatsort eltoljuk, akkor késleltetett hatások is felfedezhetők. A függvények grafikus ábrázolását korrelogramnak nevezzük. [14]

A különböző időpontok, mint az idősor egyes komponens valószínűségi változói közötti korrelációt autokorrelációnak szokás nevezni az „auto” utal arra, hogy az idősor szintjén „önmagával vett” korrelációról van szó. Más szóval az autokorrelációs függvény i-edik eleme megadja a korreláció erősségét az eredeti adatsor és az „önmagához” képest i-vel eltolt adatsor között. Az autokorrelációt a különböző transzformált adtasorokon a következőképpen számolhatjuk ki:

$$AC(k) = \frac{\sum_{i=1}^{n-k}(Dxdy(i) - \overline{Dxdy}) \cdot (Dxdy(i+k) - \overline{Dxdy})}{\sum_{i=1}^{n}(Dxdy(i) - \overline{Dxdy})^2}$$

Ahol:

- $AC(k)$: az autokorrelációs függvény értéke k eltolással,
- k: az eltolás értéke, $k = 1 \dots n-1$,
- $Dxdy$: a vizsgált differenciált adatsor,
- n: a vizsgált adatsor hossza.

A parciális autokorrelációt nem két változó, hanem két változó és változók egy halmaza között értelmezzük; tartalma: a két változó közti kapcsolat erőssége és iránya, ha a köztük a megadott változókon keresztül terjedő hatásokat kiszűrjük. A parciális autokorreláció függvényt két időpont között számoljuk, kiszűrve a két időpont közötti időpontokat. Más szóval a parciális autokorrelációs függvény i-edik eleme megmutatja a korreláció erősségét az egymástól i intervallumnyi távolságra lévő idősori elemek között, kiszűrve a köztük lévő elemektől való függőségeket. Az előzőek miatt a parciális autokorrelációt az autokorrelációs függvényeken értelmezzük. A számítás során egy parciális autokorrelációs mátrixot kell előállítani, amelynek főátlóban található elemei adják meg a függvény értékeit. A parciális autokorrelációt az alábbiak szerint számolhatjuk ki, adott transzformált idősoron az autokorrelációs függvény segítségével:[14]

$k = 1{,}2$ *esetében*:

$$PAC(1{,}1) = AC(1) \qquad PAC(2{,}2) = \frac{AC(2) - AC(1)^2}{1 - AC(1)^1}$$

$k = 3 \dots k_{max}$ *esetében*:

$$k_{max} = n - 1 \qquad j = 1 \dots k - 1$$

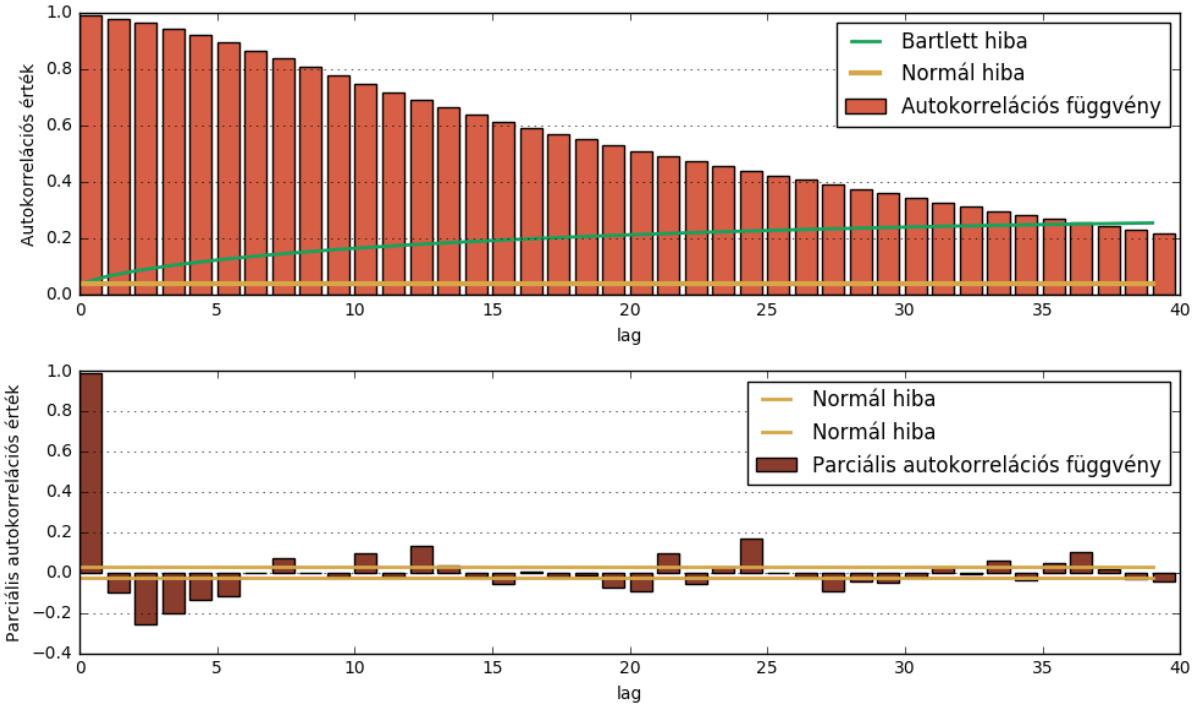
$$PAC(k,j) = PAC(k-1,j) - PAC(k,k) \cdot PAC(k-1,j-1)$$

$$PAC(k,k) = \frac{AC(k) - \sum_{j=1}^{k-1} PAC(k-1,j) \cdot AC(k-j)}{1 - \sum_{j=1}^{k-1} PAC(k-1,j) \cdot AC(j)}$$

Ahol:

- $PAC(k,j)$: a parciális autokorrelációs mátrix k -adik sorának és j -edik oszlopának eleme,
- $PAC(k,k)$: a parciális autokorrelációs függvény értéke k eltolás mellett,
- n: a vizsgált adatsor hossza.

Az alábbi tizenkettedik ábrán, egy havi szinten aggregát adatsor autokorrelációs és parciális autokorrelációs függvényei láthatóak. Az autokorrelációs függvény lassú lecsengést mutat, míg a parciális autokorrelációs függvényen a második elemtől gyakorlatilag nulla közeli értékek láthatóak. Az autokorrelációs függvényből az olvasható ki, hogy a következő időpillanatra a legnagyobb hatással a közeli múlt elemei vannak, míg a parciális autokorrelációs függvényből az olvasható ki, hogy egy eltolás, más néven egy lag esetén van a legnagyobb korreláció. Tehát az év második hónapját elsősorban az első befolyásolja és a harmadik hónapra az első hónap leginkább csak a második hónapra való hatásán keresztül van befolyással.[14]



12. ábra *A korrelációs függvények korrelogramjai*

A hagyományosan alkalmazott autokorrelációs és parciális autokorrelációs függvények mellett, szezonalitás fennállása esetén a szakemberek, gyakran szezonhatárokra szűrten alkalmazzák az előbbiek tárgyalt függvényeket. Ez azt jelenti, hogy a feltételezett szezonhatárok értékeit kiemelik, és a szezonhatárokra szűrt értékekből képeznek új függvényt, majd ezen végzik el a korrelációs teszteket.[14]

Az autokorrelációs és parciális autokorrelációs függvények előállítását követően a hibahatárok előállítása következik, amelyek alapján a későbbiek folyamán következtetéseket tudunk levonni a helyes modell paramétereiről, valamint az idősorban megbújó jelenségekről. Amennyiben egy adott függvényérték meghaladja a hibahatárt, akkor arra azt mondjuk, hogy szignifikáns. Ha a vizsgált függvényben nem található szignifikáns érték, akkor azt fehér zajnak minősítjük, ami egyben azt is jelenti, hogy a vizsgált adatsorból nem tudunk alkalmazható modell paramétereket megállapítani. Autokorrelációs függvények esetében a Bartlett hibát és normál hibát egyaránt szokták alkalmazni, míg a parciális autokorreláció esetében kizárólag csak a normál hibahatárt szokták kiszámolni, amelyeket az alábbi egyenletek szerint lehet kikalkulálni:[14]

Bartlett hiba:

$$FH\_BA(k) = MS \cdot \sqrt{\frac{1}{n} \cdot (1 + 2 \cdot \sum_{i=1}^{k-1} AC(i)^2)}$$

Ahol:

- $FH\_BA(k)$: a statisztikai hiba felső határa, Bartlett hiba szerint,
- $AH\_BA(k)$: a statisztikai hiba alsó határa, Bartlett hiba szerint,
- n: az adatsor hossza,
- MS:az adatsor szórása.

Normál hiba:

$$FH(k) = \frac{MS}{\sqrt{n}}$$

$$AH(k) = -FH(k)$$

Ahol:

- FH: a statisztikai hiba felső határa, normál hiba szerint,
- AH: a statisztikai hiba alsó határa, normál hiba szerint
- n: az adatsor hossza,
- MS:az adatsor szórása.

A Bartlett hibát alapvetően azért szokták preferálni, mivel figyelembe veszi, hogy az eltolás növelésével csökken a statisztikai megbízhatósága a rendelkezésre álló adatoknak.

A különböző differenciált adatsorok statisztikai függvényeinek előállítását követően az identifikációs tesztek végrehajtása következik az autokorrelációs és parciális autokorrelációs függvényeken. Az adatsorokon a minták azonosítását, szezon teszteket, valamint levágási teszteket kell végrehajtani.[14]

A minták azonosítás során a célunk a szignifikáns értékek által fölvett minta meghatározása, a szakirodalom által javasolt szélsőértékek segítségével. A minták azonosítása kétféle teszt alkalmazása útján történhet, amelyek nem különben könnyen eltérő eredményre vezethetnek. Az egyik eljárás során a lag figyelembevétele mellett vizsgáljuk az utolsó szignifikáns érték elhelyezkedését. Az adatsor tulajdonságairól ebben az esetben a szakirodalom által meghatározott minimális és maximális lag értékek segítségével tudunk következtetéseket levonni. Az azonosítást az alábbi második táblázat segítségével tudjuk elvégezni.[14]

2. táblázat *Pattern teszt segédtáblázat lag számolás esetén*

Mintázat típusa	Meghatározott lag értéke
WN	0
CUT	lag≤lag_min
SLW	lag≥lag_max
EXP	lag_min<lag<lag_max

$lag\_min_{max} = 3$ $lag\_max_{min} = 10$

A teszt során azt vizsgáljuk, hogy hol található a különböző korrelációs függvények segítségével előállított adatsorokban az utolsó szignifikáns érték. Bár a szakirodalom tesz javaslatot a lag minimális és maximális értékére, fontos megjegyezni, hogy ezek nem általános érvényességűek, nagyságuk tapasztalati úton került meghatározásra, nem egzakt analitikus úton.[14]

A másik lehetséges minta azonosítási eljárás, a korrelációs értékek átlagos változásának, vizsgálata útján történik. A vizsgálat logikája hasonló a lag számolás metodikájához, azonban itt a korrelációs értékek változásának átlagát vesszük a minta azonosítás alapjául, a szakirodalom által javasolt delta változó lehetséges minimális és maximális értékének figyelembevétele mellett. Az azonosítást az alábbi harmadik táblázat segítségével tudjuk elvégezni.[14]

3. táblázat *Pattern teszt segédtáblázat a korrelációs értékek átlagos változásának vizsgálatához*

Mintázat típusa	Meghatározott lag értéke
WN	0
SLW	delta≤delta_min
CUT	delta≥delta_max
EXP	delta_min<delta<delta_max

$delta\_min_{max} = 10\%$ $delta\_max_{min} = 65\%$

Fontos megjegyezni, hogy az itt használatos delta minimum és maximum értékek sem általános érvényességűek, hasonlóan a lag számítás esetéhez.

A mintázatok azonosítását követően a szezonalitás vizsgálata következik, a feltételezett szezon periodicitás alapján. Ezeket a teszteket csak az autokorrelációs függvényeken kell elvégezni, mivel szezonalitás szempontjából ezekből lehet mérvadó következtetéseket levonni. A szezonalitás eldöntése egy tudás bázis és a vélt szezonhatárokon történő szignifikancia erősségének vizsgálatával történik. A szezonhatárok maximális száma az adatsorban az alábbiak szerint határozható meg:[14]

$$n_{max} = (k_{max} - 1)/s$$

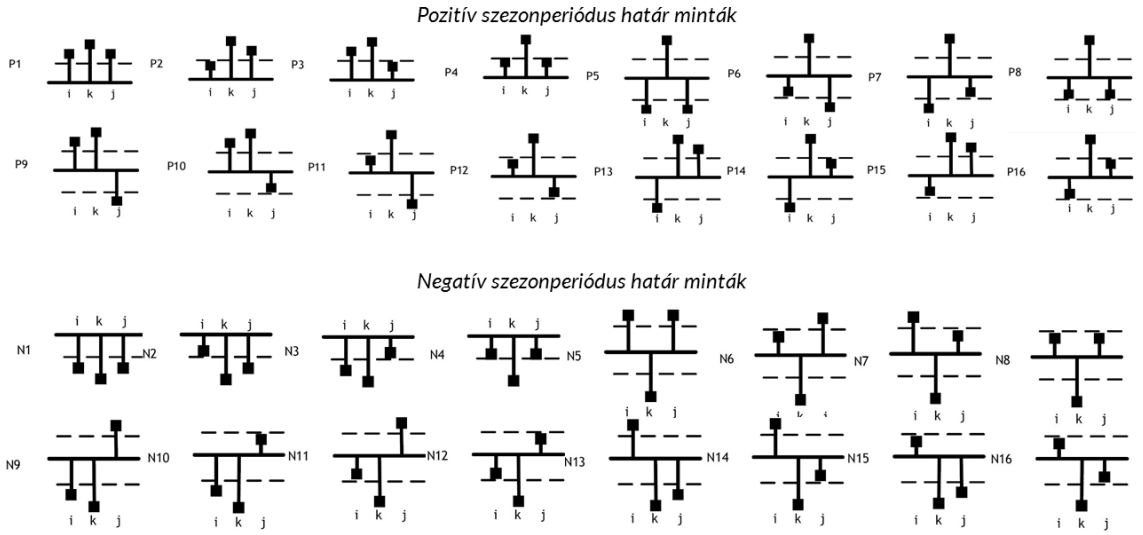
Ahol:

- n_{max}: a szezonhatárok maximális száma,
- k_{max}: a lehetséges eltolás felső határa,
- s: a feltételezett szezon periódus.

A vizsgálatokat minden szezonhatáron el kell végezni, vagyis matematikaformába öntve, mindenhol ahol teljesül az alábbi formula:

$$k \bmod s = 0$$

Első lépésben a vélt szezonhatárokat és azok közvetlen környezetét megvizsgálom, hogy felfedezhető-e hasonlóság az adatsor mintájában illetve az alábbi tizenharmadik ábrán látható tudásbázis mintáiban.[14]



13. ábra *Az autokorrelációs függvényben azonosítható szezonminták*

Ezt követi a szignifikancia erősségét vizsgáló teszt. A teszt során mesterségesen előállítunk mesterséges autokorrelációs értékeket és ezeket össze kell vetni a tényleges értékekkel és ebből állítható elő a szignifikancia erősségének értéke. A számításokat az alábbi módon lehet elvégezni:

$$SS(k) = abs(AC(k)/COT(k))$$

$$COT(k) = e^{-bk}$$

$$b = -\ln(FH)\,/(n_{max} \cdot s)$$

Ahol:

- $SS(k)$: az autokorrelációs függvény k –adik elemének a szignifikancia erőssége,
- $AC(k)$: az autokorrelációs függvény k-adik értéke,
- $COT(k)$: a mesterségesen előállított autokorrelációs függvény k-adik értéke,
- b: a mesterségesen előállított autokorrelációs függvény paramétere,
- FH: a normál hiba felső határa,
- n_{max}: szezonhatárok maximális száma,
- s: szezon periódus értéke.

A szignifikancia szint erőssége nagyobb értéket is felvehet, mint 100 százalék, azonban a lag értéket növelve, jellemzően csökken a szóban forgó változó által felvett érték. A szakirodalom azt javasolja, hogy az elfogadási határértéke a szignifikancia erősségének a meghatározott helyeken legalább 60-80 százaléknak kell lennie. Azt, hogy mely szezonhatárokon szükséges a tudásbázis alapján mintát azonosítani, nagyban függ az adatsor hosszától. A részletes értékeket

az alábbi negyedik táblázat tartalmazza. Fontos megjegyezni, hogy a szezon teszteket csak a D0d0, D0d1 és D0d2 adatsorokon végezzük el. A táblázatban a TSB a tudásbázisra utal, a k paraméter pedig értelemszerűen az eltolást határozza meg.[14]

4. táblázat *Segédtáblázat a szezonalitás eldöntéséhez*

Idősor hossza	k							
	12		24		36		48	
	TSB	SS	TSB	SS	TSB	SS	TSB	SS
48	Px v. Ny	-	-	-	-	-	-	-
60	Px v. Ny	≥60...80%	Px v. Ny	-	-	-	-	-
72	Px v. Ny	≥60...80%	Px v. Ny	-	Px v. Ny	-	-	-
≥84	Px v. Ny	≥60...80%	Px v. Ny	≥60...80%	Px v. Ny	-	Px v. Ny	-

Amennyiben a szezonalitás tényét sikeresen megállapítjuk a fenti táblázat irányelvei alapján, akkor speciális a transzformált adatsorok autokorrelációs és parciális autokorrelációs függvényeiből leválasztjuk, az első szezon periódust előállítjuk az SP, illetve a szezonhatárok elemeit kiemelve és harmonikaszerűen összetolva az értékeket, megszerkesztjük az SB adatsorokat. Az újonnan konstruált függvényeken el kell ezt követően végezni a korábban tárgyalt pattern tesztet, illetve az alábbiakban taglalt levágási, idegen kifejezéssel élve cut off tesztet. Fontos kiemelni, hogy a levágási tesztet nem csak a szezon hatás igazolása esetén kell elvégezni, azonban ha az adatsor nem bír szezonális jelleggel, elégséges csak a D0d0, D0d1 és D0d2 adatsorokon elvégezni.[14]

A cut off teszt során a levágási küszöbértékek számítjuk az első két releváns korrelációs indexre, vagyis k=1,2 helyeken, mind az autokorrelációs és parciális autokorrelációs függvények esetében, amely igaz a speciális SP és SB adatsorokra egyaránt. A levágási küszöbértékeket az alábbiak szerint számolhatjuk:[14]

Autokorrelációs függvény esetén:

$$CT\_LAG(k) = \frac{\log(FH(k))}{log|AC(k)|}$$

Parciális autokorrelációs függvény esetén:

$$CT\_LAG(k) = \frac{\log(FH(k))}{log|PAC(k)|}$$

Ahol:

- $CT\_LAG(k)$: a k-adik releváns tag levágási küszöbértéke,
- $FH(k)$: a normálhiba felsőhatárának értéke a k-adik releváns helyen,
- $AC(k)$: az autokorrelációs függvény k-adik releváns tagja,
- $PAC(k)$: az autokorrelációs függvény k-adik releváns tagja.

Az identifikáció szempontjából ez azért is kritikus fázis mivel az ARIMA, SARIMA modellek esetében az AR és MA tagok megállapítása ebben a fázisban történik. Az egyszerűbb ARIMA modellek esetében a vizsgált adatsor autokorrelációs függvényéből az MA tag mélységét határozzuk meg, míg a parciális autokorrelációs függvényekből pedig az AR tag kerül

meghatározásra. Szezon hatás esetében az SP adatsoron történő levágási küszöbértékekből a SARIMA modell nem szezonális részének tagjai kerülnek identifikációra, míg az SB soron a szezonális rész AR és MA tagjait határozzuk meg. Az integrált tagot minden esetben a differenciálás nagysága határozza meg. A tagokhoz úgy jutunk, hogy az adott korrelációs függvényből kapott két küszöbértéket lefelé kerekítjük, és a nagyobbat vesszük alapul. A szakirodalom azt javasolja, hogy amennyiben valamelyik modell érték nagyobbra adódna, mint kettő, akkor a kettőt tanácsos alapul venni, mivel a modell mélységének növelésével csökken az általánosító képesség és romlik a predikció pontossága. A modell struktúrájának azonosítását a pattern teszt az alábbi ötödik táblázat alapján tudja segíteni. Amennyiben valamelyik korrelációs függvénye a vizsgált adatsornak „slow" típusú vagy fehérzaj tulajdonságot mutat, úgy az identifikáció nem végezhető el. Fontos kiemelni ismét, hogy a slow jelleg megbújó trendre vagy szezonalitásra utalhat.[14]

5. táblázat *Segédtáblázat a modell identifikációhoz*

ACF	PACF	Identifikáció	Megjegyzés
EXP	EXP	AR(I)MA (p,q)	
CUT	CUT	AR(p) vagy MA(q)	A jobban "cut-oló" érvényesül
EXP	CUT	AR(p)	
CUT	EXP	MA(q)	

2.1.3. Modell paraméterezés

Abban az esetben, ha az identifikáció során nem sikerült azonosítani, valamilyen ARIMA, vagy SARIMA modellt, még mindig van lehetőség könnyebb modellek alkalmazására, mint például a Holt, Winter módszer, vagy éppenséggel valamilyen simító eljárás. Ezeknek az úgynevezett könnyű eljárásokat a szakirodalom, rendkívül jól körül járta, és alkalmazásukhoz nem szükséges identifikációt elvégezni. A dolgozat a keretei miatt ezek részletezésére nem tér ki a fennálló tartalmi korlátok végett.[14]

Amennyiben sikerült az identifikáció során a modell mélységeket megállapítani, úgy az operátor egyenlet megfelelő helyeire be kell helyettesítenünk a megfelelő paramétereket, majd a matematika szabályai szerint ki kell fejteni így hozzájutva az optimalizálandó generátor egyenlethez.[14]

Az ARIMA modellek operátor egyenlete:

$$\varphi_p(B) \cdot \nabla^d \cdot Y(t) = \vartheta_q(B) \cdot e(t)$$

A SARIMA modellek operátor egyenlet:

$$\varphi_p(B) \cdot \psi_p(B^s) \cdot \nabla^d \cdot \nabla_s^D \cdot Y(t) = \vartheta_q(B) \cdot \theta_Q(B^s) \cdot e(t)$$

Ahol:

- φ_p:autoregresszív polinom,
- B: az eltolás operátor,
- ∇^d:normál differencia tag

- $Y(t)$: az eredeti idősor,
- ϑ_q: mozgóátlag polinom,
- $e(t)$: a reziduálsor,
- ψ_P: szezonális autoregresszív polinom,
- s: szezonperiódus,
- ∇_s^D: szezonális differencia,
- θ_Q: szezonális mozgóátlag polinom.

Az eltolás operátornak a modellek felírásában igencsak nagy szerepe van, hiszen ezzel tudjuk megállapítani, hogy az idősor mely múltbeli tagjából tudunk a jövőbeli alakulására következtetni. Az alábbi egyenletek bemutatják, a hatványkitevők és az eltolás operátor szerepét az egyenletrendszerek felírásánál:[14]

$$B \cdot Y(t) = Y(t-1)$$

$$B^s \cdot Y(t) = Y(t-s)$$

$$B \cdot e(t) = e(t-1)$$

Minden egyes ARIMA modell kifejtésénél matematikailag (algebrailag) helyesen kell a szorzásokat végrehajtani az operátor egyenletben. Az alábbi példán a legbonyolultabb, legtöbb tagból álló ARIMA (2, 2, 2) modell esetében mutatjuk be az operátor egyenlet használatát:[14]

$$\varphi_2(B) \cdot \nabla^2 \cdot Y(t) = \vartheta_2(B) \cdot e(t)$$

A generátor egyenlet:

$$F(t) = Y(t) - e(t) =$$
$$\varphi_1 \cdot [Y(t-1) - 2 \cdot Y(t-2) + Y(t-3)] +$$
$$\varphi_2 \cdot [Y(t-2) - 2 \cdot Y(t-3) + Y(t-4)] +$$
$$2 \cdot Y(t-1) - \vartheta_1 \cdot e(t-1) - \vartheta_2 \cdot e(t-2)$$

A modell hatékony működése végett a paramétereket a vizsgált idősor sajátosságaihoz kell igazítani, ami egy optimalizálási feladat. Ahhoz, hogy ezt elvégezhessük, az adatsort fel kell bontani egy tanuló és teszt állományra, hasonlóan, mint a neurális hálók esetében. A tanulóállományon valamilyen optimalizációs eljárás segítségével, megpróbáljuk a modell paramétereit olyan módon változtatni, hogy az optimalizálási célkritériumunk a lehető legkisebb értéket vegye fel. A célkritériumunk jellemzően egy olyan hibamutatót, amely a modell által adott predikció és a tényleges adat között méri a hibának valamilyen formáját. A modell helyességét ezt követően a teszt állományon ellenőrizzük, és a legjobb eredményt adó modellt választjuk ki, az identifikált és paraméteroptimalizált függvények közül. A paraméter együttesek megállapításához általában, probléma komplexitásából adódóan valamilyen metaheurisztikát szoktak alkalmazni. Az egyenlet együtthatóinak meghatározásakor az alábbi peremfeltételeket kell figyelembe venni, amennyiben a maximális megengedett mélysége a szezonális paramétereket tekintve egy, nem szezonális esetben pedig kettő:[14]

$$|\varphi_1| < 1 \qquad |\varphi_2| < 1$$

$$\varphi_1+\varphi_2 < 1 \qquad \varphi_2-\varphi_1 < 1$$

$$\varphi_2 - 1 < \varphi_1 < 1 - \varphi_2$$

$$|\vartheta_1| < 1 \qquad |\vartheta_2| < 1$$

$$\vartheta_1+\vartheta_2 < 1 \qquad \vartheta_2-\vartheta_1 < 1$$

$$\vartheta_2 - 1 < \vartheta_1 < 1 - \vartheta_2$$

$$|\psi_1| < 1$$

$$|\theta_1| < 1$$

2.1.4. Az előrejelzés mérőszámai

Nostradamus szerint: „A múlt megsúgja a jövőt, mert a jövő ismétli a múltat!" Bár a mondat igazságtartalma kétségtelenül vitathatatlan, amennyiben előrejelzéseket készítünk, jószerivel csak abban lehetünk biztosak, hogy a predikció a valós értéktől eltérést fog mutatni. A modellezés során, annak érdekében, hogy előrejelző rendszerünk pontosságát fel tudjuk mérni, az eltéréseket jellemezni kell. Erre szolgálnak a modellezés és az előrejelzési hibák mérőszámai.[14]

Első lépésben meg kell bizonyosodnunk arról, hogy az előrejelzési modellünk jól működik, legyen szó akár hagyományos modellről, akár mesterséges intelligenciáról. Ezt olyan módon tehetjük meg, hogy a valós és az előrejelzett adatokból képzett reziduálsoron fehérzaj tesztet hajtunk végre. Ennek lényege, hogy a hibasornak előállítjuk az autokorrelációs függvényét és megvizsgáljuk, hogy az értékek között találunk-e szignifikánst. A grafikonok felrajzolásának előnye, hogy látványos, könnyű értelmezni, azonban gyakran előfordul, hogy egy-két érték a kijelölt határon kívülre esik, amiből tévesen azt a következtetést vonhatjuk le, hogy található autokorreláció a maradékokban. Ennek kiküszöbölésére az alábbi statisztikai teszt azt vizsgálja, hogy az első h autokorrelációs érték vajon szignifikánsan különbözik-e egy fehérzaj folyamat értékeitől. Az autokorrelációs értékek egy csoportján elvégzett tesztet portmanteau tesztnek hívjuk. Az egyik ilyen teszt a Box-Pierce teszt és a hozzá hasonló, ám annál akkurátusabb Ljung-Box teszt. A kiszámolásukat az alábbi képletek mutatják be:[14]

$$Box\ Pierce: Q = T \cdot \sum_{k=1}^{h} r_k^2 \qquad Ljun - Box: Q^* = T(T+2) \cdot \sum_{k=1}^{h} (T-k)^{-1} \cdot r_k^2$$

Ahol:

- h: a maximum visszaléptetés, ameddig az autokorrelációs kapcsolatot vizsgáljuk,
- T: a megfigyelések száma,
- r_k: az autokorreláció értéke a k-adik visszaléptetésnél.

Amennyiben az r_k értékek közel esnek nullához (alacsony az autokorreláció), úgy négyzetes értékeiknek összege is nagyon kicsi lesz. Amennyiben néhány r_k érték jelentősen eltér 0-tól (akár pozitív, akár negatív irányba), úgy Q és Q^*értéke magas lesz. A h érték megválasztását befolyásolja, hogy szezonális idősorról beszélünk-e, ha nem h értékének 10 javasolt, ha igen a

periódusok számának kétszerese, azaz 2s, alapszabály azonban, hogy ne legyen nagyobb a megfigyelések számának ötödénél. Annak eldöntése érdekében, hogy Q vagy Q^* értékek nagynak számítanak-e, megvizsgáljuk, hogy χ^2 próbát hajtunk végre, (h-K) szabadságfokon, ahol K a modell paramétereinek száma, amennyiben azonban nyersadatokból kerültek számításra K=0 –át feltételezünk. Amennyiben adott szignifikancia szint mellett kisebb a χ^2 érték az elméleti értéknél úgy a maradvány értékek közötti autokorreláció nem különbözik jelentősen egy fehérzaj folyamattól, jól becsültük a modell paramétereket.[14]

Azonban a fentiek alkalmazásával csak arról tudunk megállapítást tenni, hogy az alkalmazott modell mentes-e a szisztematikus hibáktól, az előrejelzés pontosságáról szinte semmit nem árul el a fehérzaj teszt. [14]

A vizsgálatok során a leggyakrabban használt mérőszámokat a devianciából származtatjuk. Ennek oka, hogy segítségükkel gyorsan képet lehet alkotni a modell pontosságáról, valamint megmutatják mekkora változékonyság. A modellek hibájának vizsgálatakor leggyakrabban a reziduálsor mediánját és várható értékét vesszük alapul a mutatószámok kiszámításához. [14]

Amikor egy helyi vegyesbolt kenyér eladását becsüljük meg kilóban és az átlagos négyzetes eltérés például 5 kg lesz, arról nem kapunk információt, hogy ez vajon az adott helyzetben soknak vagy kevésnek számít, éppen emiatt szükséges a mutatókat arányosítani, ezáltal pontosabb képet kaphatunk a modell jóságáról. A legfontosabb devianciából származtatott és arányosított mutatókat az alábbi hatodik táblázat foglalja össze.[14]

6. táblázat *Devianciából származtatott és arányosított mutatók*

Típus	Rövidítés	Megnevezés	Képlet	Megjegyzés
Devianciából származtatott mutatók	MAE	Az abszlút hiba átlag	$\frac{1}{n}\cdot\sum_{t=1}^{n}\lvert e_t\rvert$	
	MdAE	Az abszolút hiba mediánja	$medián(\lvert e_t\rvert)$	
	MSE	Az átlagos hiba négyzete	$\frac{1}{n}\cdot\sum_{t=1}^{n}{e_t}^2$	
	RMSE	Az átlagos hiba átlagának gyöke	$\sqrt{MSE}$	
Arányosított mutatók	MAPE	abszolút százaléákos hiba átlaga	$\frac{1}{n}\cdot\sum_{t=1}^{n}\lvert p_t\rvert$	$p_t=\frac{e_t}{Y_t}\cdot 100$
	MdAPE	abszolút százaléákos hiba mediánja	$medián(p_t)$	
	sMAPE	abszolút százalékos hiba szimetrikus átlaga	$\frac{1}{n}\cdot\sum_{t=1}^{n}\frac{\lvert e_t\rvert}{Y_t+F_t}\cdot 200$	F_t a t-edik periódus előrejelzése
	sMdAPE	abszolút százalékos hiba szimetrikus mediánja	$medián(\frac{\lvert e_t\rvert}{Y_t+F_t}\cdot 200)$	
	RMSPE	négyzetes százalékos hiba átlagának négyzetgyöke	$\frac{1}{n}\cdot\sum_{t=1}^{n}\sqrt{p_t^2}$	
	RMdSPE	négyzetes százalékos hiba mediánjának négyzetgyöke	$\sqrt{medián(p_t)^2}$	

A relatív és a skálázott mutatószámok segítségével információt kaphatunk a modell reakció képességéről. Azt vizsgáljuk, hogy a hiba mértéke, mekkora keresleti érték változásához képest. Előfordulhat, hogy a modell nem képes kellő gyorsasággal lekövetni az idősorban fellépő változásokat és csak gyenge látenciával követi a tényleges adatsort. A relatív és skálázott mutatók segítségével a modell adaptivitásáról alkothatunk képet.[14]

További fontos mutatószámcsoport az információs kritériumok alapján képzett indexek, amelyeknek a célja a mintában lévő információ felhasználásának maximalizálása. Büntetik a hibák négyzetösszegének csökkenését az újabb paraméterek bevonásával. Minél alacsonyabb értéket vesznek fel ezek a mutatók, annál jobb a modell. A legfontosabb relativizált mutatószámokat és információs kritériumokat az alábbi hetedik táblázat tartalmazza.[14]

7. táblázat *Relatív mutatók és információs kritériumok*

Típus	Rövidítés	Megnevezés	Képlet	Megjegyzés
Relatív és skálázott mutatók	MASE	abszolút skálázott hiba átlaga	$\frac{1}{n}\cdot\sum_{t=1}^{n}\lvert q_t\rvert$	$q_t=\frac{e_t}{\frac{1}{n-1}\cdot\sum_{t=2}^{n}\lvert e_t^*\rvert}$
	MRAE	relatív abszolút hiba átlaga	$\frac{1}{n}\cdot\sum_{t=1}^{n}\lvert r_t\rvert$	$r_t=\frac{e_t}{e_t^*}$
	MdRAE	relatív abszolút hiba mediánja	$medián(\lvert r_t\rvert)$	
Információs kritériumok	AIC	Akaike információs kritérium	$n\cdot\ln(MSE)+2\cdot k$	k a modell paramétereinek száma
	BIC	Bayesian információs kritérium	$n\cdot\ln(MSE)+k\cdot\ln(n)$	
	nBIC	Normalizált Bayesian információs kritérium	$\ln(MSE)+\frac{k\cdot\ln(n)}{n}$	
	FPE	Final prediction error	$\frac{1+k/n}{1-k/n}\cdot VE$	$VE=\frac{1}{n}\cdot\sum_{t=1}^{n}(e_t-ME)^2$
*Megjegyzés:			$ME=\frac{1}{n}\cdot\sum_{t=1}^{n}e_t$	$e_t^*=Y_t-Y_{t-1}$

2.1.5. A finomtervezés

A finomtervezés során beépítésre kerülnek mindazon idősori jellegzetességek, illetve tulajdonságok, melyeket a kereslettervezés első fázisa során leszűrtünk az adatsorról, nevezetesen a pozitív és negatív kiugrásokat, amelyek visszavezethetőek valamely egyéb paraméter befolyására (hőmérséklet, promóció, tőzsdei és/vagy piaci folyamatok, stb.). A finomtervezés célja az előrejelzés pontosságának fokozása, a járulékos információk, torzító hatásának feltárása és érvényességre juttatása. Fontos kiemelni, hogy a tényezők együttes hatását rendkívül nehéz mérni.[14]

Ebben a fázisban történik az adatok dezaggregációja. Az előrejelzések kritériumok menti lebontásával kapcsolatban számos kérdés merül fel. A megoszlás jövőjét sokféleképpen meg lehet jósolni, azonban nincsen általánosan elfogadott módszer. Előfordul, hogy a múltbéli arányok alapján kerül szétbontásra a kereslet, vagy pedig az arányokat próbáljuk megjósolni. Az arányok megjóslásának esetében általában valamilyen könnyű módszert szoktak

alkalmazni. További kérdés a dezaggregáció során, hogy milyen sorrendben történik (pl. előbb terület, majd idő, azt követően értékesítési csatorna és végül termékek szerint).[14]

A finomtervezés során fontos elkülöníteni egymástól a pontszerű, valamint az időszakos hatásokat. Az előrejelzésnek, ennek a fázisában nagy hangsúlyt kap az intuíció. A finomtervezés során a szakemberek legtöbbször korrelációs és egyéb statisztikai eszközöket használnak fel. A kereslettervezés azonban nem fejeződik az iparban a finomtervezés lefolytatásával. A folyamat szerves részét képezi a modellek utókövetése a fentebb tárgyalt mérőszámok segítségével, azonban erre a dolgozat keretei miatt nem térek ki. [14]

2.2. A neurális hálók szerepe kereslet előrejelzésben

Ahogy a fentiekben láthattuk a hagyományos kereslet előrejelzés folyamata rendkívül időigényes és az egymást követő lépések során számtalanszor felmerülhet több, komolyabb döntési helyzet, amely során nem lehet deklarált szabályokra hagyatkozni, hanem a szakértő belátására van bízva a megfelelő irány kiválasztása az előrejelzés szempontjából. Ezeknek az elhatározásoknak a meghozatalában rejlik a hagyományos kereslettervezésnek a sikeressége. Az ilyen szituációk jó megoldásához elengedhetetlen, hogy a döntéshozó megfelelő tapasztalattal bírjon, ez azonban csak hosszú évek munkájának útján érhető el. A fent említett nehézségek kiküszöbölésére és megoldásának támogatására célszerű lehet megfontolni a mesterséges intelligenciák alkalmazását.

2.2.1. Adatelőkészítés mesterséges intelligenciákkal

Alapesetben, ahhoz egy rendkívül zajos adatsort valamilyen szűrő algoritmussal meg tudjunk tisztítani a torzító hatásoktól és kiugró értékektől, szükségünk van valamilyen apriori információra az adatsor zajának tulajdonságait illetően annak érdekében, hogy alkalmazni tudjuk a megfelelő alul-, felül-áteresztő, vagy Kálmán-szűrőt. Azonban egy megfelelően betanított neurális-szűrő esetében ilyenre nincsen szükség. A rendszer architektúrája magában és dinamikus állapota magában foglalja mindazon információk összeségét, amivel a legpontosabb szűrési teljesítmény elérhető.[15]

A neurális szűrők problémaköre alapvetően jól feltárt tudományos ágazat. Módszertan szempontjából alapvetően kéz megközelítést szoktak használni, szimulált adatok útján való tanítás, vagy pedig valós adatokon való tanítás. Az előbbi azonban csak akkor lehetséges, ha ismerjük a rendszer modelljét. Az optimális szűrés kérdés körét az 1990-es évek elején megoldották visszacsatolt hálók segítségével. A metódus lényege, hogy a meglévő adatokat egy rekurzív szűrőn keresztül szintetizálják, amelynek kimenete optimális becslést ad a zajoktól megtisztított jelre egy előre meghatározott pontossággal, a becslési kritérium alapján.[15]

A leggyakrabban olyan struktúrákat valósítanak meg neurális szűrők esetében, amelyekben rétegen belül összekapcsolt perceptronok vannak, vagy egyszerű kimenti, visszacsatolással bírnak. Az előbbiek nagyobb népszerűségnek örvendenek, mivel általában kevesebb processzáló elemmel is elérhető a szükséges szűrési pontosság. A szakirodalom kiemeli, hogy mivel az alkalmazási területek nagy többsége igen magas pontosságot követel meg, a meghatározott hiba mutatók elérésének érdekében, nagy tanulóállományra van szükség.

Amennyiben ez nem lehetséges, úgy még további lehetőséget biztosít az adatok megfelelő elő- és utófeldolgozás az általánosítóképesség biztosításának érdekében.[15]

A fentebb tárgyalt módszer egy lehetőséget biztosít, az adatelőkészítési részfolyamat támogatására, azonban alkalmazása több kérdést vett fel. Alapvetően, ha figyelembe vesszük a ma uralkodó vállalati trendeket kijelenthető, hogy kevés esetben állna rendelkezésre olyan mennyiségű adat, hogy megfelelő neurális szűrőt lehessen készíteni. Bár lehetséges, hogy a meglévő adatok statisztikai elemzése alapján fel lehessen állítani egy matematikailag helyes szimulációt és abból többszöri futtatás eredményeként megkapjuk a megfelelő tanulóállományt, azonban ekkor felmerül a kérdés, hogy vajon megéri-e, hiszen a matematikai modell ismeretében már ki lehet választani a megfelelő szűrő algoritmust.

A másik lehetséges megoldás, hogy előzetesen megvizsgáljuk, hogy milyen komponensek milyen mértékben befolyásolják az idősor alakulását és ez alapján felépítünk egy tanulóállományt. Egy komplex rendszer elméletileg képes lehet a befolyásoló jellemzők megtanulására és eredményül visszaadni a torzításoktól mentes idősort. Itt azonban ismét csak felmerül a kérdés, hogy vajon rendelkezésre áll-e minden szükséges adat egy ilyen rendszer megépítéséhez egy átlagos vállalatnál.

2.2.2. Identifikáció mesterséges intelligenciákkal

A hagyományos kereslettervezés legnehezebb része a modell identifikáció. A részfolyamat során megpróbáljuk kinyerni az adatsorban rejlő mintákat, sajátosságokat és esetleges szezonhatárokat kinyerni különböző statisztikai tesztek segítségével. Az identifikáció eredményül a potenciálisan alkalmazható hagyományos modellek egy listáját adja, amelyből a kereslettervezőnek kell kiválasztania, az adatsorról meglévő információ és szakértői tudása alapján a legmegfelelőbbnek vélt eljárást. Könnyen körvonalazható, hogy ez a fázis szorul leginkább támogatásra hagyományos előrejelző modellek konstruálása során.

A problémát jobban körül járva könnyen megállapítható, hogy egy klasszifikációs problémával állunk szemben. A hagyományos klasszifikációs problémáról van, aminek esetében összesen 216 lehetséges kategóriába lehet besorolni egy idősort. A probléma megoldását kétféleképpen lehet megközelíteni: vizuális, valamint számszaki szempontokból. A korrelációs függvények előállítása azonban mindkét esetben megkerülhetetlen.

Amennyiben a vizuális utat választjuk a mesterséges intelligencia kiindulási alapját a korrelogramok képzik. Ekkor egy olyan adatbázist kell felépíteni, amely megfelelő fundamentumot teremt arra, hogy a háló kellően pontosan megtanulja a háló a klasszifikáció szabályait. A képfelismerés és feldolgozás során jellemzően konvolúciós hálókat szoktak alkalmazni, mivel már számtalanszor jól vizsgázott ilyen jellegű problémák megoldásánál. A konvolúciós neurális háló a kétdimenziós adatok, például képek feldolgozására alkalmas. A neurális hálókban a feldolgozás egymásra épülő rétegekben történik, amelyek a képet apró szegmensenként elemzik, és e folyamat tökéletesítését tanulja meg az algoritmus [imagenet]

A másik lehetőség, hogy a korrelogramokat kihagyva, közvetlenül a függvény értékeket használjuk fel a klasszifikáció elvégzésére. Ezt a feladatot meg lehet oldani egy hagyományos

MLP-vel. A háló bemenetét ebben az esetben a transzformált adatsorokból származtatott korrelációs függvények képzik.

A problémát, illetve az akadályt mind a két esetben a tanuló adatbázis előállítása jelenti, hiszen ez többezer idősor teljesen pontos identifikációjának elvégzését jelenti, hiszen maguknak a kategóriáknak a számossága egy olyan komplexitású problémát eredményez, ami megköveteli a kategóriánként többszázas nagyságrendű tanulóállományt. Vitán felül áll, hogy egy ilyen rendszer létrehozása rendkívül hasznos lehetne, azonban a tanuló adatbázis létrehozása nehezen leküzdhető akadályokat támaszt.[16]

Bár tisztán kivehető, hogy a teljes identifikáció folyamatát nem lehet mesterséges intelligenciák segítségével automatizálni, azonban a döntéshozatalt potenciálisan lehet. Gyakran nehezen eldönthető egy adatsorról, hogy szezonális jellegűe-e, mivel ezt az idősori sajátosságot könnyen elfedheti, zaj vagy akár torzíthatja trend folyamat is. Ezt a fajta problémakört lineáris szeparációként is fel lehet fogni. Ebben az esetben azt kell tudni eldönteni egy adott adatsorról, hogy van-e benne szezonhatás. Az ilyen jellegű problémákat könnyen meg tudja oldani egy MLP, vagy akár egy szupport vektoros gép. Mivel a feladat ebben az esetben inkább hasonlít egy eldöntendő kérdésre, a tanuló adatbázis méreteit sem kell olyan grandiózusra szabni, mindössze 800-1000 közötti adatállomány, amely kellően diverz már elégségesnek bizonyulhat.[17]

Ebben az esetben a bemeneteket pusztán az autokorrelációs függvények értékei képeznék a szezonálisan nem differenciált adatsoroknak. Úgy gondolom, hogy bár ez lehetőség kivitelezhetőségét tekintve kivitelezhető, a megvalósításba fektetendő munka nem áll arányban az eredményekből származtatható előnyökkel.

2.2.3. Előrejelzés mesterséges intelligenciákkal

A gondolat, hogy mesterséges intelligenciákkal jelezzenek előre, gyakorlatilag a 90-es években megjelent, a neurális hálók robbanásszerű térnyerésével. A legelső kísérletek az egyszerű MLP architektúrák előrejelző képességét vizsgálták a hagyományos Box – Jenkins modellekkel összevetve. A kísérletek azonban ekkor még arra vezettek, hogy a konvencionális modellek minden előrejelző mutató esetében jobb eredményre vezetnek. [18]

Ennek oka azonban elsősorban az MLP tulajdonságaiban keresendő, hiszen a struktúra, ahogy korábban is említésre került, nem alkalmas az időbeli függések megtanulására. Azonban visszacsatolt hálók elterjedésével a tendencia, ha nem is megfordulni látszik, egyértelműen kijelenthető, hogy a mesterséges intelligenciák kompetens előrejelző rendszerként kezelendők akár keresleti adatsorokon esetében is. A feladat rendkívüli peremfeltételei és az idősor, mint jelenség sajátosságai miatt a szakirodalom a kereslettervezési feladatok megvalósítására jellemzően az LSTM cellák alkalmazását javasolja. Bizonyítható, hogy az LSTM számos olyan tulajdonsággal bír, amelyeknek köszönhetően felülmúlja az egyszerű visszacsatolt hálókat az idősor előrejelzés feladatokat illetően. A háló típusnak a tesztek alapján bizonyíthatóan meg van az a képessége, hogy nagy időközönként ismétlődő jelenségek szisztematizmusát is megragadja, akár úgyis, hogy köztük több bementi szekvencia is elhelyezkedik. A kísérletek egyértelműen azt mutatják, hogy amennyiben a cella állapotot is bekapcsoljuk a memória kapukba akkor még bonyolult időzítő algoritmusok megtanulására is alkalmassá válik a

mesterséges intelligencia, és egyértelműen a legjobb eredményeket ez a típusú cella mutatja az idősoros előrejelzés feladatoknál.[19]

Alapvetően a hozzáértő szakemberek elsősorban nem keresleti adatok előrejelzésére használják az ilyen jellegű megoldásokat, hanem az alábbiakra:

- erőművek teljesítménye,[20]
- háztartások elektromos fogyasztása,[21]
- tőzsdei adatok alakulása,[22]
- lézer jel időbeli alakulása,[23]
- több változós időbeli repülési modellek. [24]

A fenti cikkek eredményei mindenképpen biztatásra adnak okot és gyakorlati példákkal is validálják a visszacsatolt neurális hálók gyakorlati alkalmazhatóságát. A szakirodalomban azonban nem igazán lehet olyan tanulmányokat találni, amelyek célzottan kereskedelmi, vagy más egyéb kontextusban a logisztikához szorosan köthető idősort próbálna előre jelezni. Ebben az esetben látom a leginkább kézenfekvőnek a neurális hálók alkalmazását a kereslettervezésben, mivel az első fejezetben elhangzottak alkalmassá teszik a keresleti sorok megtanulására, valamint egy működőképes háló felépítése nem igényel különösebb időbefektetést egy tanuló adatbázis létrehozására, mi több az identifikáció fázisát elméletileg meg lehet spórolni, valamint megfelelő dropout alkalmazása esetén robosztussá is tehető a zajjal szemben, valamint a szisztematizmust nélkülöző kiugró jelek komponens összetevője a rendszerben várhatóan nulla lesz. [2]

A fentiek miatt kutatásomat egy olyan irányba javasolom elvinni, amely magában foglalja a kereslettervezés vizsgálatát neurális hálókkal. Cél kísérleti úton igazolni, hogy lehetséges kereskedelmi idősorokat előre jelezni visszacsatolással rendelkező mesterséges intelligenciákkal.

2.2.4. A finomtervezés beépítése a mesterséges intelligenciákkal való előrejelzésbe

A konvencionális előrejelzés esetében egy dimenziós adatok alapján próbálunk következtetéseket tenni az idősor jövőbeli alakulásáról és ezt követően próbáljuk a külső tényezők hatását figyelembe venni és a korrelációs értékek tükrében rászuperponálni a hatásokat az előrejelzésekre. Azonban a neurális hálók alkalmazása lehetőséget nyújt a finomtervezés beépítésére magába az előrejelzésbe, ezáltal egy újabb munkafázist megspórolva a kereslettervezésben.

A módszertan lényege, hogy bemenetként több dimenziós szekvenciákat használunk a háló tanítása és használata során. Neurális háló általánosító képességének köszönhetően képes megragadni a paraméterek közötti összefüggéseket és finomhangolni a rendszer kimenetét az észlelt összefüggések alapján. Fontos azonban kiemelni, hogy a parametrikus komplexitás növelése esetén a futási idő is jócskán megnövekszik, ebben az esetben a szakirodalom komplex statisztikai eszközök alkalmazását javasolja a paraméterek kiemelésére, mint például a PCA.

A főkomponens-analízis vagy főkomponens-elemzés (angolul Principal Component Analysis, rövidítve PCA) egy többváltozós statisztikai eljárás, mely az adatredukciós módszerek közé sorolható, s a faktoranalízis egy speciális esetének tekinthető. Lényege, hogy egy nagy adathalmaz ─ melynek változói kölcsönös kapcsolatban állnak egymással ─ dimenzióit lecsökkentse, miközben a jelen lévő varianciát a lehető legjobban megtartja. Működése felfogható úgy, mint az adat belső struktúrájának feltárása oly módon, hogy az a legjobban magyarázza az adathalmaz szóródását. Ha egy többváltozós adathalmaz egy nagy-dimenziós adattérben koordináták halmazaként ábrázolt, a főkomponens-analízis egy alacsonyabb dimenziójú képet szolgáltathat a felhasználó számára, a leginformatívabb nézőpontból nézve az objektum egy levetítése vagy „árnyéka" által. Ez az első néhány főkomponens felhasználásával történik úgy, hogy a transzformáltadat dimenzióit lecsökkentjük.[25]

Azonban ha az adatok paraméter csökkentése szükséges, ez megvalósítható lineáris autoenkóderekkel, ami a neurális hálók egy olyan típusa, amelyeket nem ellenőrizetten tanítunk, valamint nem alkalmazunk a rétegek kimenetén aktivációs függvényeket. Sok esetben a rejtett rétegekben található súlyokat transzponálással összekötjük. [26]

3. Kereslet előrejelzés mesterséges intelligenciákkal

A kereskedelmi idősorok előrejelzése az ökonometria legégetőbb problémái közé tartozik szinte a tudományág kezdete óta. Az ellátási-elosztási láncban szereplő vállalatok életében alapesetben kulcsszerepet tölt be kereslettervezése, hiszen ez alapján tudják erőforrás szükségleteiket méretezni, valamint a vállalatot felkészíteni a jövőben potenciálisan tapasztalható nehézségekre.

A piacgazdaságban uralkodó trendek egyre nagyobb ütemben és egyre nagyobb dinamikával változnak. A termékek és szolgáltatások bonyolultsága, komplexitása folyamatosan növekszik. Az értékalkotási láncok vertikális és horizontális tagoltsága szintén növekvő trendeket mutat. A piacgazdaság globalizációjának eredményeként a piaci környezet rendkívül gyorsan változik, amik rendkívüli módon megnehezítik a hagyományos előrejelző módszerek hatékony alkalmazását. Az előzőek eredményeként az előkészítési és identifikációs fázisokra egyre kevesebb időt lehet szánni. Továbbá könnyen előfordulhat, hogy az azonosított modellek reakcióképessége nem bizonyul elégségesnek és csak bizonyos látenciával képesek megjósolni az idősor jövőbeli alakulását.

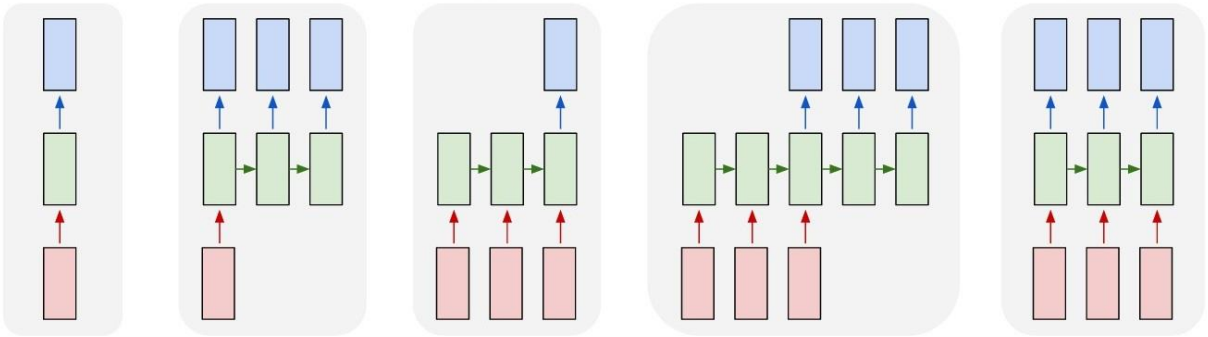
Az előbbi nehézségek kiküszöbölésére kínálhat alternatívát a mesterséges intelligenciákkal történő kereslettervezés. A visszacsatolt neurális hálókat az első fejezetben tárgyalt képességeik potenciálisan alkalmassá teszik olyan kereslet előrejelző rendszerek létrehozására, amelyeket gyorsan fel lehet építeni, valamint kellő rugalmasságot biztosítanak az idősor esszenciájának megragadásához.

3.1. A modellezés lehetséges megközelítései

A visszacsatolással rendelkező neurális hálókat, kiváltképpen az LSTM architektúrát a bemenetek szekvenciális feldolgozásának képessége teszi igazán különlegessé, hiszen ezáltal képes az időbeli függőségeket megtanulni a rendszer. A hagyományos visszacsatolásmentes neurális hálók esetében végtelenül korlátozó tényező, hogy a koncepció a struktúrát egy meghatározott méretű vektor bevételére teszi alkalmassá és a kimeneten eredményül szintén egy adott méretű vektort ad. Ezek a modellek az eredmény leképzését fix számú matematikai művelet elvégzése útján kalkulálják. Ez a struktúra a hagyományos klasszifikációs és regressziós feladatoknál minden kétséget kizáróan megállja a helyét, azonban az időbeli függőséget mutató feladatoknál domborodnak ki igazán a visszacsatolással rendelkező hálók előnyös tulajdonságai.[27]

A visszacsatolás számos struktúra leképzésére biztosít lehetőséget. A megoldandó feladat függvényében a háló dolgozhat szekvenciákkal akár a bemenő és akár a kimenő oldalakon, mi több akár egyszerre mind a két oldalon is. Mivel a neurális hálók alkalmazásának igen nagy szerepe van a képfeldolgozásban, a visszacsatolással bíró hálókat is első körben ezen a területen alkalmazták. A vektoriális bemenetet és szekvenciális kimenetet alkalmazó hálókat elsősorban a kép leírási feladatoknál szokták alkalmazni, vagyis a háló bevesz egy képet és eredményül egy mondatot ad. Szekvenciális bemenettel dolgozó és vektoriális kimenetet adó hálókat, általában szöveg tartalmi feladatoknál szokták alkalmazni. A szekvenciális bemenetekből szekvenciális kimeneteket leképző hálóknak két tipikusan nagy csoportját szokták

megkülönböztetni. Az első ilyen struktúrát elsősorban szövegfordító alkalmazásoknál, illetve chatbotoknál szokták alkalmazni, míg a második architektúrát jellemzően videó klasszifikációs feladatok elvégzéshez szokták felhasználni. A neurális hálók szekvenciákkal és vektorokkal való dolgozásának megértéséhez nyújt némi segítséget az alábbi tizennegyedik ábra. A képen az első modell felépítés a klasszikus csak vektorokkal dolgozó hálókat hivatott ábrázolni. A második struktúra a vektoriális bemenetre szekvenciális kimenetet leképző hálók működését mutatja. A harmadik architektúra a szekvenciális bemenetre vektoriális választ adó hálók szerkezetét mutatja be. Az utolsó két modell konstrukció a szekvenciákból szekvenciákat leképző rendszereket mutatja be. Az első szerkezetet jellemzően a fordító és az ehhez hasonló alkalmazásokban használják fel, míg a másodikat a videó klasszifikáció feladatokban. A piros téglalapok a bemeneteket mutatják, a kék a kimeneteket ábrázolja, a zöld blokkok pedig egyetlen visszacsatolással bíró elem időbeli „kigörgetését” mutatják be.[27]



14. ábra *A visszacsatolt hálók lehetséges struktúrái*

A fent leírt struktúrák számos lehetőséget biztosítanak egy olyan mesterséges intelligencia kiépítésére, amelynek elsődleges célja az idősorok jövőbeli alakulásának megjóslása. Azonban a rendelkezésre álló cikkek és információk alapján nem következik közvetlenül, hogy melyik architektúra a legalkalmasabb egy ilyen rendszer létrehozására.

Amennyiben közvetlenül a hagyományos előrejelző Box-Jenkins, közismertebb nevén ARIMA-SARIMA modelleket vesszük alapul, úgy logikus választásnak tűnhet a fenti ábrán látható harmadik struktúra. A cél ebben az esetben a szekvenciákban meglévő rejtett és nyilvánvaló időbeli összefüggések megtanítása útján adott szekvenciális bemenetből megjósolni azt, hogy az idősor t+1-edik időpillanatban milyen értéket vesz fel. A másik megközelítés alapján a negyedik struktúrát kell felhasználnunk és ekkor a cél egy olyan kimeneti szekvencia előállítása, amely a lehető legpontosabban illeszkedik az idősorra.

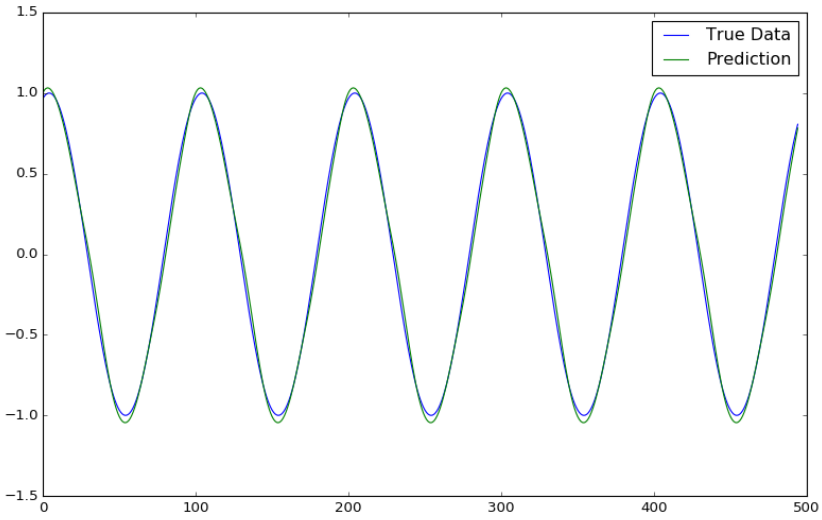
Mivel általános szabály nincs érvényben a neurális hálók modelljeinek építésekor, így a mélységi összefüggések vizsgálatát megelőzően a különböző architektúrák működését próbáltam feltárni tapasztalati úton, hogy elsőkézből nyerjek információt arra vonatkozóan, hogy melyik modell struktúra működhet a legjobban a kereslettervezési feladatok esetében.

3.1.1. Előzetes kísérletek

Az előzetes kísérletek során az első struktúra, amit megvizsgáltam az a szekvenciális bemenetekből vektort leképző architektúra volt. A vizsgálat során a kísérleti mezőt olyan módon próbáltam feltárni, hogy adott egy adott kísérlet végrehajtása során kiválasztottam egy

faktort, amely hatással lehet a végeredményre, majd azt kislépésekben változtatva vizsgáltam a rendszer eredményeit, ezáltal igyekezve tapasztalatot szerezni a struktúra működéséről és az egyes paraméterek fontosságáról. Ezt a struktúrát a Keras program segítségével vizsgáltam. Fontos kiemelni, hogy a Kerasban végrehajtott kísérletek során a tanulási ráta paraméterét a Keras alap beállításain hagytam meg.

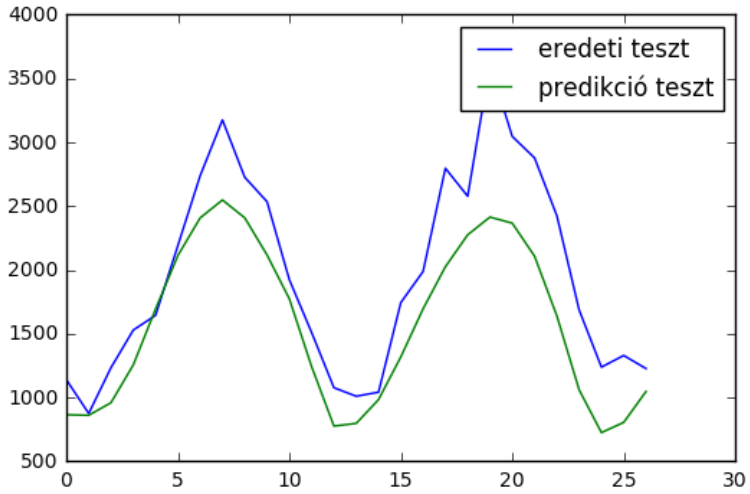
Az első kísérleteket a szinusz hullám megtanítására tettem. A kiindulási alapot egy kettő rejtett réteget tartalmazó háló adta, aminek a kimenetén egy perceptront alkalmaztam lineáris aktivációs függvénnyel. A rejtett rétegek bázis struktúrájának az ötletét az huszonkettedik hivatkozás alapján állítottam fel. Az első rétegben 50 LSTM, másodikban 100 LSTM blokk volt található. A vizsgált szinusz függvény, amelyre a modellt építettem összesen 120 darab elemet tartalmazott. Az adatállomány 90%-át tanításra, a fennmaradó 10%-ot pedig tesztelésre használtam fel. A bemenetet képző időablakok hossza 50 elemből állt. A tanuló batchek méretét pedig 1-re állítottam be. A háló pusztán 5 tanítási iterációt követően is minimális hibával reprodukálni tudta a vizsgált függvényt, ami bizakodásra adott okot. A kísérlet eredményét az alábbi tizenötödik ábra mutatja be. A zölddel jelölt függvény mutatja a predikciót, a kék pedig a tényleges szinusz hullámot. Az előrejelzést 500 elemmel görgettem előre, az 50-edik elemtől kezdődően csak jósolt elemekre támaszkodik a rendszer.



15. ábra *A szinusz hullám tanulási eredménye szekvencia-vektor leképzéssel*

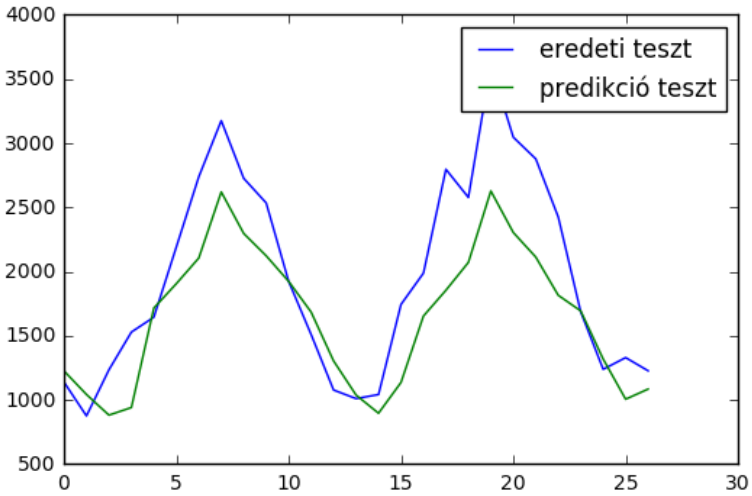
A kezdeti könnyű sikereken felbuzdulva a következő teszteken már valós idősorok vizsgálata útján próbáltam az LSTM hálók mélyebb működéséről képet alkotni. Azonban sajnálatos módon az erős kezdést követően gyors visszaesést tapasztaltam. Az első néhány kísérlet alkalmával először is szomorúan kellett konstatálni, hogy alacsonyabb epoch szám, akár még néhány ezres nagyságrend esetén is a modell képtelen megtanulni az idősor alakulásának szisztematizmusait, függetlenül a rejtett rétegek számától és rejtett rétegekben található processzáló elemektől és ezek aktivációs függvényeitől, valamint a batch-ek méretétől. A teszteket a kereslettervezésben ismert „airline-passenger” adatsorra hajtottam végre. Az adatsort korábban már láthattuk a második fejezetben az tizenegyedik ábrán. A klasszikus kereslettervezésben a szóban forgó adatsorra illeszthető SARIMA$(0,1,1)x(0,1,1)_{12}$ modellt „airline” modellnek hívják. A kísérleti eredmények rendre hasonló jelleget mutattak, mint ami

az alábbi tizenhatodik ábrán látható. A tizenkettedik elemtől a rendszer csak előrejelzett elemekből dolgozik. Az MSE értéke ebben az esetben 38 732 volt.



16. ábra *Kísérlet az „airline-passanger” adatsor megtanulására I.*

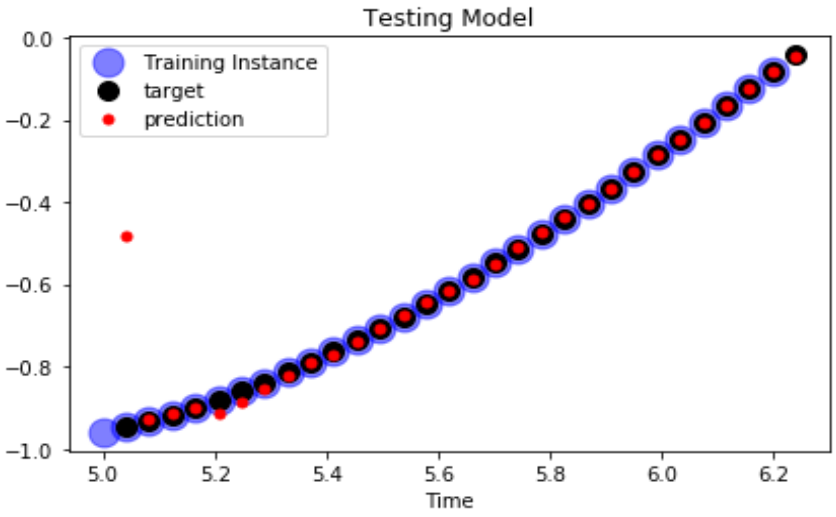
A konzulensemmel közösen egyeztetve, arra a következtetésre jutottunk, hogy meg kell vizsgálni, hogy a rendszer válasza milyen abban az esetben, ha a tanítási iterációk számát megnöveljük egy nagyságrenddel, tehát tízezres nagyságrendű epoch tartományban vizsgáljuk a többi paraméter előrejelzés pontosságára gyakorolt hatásait. A korábbi szinusz függvényre elvégzett teszttel szemben a bemeneti adatok hosszát tizenkettőben határoztuk meg, mivel az adatsorban egyértelműen azonosítható a szezonok tizenkét havonkénti ismétlődése. A tesztek során azt tapasztaltuk, hogy a nagyobb epoch szám, ugyan jobb eredményre vezet, azonban korántsem bizonyulnak olyan ígéretesnek, mint a hagyományos módszertanok által szolgáltatott eredmények. A legjobb előrejelzés eredményét az alábbi tizenhetedik ábra mutatja be. A tizenkettedik elemtől a rendszer csak előrejelzett elemekből dolgozik. Az MSE értéke ebben az esetbe 18 782 volt.



17. ábra *Kísérlet az „airline-passanger” adatsor megtanulására II.*

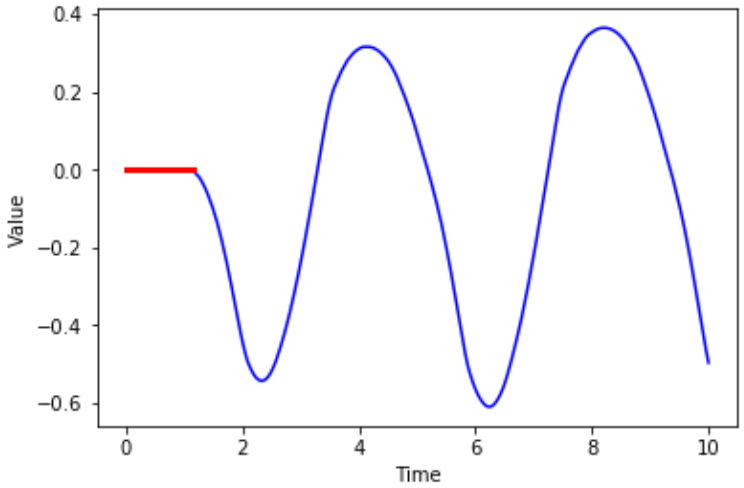
Az előzetes kísérletek során több mint húsz tesztet hajtottunk végre. Az alacsony futtatási szám annak az eredménye, hogy a futási idő ilyen magas tanítási ciklusok eredményeképpen drámaian megnőtt. Egy neurális háló futtatása akár egy teljes napot is igényelhetett. A fenti ábrán látható kimeneti választ egy 10 000 epoch-os futtatás eredményeképpen kaptam. A háló öt darab rejtett réteget tartalmazott 200,400,300,200 és 100 számosságú LSTM blokkokkal. Az aktivációs függvényeket az LSTM blokk sztenderd logisztikus függvény beállításán hagytam. Ennek oka abban keresendő, hogy függetlenül a többi változtatható paraméter értékétől, ha akár egy rejtett réteg aktivációs függvényét megváltoztatjuk a rendszer nem képes még a fenti eredmények reprodukálására sem. A fenti eredmények fényében arra jutottunk közösen a témavezetőmmel, hogy mivel a Kerasban létrehozott modellek tanítási ideje rendkívül hosszú és mivel a fenti eredmények nem adnak komolyabb bizakodást a szekvenciából vektort előállító struktúrát illetően, a kísérleteket Tensorflowban kell folytatni a gyorsabb futás miatt és az átláthatóbb modellezés mellett. A témavezetőmmel egyhangúan úgy döntöttünk, hogy a kísérleteket már a szekvenciális bemenetekből szekvenciális kimeneteket képző struktúrára folytatjuk. Fontos kiemelni, hogy a Tensorflowban a tanulási rátára vonatkozóan nincsenek alapbeállítások és ezt a felhasználónak kell felvenni, ami egy kritikus pont, hiszen a szakértői vélemények szerint egyértelműen komoly ráhatása van a paraméter értékének

A célunk a másik megközelítés esetében az, hogy a modellünk egy olyan kimenetet adjon, amelynek a hossza megegyezik a bemeneti adatsorunk hosszával. Ennek nagy előnye, hogy a neurális háló a tanítási jóságát minősítő célfüggvényt nagyobb mintán tudja ellenőrizni és ebből kifolyólag pontosabban képet lehet alkotni arról, hogy hogyan kell a súlyokat megváltoztatni az egy tanítási ciklusok végén. Az első teszteket itt is a szinusz hullám esetén végeztem el, azonban itt kezdetektől fogva csak egy rejtett réteget alkalmaztam, mivel a több réteg alkalmazása egyértelmű túltanulást eredményezett. A réteg összesen 100 LSTM blokkot tartalmazott. A bemeneti adatok 30 időpillanatot ölelnek fel. A tanulási batch értéke 1-re lett beállítva, a tanulási ciklusok száma pedig 2000. A tanulási ráta értékét 0.0001-re vettem fel. A modell analógiája úgy működik, hogyha bevesz egy bemenetet például [1,2,3,4] elemeket azt várjuk eredményül, hogy visszaadja [2,3,4,5] számokat predikció gyanánt. A jóslást így továbbra is csak egy darab elemmel gördítjük előre a jövőbe, ellenben nagyobb az ellenőrzési spektrum és a szekvenciákkal is könnyebben tud a rendszer dolgozni. Az alábbi tizennyolcadik ábrán azt láthatjuk hogy a kék pontok a bemenetek, a feketék adják a cél jóslati elemeket, a piros pontok pedig azt mutatják, hogy mit jósolt a rendszer a fenti paraméterek fennállása esetén a tanulást követően.



18. ábra *Szekvenciából szekvencia leképzés szinusz hullámon*

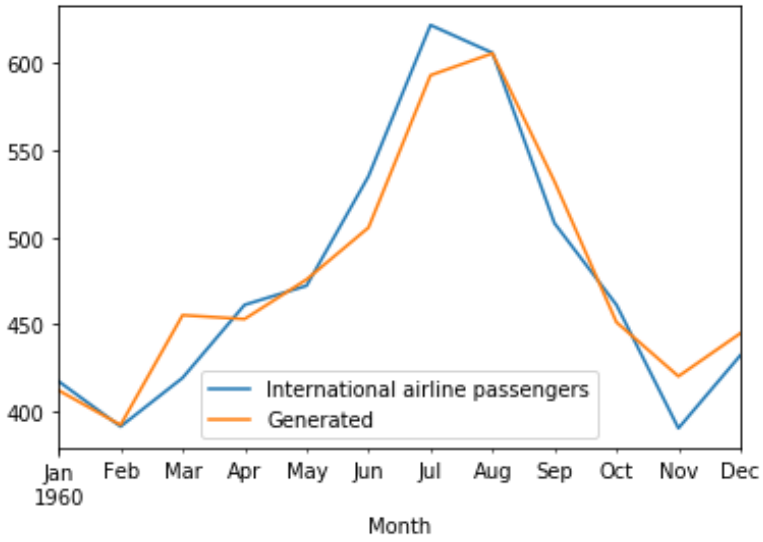
Bár a kimenet elején a pontok nem mutatnak teljesen jó illeszkedést, sőt egy kiugró pontot is láthatunk a kapott idősor vége felé az illeszkedés gyakorlatilag a nulla hibát közelíti meg. A modell külön érdekessége, hogyha a tanulást követően csak csupa nullából álló kezdő bemenetet adunk rá a rendszerre. A neurális háló a korábban megtanult jelenségek alapján megpróbálja a „rossz” bemenet ellenére is a kimenetet szinuszos jellegű függvénnyé alakítani. Az alábbi tizenkilencedik ábra ezt mutatja be. A piros vonal az ábrán a csupa nulla bemenet mutatja, míg a kék a rendszer bemenetre adott válaszát.



19. ábra *Hibás bemenet alapján próba a szinusz hullám reprodukálására*

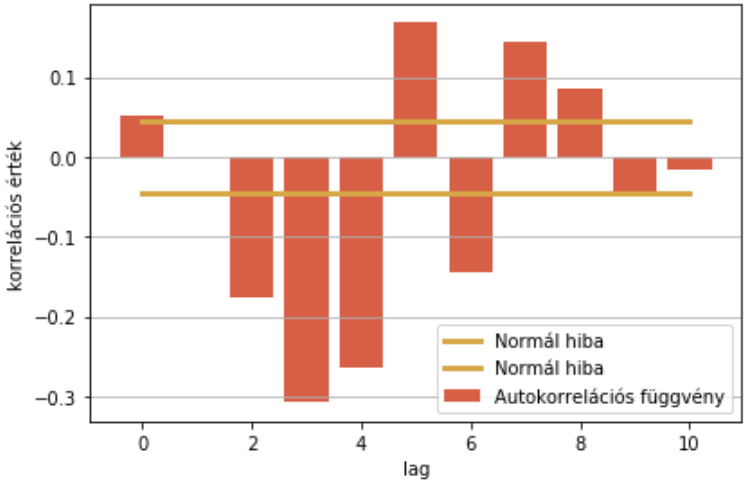
A fenti sikerek láttán úgy gondoltam, hogy célszerű megvizsgálni, hogy a modell mit képes produkálni az „airline-passanger” adatállományon. A modellhez annak érdekében, hogy jó eredményeket kapjunk a következő struktúrát vettem fel: a tanulási ráta értékét 0.002-re állítottam, a rétegben a neuronok számát 400-ra vettem fel, míg a tanítási ciklusok számát 11000-ben állapítottam meg, míg a tanulási batch-ek méretét továbbra is 1-re rögzíttetem. A kapott eredményt vizuálisan az alábbi huszadik ábra mutatja be. Az MSE értéke drasztikus javulást mutatott a Keras modellekkel szemben, mindösszesen 403.875 értéket vett fel, ami

nagyságrendekkel jobb. A narancssárga a prediktált eredmény, a kék pedig a tényleges adatot mutatja.



20. ábra *Az „airline-passenger” adatsor jövőjének jóslása szekvenciális kimenettel*

A modell által produkált hibát ezt követően fehérzaj tesztnek vetettem alá annak érdekében, hogy meggyőződjek a modell helyességéről. Amint az ábrából is látszik, a modell által produkált zaj nem felel meg a fehérzaj kritériumnak, mivel vannak benne szignifikáns értékek, ahogy a korrelogramon is látszik az alábbi huszonegyedik ábrán. Ebből arra lehet következtetni, hogy a tanulás során a modell alul tanult, mivel kisebb epoch számok esetében kevésbé illeszkedő predikciókat kaptam.



21. ábra *Az „airline-passanger” adatsor szekvenciális kimenettel előállított előrejelzésének fehérzaj tesztje*

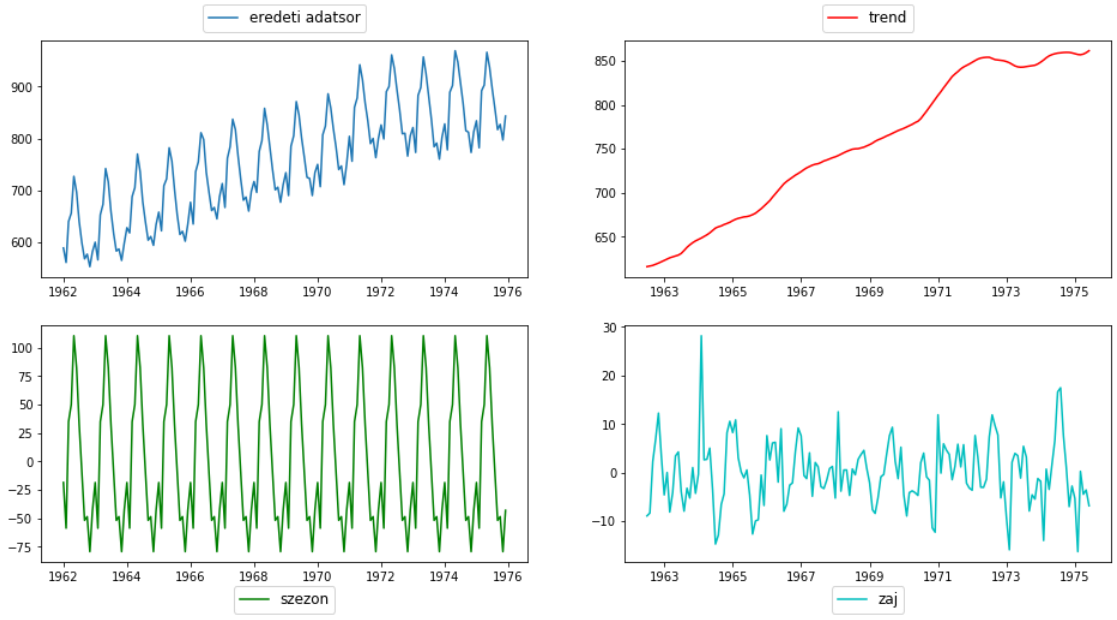
A modellt feltehetően tovább lehetne javítani, de a többszöri futtatás elegendően meggyőző bizonyítékot szolgáltatott arra vonatkozóan, hogy a további kísérletek helyes iránya az, ha ezt az utat követem. Az eredményekből az vehető ki, hogy a szekvenciából szekvenciát leképző modellek egyértelműen jobb eredményre vezetnek, ezért a kísérleteket további folytatását a témavezetőmmel közösen megegyezve ebbe az irányba folytattuk.

3.2. Kísérletterv a mesterséges intelligencia paramétereinek vizsgálatához

Az analitikus úton nehezen vagy egyáltalán nem kezelhető modellek felállítására az egyik legkézenfekvőbb módszer kísérletek elvégzése és azok kiértékelése statisztikai eszközök segítségével. Néhány rendkívüli esetben és kisebb problémáknál az úgy nevezett „trial and error" módszertana célravezető lehet, azonban az olyan komplexitású modelleknél, ahol a modell paramétereinek sokaságából adódóan kombinatorikailag robbanó problémát kell megoldani, úgy pusztán próbálgatás útján történő kísérletezés szinte minden bizonnyal nem fog eredményre vezetni. Az ilyen esetekben a kísérletek lefolytatásának menetét megfelelő mederbe kell terelni és szabályozott keretek között kell azokat lefolytatni.

3.2.1. A kísérletek célja

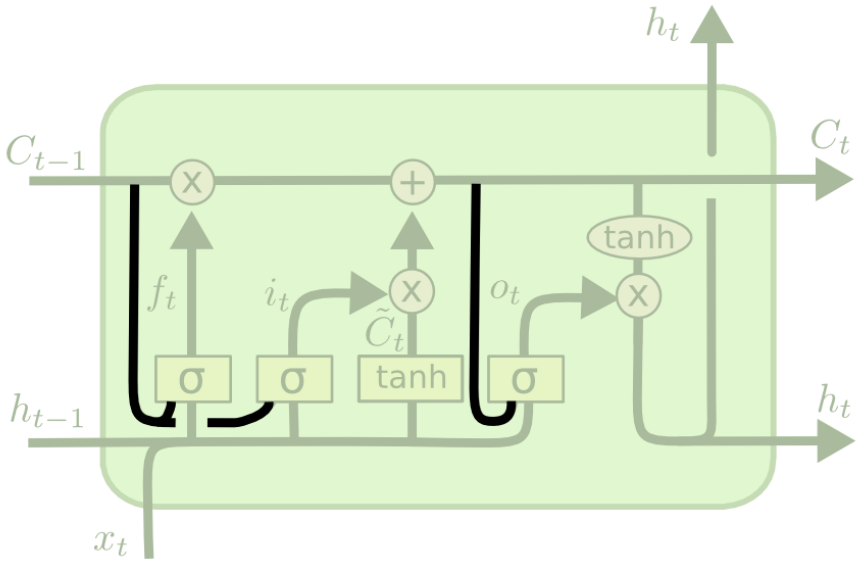
A korábban tárgyalt tulajdonságok és eredmények kiválóan alkalmassá teszik ezt az eszközt az idősorok előrejelzésére, hiszen ha a mesterséges intelligencia képes az idősorban rejtőző alapjel, trend és szezon hatásokat megtanulni, akkor alkalmassá válik az idősor jövőjének megmondására. A visszacsatolás lehetővé teszi, hogy szekvenciákat tanuljon meg és a korábbi idősori elemeket is figyelembe véve mondja meg az adatsor jövőjét a rendszer. A legalkalmasabbnak erre az LSTM „peephole" típusú hálóit találom, amely felismerésben Pierian Data szakemberei is megerősítettek. Ezek a szakemberek javasolták továbbá, hogy vizsgálódásomat a „monthly milk production" adatállományon folytassam, mivel az könnyebben kezelhető. Így javaslatukat megfogadva a kísérleteket ezen az adatsoron fogom elvégezni, az adatsort az alábbi huszonkettedik ábra mutatja be, az ábrán továbbá látható a komponensekre való bontás is.[28]



22. ábra *A „monthly milk production" adatsor elemeire való dekompozíciója*

A kísérletek lefolytatásának célja, hogy feltárásra kerüljenek azok az összefüggések, amelyek meghatározzák az LSTM „peephole" típusú hálók paramétereinek előrejelzési pontosságra gyakorolt hatását. A tesztek lefolytatásának eredményéül remélhetőleg fény derül arra, hogy bizonyos feltételezések, amelyeket a „Tézisek" alfejezetben bővebben kifejtek igaznak

bizonyulnak-e. Az LSTM „peephole” szerkezet az alábbi huszonharmadik ábrán látható. Az ábrán kiemelve látható, hogy a cella állapotot bevezetjük a kapukba, annak érdekében, hogy pontosabb eredményeket érhessünk el.[29]



23. ábra *A „peephole” LSTM vázlata*

3.2.2. A kísérletterv módszertana

A kísérletek lefolytatása során faktoroknak nevezzük azokat a paramétereket, amelyek befolyásolják a kísérlet eredményét. A faktor egy mérhető vagy minősíthető jellemző, amely adott időpillanatban meghatározott értékkel bír. Egy adott faktornak több szintje lehet. Szintnek nevezzük a faktor által, a kísérlet során felvehető értékek számosságát. Amennyiben nem lehetünk biztosak a modell lineáris viselkedésében, úgy több szintet kell definiálni a kísérlet tervezése során.

Kezdetben a kísérletek megtervezésénél a teljes kísérleti mező feltérképezésével próbálták megismerni a maximális eredmény eléréséhez szükséges körülményeket. Ez azt jelentette, hogy feltárták azokat a hatásokat (faktorokat), amelyek befolyásolhatták az eredményt (optimalizációs paramétert), majd a faktorok minden lehetséges értéke (faktor-szint) mellett kísérleteket végeztek. Először kiválasztották az egyik faktort és annak a szintjeit változtatták lehetőleg nem túl nagy lépésekben, miközben a többi faktor szintjét változatlan értéken tartották, és minden faktor-szint mellett egy vagy több kísérletet végeztek. Ezután egy másik, majd egy harmadik faktort változtatva és a többit változatlan értéken tartva újabb és újabb kísérleteket végeztek. A kísérleteket végül is minden faktor-szint kombináció (kísérleti beállítás) mellett elvégezték, és így ki tudták választani az optimális paraméter beállítást. [30]

A kísérlettervezéssel foglalkozó irodalom többféle megközelítést javasol a tesztek lefolytatására. A legalaposabb és legpontosabb eredménnyel kecsegtető módszertan a válaszfelületek feltárása. Ebben az esetben szükséges a probléma matematikai modelljének ismerete, vagy pedig nagyszámú erőforrás ráfordítást igényel és a lehetséges összes paraméterkombináció megvizsgálását. Mivel az előző feltételek egyike sem biztosított teljes mértékben, ezért a teljes faktoriális kísérlettervezés módszertana javasolt.

A teljes faktoriális kísérlet egy olyan módszer, amely lehetővé teszi az egyes faktorok és ezek együttes hatásának vizsgálatát a minőségi jellemzőre vonatkozóan. Az egyfaktoros módszerrel szemben itt egyszerre több faktort változtatnak. Ezáltal lehetővé válik a beállításokhoz kapcsolódó középértékek és az ún. hatások számítása. Megkülönböztetjük a főhatásokat, amelyek az egyes faktorok beállításából erednek, és a kölcsönhatásokat, amelyek több faktor egyidejű beállításának eredményeképpen keletkeznek.[31]

A faktoriális terv készítésénél első lépésben a vizsgálat tárgyát képező eredmény paramétereket kell meghatározni. A második lépésben azoknak a változóknak az együttesét kell feltárni, amelyek változtatása hatással van a vizsgálat tárgyát képező eredmény paraméterekre. Ezt követi a kiválasztott faktorok szintjeinek megállapítása. A szint definiálás során törekedni kell a teljeskörűség és tömörség egyensúlyának megtartására. A komplex problémák esetén minden esetben több szint definiálására van szükség. Ezt követi teljes faktoriális terv táblázatának elkészítése és a kísérletek lefolytatása, majd végül az eredmények kiértékelése.

3.2.3. A kísérletterv faktorai

A neurális hálók optimális működési tartományának megtalálása egy nehéz optimalizálási feladat, amely jórészt intuitív alapokon nyugszik. Általánosságban azt lehet mondani, hogy a mesterséges intelligencia komplexitását a feldolgozandó adatsor struktúrájához kell igazítani. Erre vonatkozóan a szakirodalomban nem található lényegében semmilyen irányelv, hogy egy-egy probléma esetében hogyan lenne célszerű a háló architektúráját kialakítani, csak a neuronok felső korlátjára vonatkozó rendkívül túlzó becslések léteznek.[2] Ez különösen igaz a keresleti adatok esetére, ezért szeretnénk ezeket az összefüggéseket megvizsgálni, amelyek alapját adhatnék egyfajta neuron háló architektúra identifikációs eljárás későbbi kifejlesztésének. A modellek konstruálása során számos paramétert lehet változtatni, amelyek a modell pontosságát nagymértékben befolyásolják. A cél a neurális hálók felállítása során egy olyan paraméter együttes megtalálása, amely segítségével a lehető legjobb eredményt kapjuk a teszt adatállományon.

Az eddig megszerzett ismereteim során az alábbi paramétereket tartom a legfontosabbnak:

- a rejtett rétegek száma (*hidden layers*),
- tanulási ráta (*learning rate*),
- adott rétegben lévő neuronok száma (*neuron numbers*),
- a tanítási ciklusok száma (*epochs*),
- tanítási kötegek nagysága (*batch size*),
- szekvencia hossz (*look_back*).

Azonban annak érdekében, hogy a kísérletek lefolytatása beleférjen az időbe, a vizsgálandó paramétereket az alábbiakra korlátozzom:

- tanulási ráta (*learning rate*),
- adott rétegben lévő neuronok száma (*neuron numbers*),
- a tanítási ciklusok száma (*epochs*).

A fent említett paraméterek hatása a modell pontosságára igen nagy valószínűséggel nem lineáris, ezért több szintű teljes faktoriális kísérlet javasolt. A szintek számát a számítási kapacitások hiánya miatt három, négy és öt szintben maximalizálom. A tesztek során a háló

kimenetén hiperbolikustangens aktivációs függvényt alkalmazok, ennek az az oka, hogy Pierian Data szakemberei kezdésnek ezt javasolták. Itt fontosnak tartom megjegyezni, hogy a szakemberek a tanuló batchek fontosságára is felhívták a figyelmemet, hogy akár komolyabb ráhatása is lehet a kísérletek eredményére, azonban az időszűkössége miatt batch-ek méretét 1 értékén rögzítem. A kísérleteket nagyban befolyásolja a véletlenszerűség, mivel az elemek súlyait véletlenszám generátor inicializálja, minden kísérletet ötször megismétlek a vizsgálatok során.

A kísérletek során a kimeneten egy úgynevezett „output projection wrapper" réteget alkalmazok, amely a korábban említett Pierian Data szakemberei szerint javíthat a modell pontosságán, mivel ez az objektum elméletileg segíti a szekvenciák kiterjesztését a válaszra, amelyet a rendszer adott bemenet esetén produkál. A vizsgált faktorok által felvehető értékeket az alábbi nyolcadik táblázat tartalmazza.

8. táblázat *Faktor táblázat*

Faktoriális terv	Faktor szintek				
	1.szint	2.szint	3.szint	4.szint	5.szint
Tanulási ráta	0,001	0,01	0,1	-	-
Neuronok száma	10	50	100	500	-
Tanítási ciklusok száma	1000	2000	3000	4000	5000

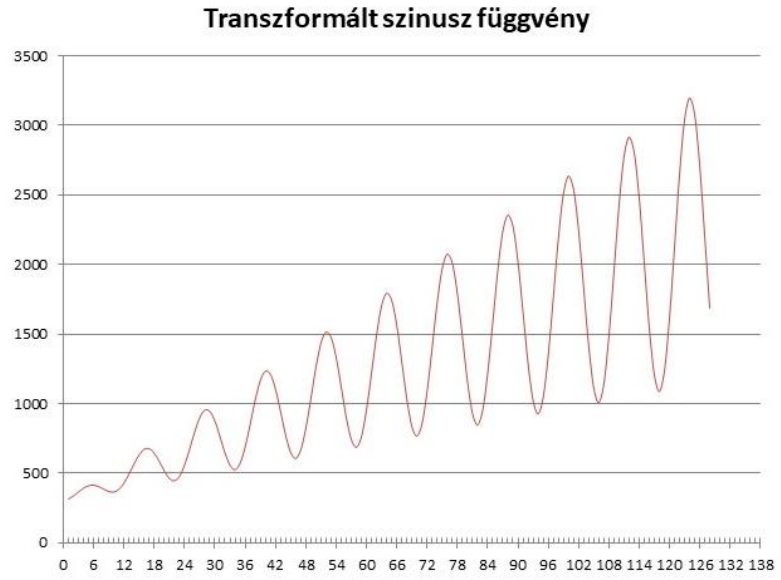
Az első fejezetben taglaltak miatt a neurális hálókat az MSE értékek alapján tanítom, de a tesztállományra adott választ több hiba mutatóra is megvizsgálom, hogy teljesebb képet alkothassak a rendszer működéséről. A MSE értéken túlmenően az alábbi értékek kerülnek vizsgálatra:

- MAPE,
- MASE,
- RMSPE,
- RMSE,

A kísérletek kontrolálása végett a teljes faktoriális kísérlettervet megismétlem egy módosított szinusz jellegű görbére, amely úgy van transzformálva, hogy az amplitúdója az idő előrehaladtával egyre nőjön, valamint trend is van benne. A függvénynek az egyenlete az alábbi módon írható le:

$$f(x) = 15 \cdot x + 300 + 2 \cdot \sin(x) \cdot x$$

A függvényben *x* csak pozitív egész értékeket vehet fel. A függvényt az alábbi huszonnegyedik ábra mutatja be.



24. ábra *Transzformált szinusz hullám*

3.2.4. Hipotézisek

A fent leírt modell és a kutatási kérdések alapján megfogalmaztam hipotéziseket, amiket a vizsgálataim során igazolni vagy elvetni szeretnék, hogy jobban megértsem, mivel javíthatnék az aggregát előrejelzések pontosságán.

- ***H1:*** A vizsgálat fókuszában álló paraméterek vannak a legnagyobb ráhatással a modell működésének helyességére.
- ***H2:*** Az egy rejtett réteggel bíró LSTM háló képes megtanulni úgy az idősori sajátosságokat, hogy a kimeneti hiba csak fehérzaj legyen.
- ***H3:*** Az epoch szám befolyásolja leginkább a modell pontosságát, azaz a vizsgált hiba mutatókban kimutatható hiba nagyságát.
- ***H4:*** A tanulási ráta és a rejtett rétegben lévő LSTM blokkok számának együttes hatása határozza meg leginkább az elérhető pontosságot, azaz a vizsgált hiba mutatókban kimutatható hiba nagyságát.
- ***H5:*** A paraméterek önálló ráhatása a modell pontosságára számottevőbb, mint ahogy a paraméterek páronként befolyásolják a pontosságot, azaz a vizsgált hiba mutatókban kimutatható hiba nagyságát.
- ***H6:*** Különböző tulajdonságokkal bíró idősorokhoz, eltérő paraméter beállításokkal bíró neurális háló vezet eredményre

A tézisek elfogadását, vagy elvetését statisztika módszerekkel próbálom cáfolni, vagy pedig elfogadni.

3.2.5. A vizsgálati módszerek

A kísérletek analízisét a Minitab17 szoftver segítségével végzem el és a program által kínált leíró statisztikai eszközök segítségével próbálok következtetéseket levonni a kísérletekről,

valamint a modellt befolyásoló paraméterek fontosságáról. A teljes faktoriális terv eredményeit először is regresszió analízissel vizsgálom.

A statisztikai vizsgálatok során gyakran szükséges, hogy a változók közötti kapcsolatot függvény formájában fejezzük ki. Az egyik kitüntetett változót (függő változót) két vagy több változó (független változó) függvényeként (egyenleteként) akarjuk kifejezni. Az így meghatározott egyenlet azt fejezi ki, hogy a függő változót hogyan magyarázza, a többi változó, vagyis milyen hatással vannak rá. A változók közötti kapcsolat egyenletszerű megismerése azért fontos, mert az egyenlet (a modell) használata általánosítja és függetleníti a probléma megoldást a mintaválasztástól.[32]
Amennyiben a modell helytálló, a regressziós hibáknak normális eloszlást kell követnie. Ha nem felel meg a normalitás tesztnek, a reziduál sor az a következőket jelenti:[33]

- hiányzik egy vagy több vizsgált változó,
- egy vagy több változó faktor szintjei nem megfelelően lettek felvéve,
- nem létező interakciók szerepelnek a modellben.

Azonban jóhírnek mondható, hogy szimulációs eredmények azt mutatják, hogy a regresszió analízisből számított további deszkriptív statisztikai mutatók megbízhatóak amennyiben nagyszámú minta áll rendelkezésre (legalább 15). [34] Az adatok normalitását Anderson-Darling teszt segítségével fogom elvégezni.

A regresszió analízist követően ANOVA vizsgálatot hajtok végre. Az ANOVA eltérő módon lerögzített varianciák segítségével viszonyítja egymáshoz a populáció különböző középértékeit. Adott vizsgálat során előálló teljes adatmennyiség, mint alaphalmaz összesített szórását, konkrétabban, összesített varianciáját analizálja abból a nézőpontból, hogy az ingadozás okára keresi a választ. Annak a tisztázását segíti, hogy a fentebb említett szórás béli eltérések mögött a véletlen vagy egy másik magyarázó tényező hatása bújik-e meg. Ilyen tényezőnek tekinthető adott populáción belüli csoportok átlagai közti eltérést. A varianciák számítását és becslését arra a matematikai tényre alapozva vezeti le, hogy a teljes variancia számlálója, azaz a teljes eltérés-négyzetösszeg független elemek összegeként állítható elő, emellett a nevező, azaz a szabadsági fok az adott komponensek szabadsági fokainak összegeként áll elő.[35]

Az ANOVA F-tesztet használ arra, hogy megállapítsa a csoportok várható értékeinek változékonyságát. Az F-értéket kiszámítva, megvizsgálva az F-eloszlás alapján, ha szignifkáns értéket kapunk ($p<0.05$) az azt jelenti, hogy a csoportok várható értéke nem egyező. Ezzel lényegében arra lehet következtetni a faktoriális kísérletterv keretein belül, hogy minél kisebb egy faktor szignifikanciája annál inkább nagyobb hatást gyakorol a kísérlet végeredményére.[36]

3.3. A kísérletterv eredményeinek statisztikai vizsgálata

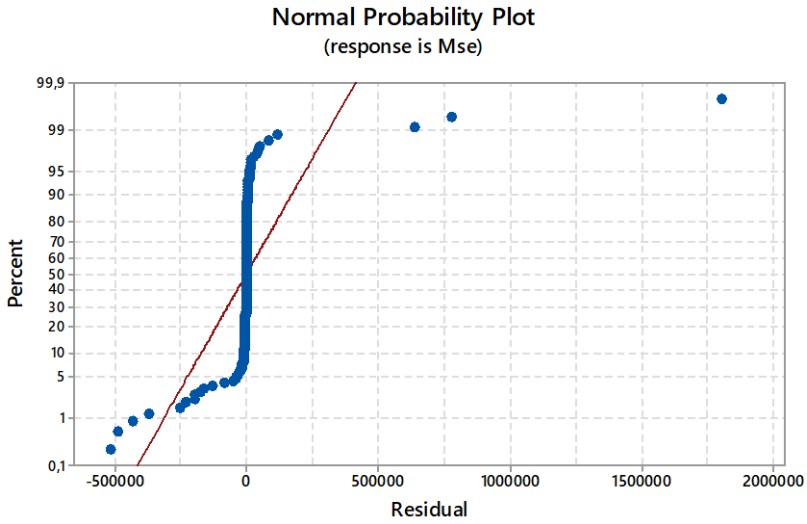
A kísérletek során a modellek tesztelésénél nemcsak egy hibamutató vizsgálatát végeztem el, ezért első lépésben célszerűnek találom ezeknek a mutatóknak a korrelációját kiszámolni. A hibamutatók korrelációját a Pearson féle korreláció alapján vizsgáltam meg és az eredményeket az alábbi kilencedik táblázat mutatja be. Az adatokból tisztán látszik, hogy a hibamutatókra a

paraméterek beállításai gyakorlatilag egyformán hatnak, amire előzetesen számítani is lehetett, hiszen a származtatási alapjuk azonos, ahogyan az a második fejezetben kifejtésre is került.

9. táblázat *A hibák korrelációs vizsgálata I.*

Mutatók	Mape	Mase	Mrae	Mse
Mase	1			
Mrae	0,999	0,999		
Mse	0,913	0,912	0,906	
Rmspe	0,911	0,91	0,905	1

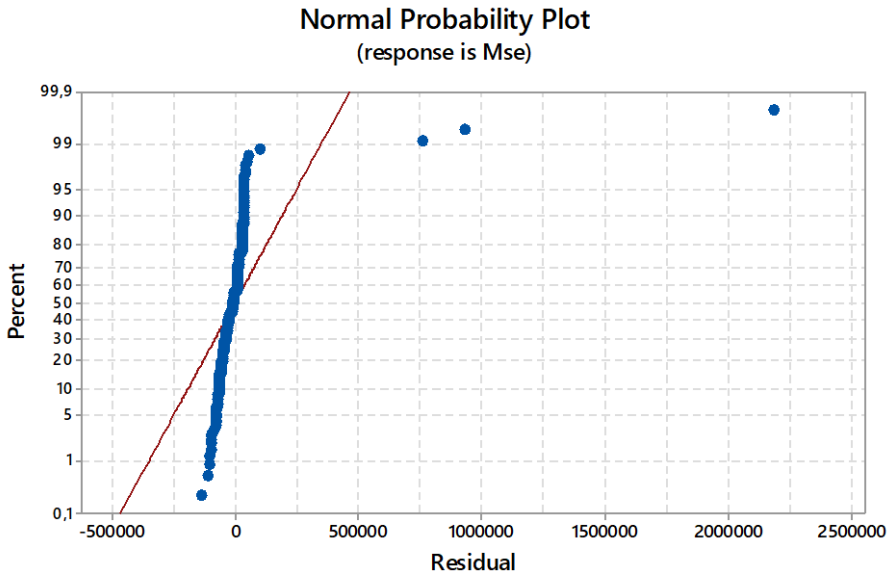
Mivel a korrelációs mutatók nagyon szoros összefüggést mutatnak, ezért a további statisztikai vizsgálatokat az MSE mutatóra korlátozom, mivel véleményem szerint ez a leginkább kardinális hiba mutató. A korrelációs számítást követően azt vizsgáltam meg, regresszió analízis útján, hogy a vizsgált paraméterek függvényében fel lehet-e állítani egy lineáris modellt, ami leírja a rendszer működését. Sajnálatos módon a felállított regressziós modell és valós adatokból származtatott hiba a normalitás teszten jelentős mértékben megbukott, ahogy az alábbi logaritmikus skálán is lehet látni. Az alábbi huszonötödik ábrán a piros vonal a Gauss eloszlást mutatja logaritmikus skálán, a kék pontok pedig a vélt modell eredményeinek és a tényleges kísérleti eredmények különbségéből származó reziduálokat mutatja. Amennyiben normális eloszlású lenne a hiba úgy sokkal pontosabb illeszkedést mutatna.



25. ábra *Az első kísérletsorozat normalitás tesztje paraméter interakciókkal*

A nem elfogadható eredmény valószínűleg annak köszönhető, hogy a modell pontosságára, ahogy arra több szakember is felhívta a figyelmemet, a tanuló batch-ek mérete is hatással van, valamint több mint biztos, hogy a kísérleti faktorok szintjei (a rendelkezésre álló idő hiányában) nem megfelelően lettek megállapítva. A nem létező interakciók tényét azért nem kell megemlíteni, mivel ha a feltételezett interakciók vizsgálatát kivesszük a rendszerből, úgy még

a fentihez hasonlóan rosszul illeszkedő eredményt kapunk. Ezt mutatja be az alábbi huszonhatodik ábra.



26. ábra *Az első kísérletsorozat normalitás tesztje paraméter interakciók nélkül*

A fentiekből messzemenő statisztikai következtetéseket ugyan nem lehet levonni, de a gyakorlati tapasztalatok azt mutatják, hogy a paraméterek interakciók útján igenis befolyással vannak modell pontosságára. Az egyes paraméterek és a paraméter interakciók modellre gyakorolt hatásait az alábbi tizedik táblázatból lehet leszűrni. Fontos kiemelni, hogy mivel a modell a normalitás tesztnek nem felelt meg, az ANOVA eredmények megbízhatósága is csökken, azonban továbbra is mérvadónak tekinthető nagyvonalakban, ahogy az fentebb tárgyalásra került. Az eredményekből egyértelműen az vehető ki, hogy a legnagyobb ráhatása a hibamutatókra a tanulási rátának és a rétegben található neuronok számának volt, valamint az előbb említett két paraméter együttes hatása volt még rendkívül szignifikáns.

10. táblázat *Az első kísérletsorozat ANOVA analízisének eredménye*

Factors	DF	Adj SS	Adj MS	F-Value	P-Value
Model	59	1,92E+12	32515957423	1,44	0,029
Linear	9	5,95E+11	66071796592	2,93	0,003
Learning_rate:	2	2,49E+11	1,25E+11	5,53	0,004
Number_of_neurons:	3	2,77E+11	92233935943	4,1	0,007
Number_epochs:	4	68733295078	17183323769	0,76	0,55
2-Way interactions	26	9,04E+11	34781176793	1,54	0,049
Learning_rate:*Number_of_neurons:	6	5,50E+11	91654766091	4,07	0,001
Learning_rate:*Number_epochs:	8	1,40E+11	17507455849	0,78	0,623
Number_of_neurons:*Number_epochs:	12	2,14E+11	17860196107	0,79	0,657
3-Way interactions	24	4,19E+11	17478530084	0,78	0,765

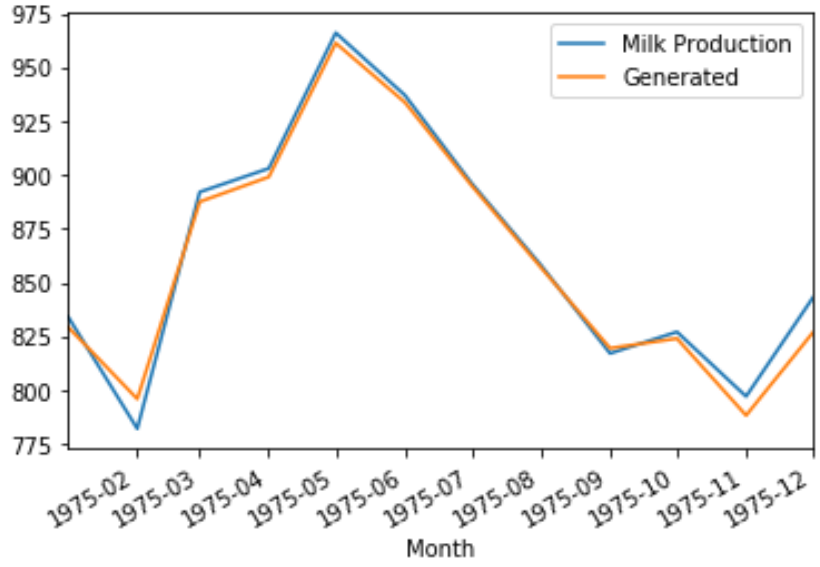
A továbbiakban érdemes lehet megvizsgálni, hogy melyek azok a paraméterek, amikkel futtatva a mesterséges intelligenciát a lehető legjobb tesztelési eredményeket kaptam. Ennek a táblázatos összefoglalását mutatja be az alábbi tizenegyedik táblázat. A táblázatban szereplő

értékek az MSE célkritérium alapján vannak sorba rendezve. A kísérleteket áttekintve be kell látni, hogy a vizsgált adatsor esetében, a hibamutatókat leginkább úgy lehetett minimalizálni, ha a tanulási ráta értékét 0,01-re állítottam be. További érdekesség, hogy a vizsgált adatsor esetében leginkább a 100 értékre beállított neuron szám vezetett eredményre. A táblázat adatai egybevágnak az ANOVA azon következtetésével, miszerint az epoch-ok száma kevésbé játszik meghatározó szerepet a tanulás pontosságában.

11. táblázat *Az első kísérletsorozat legjobb eredményei mutatókkal*

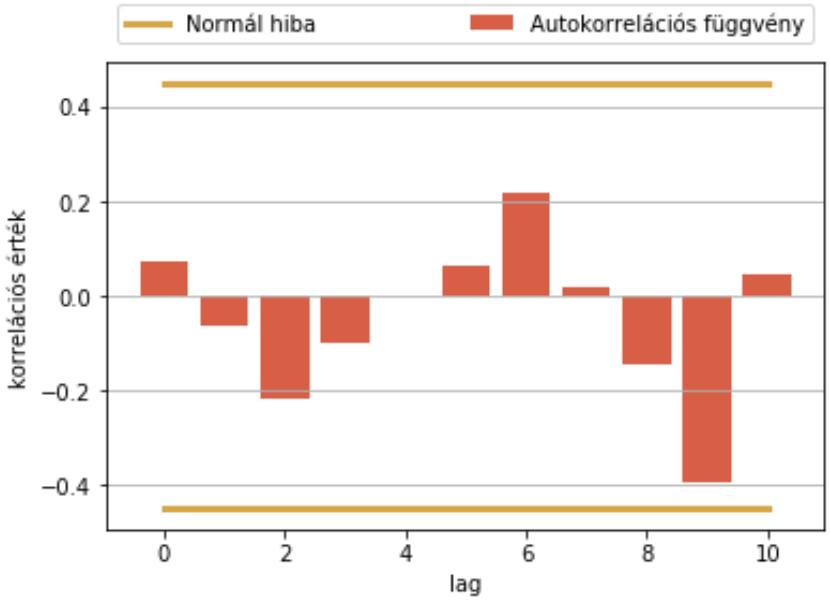
Mape	Mase	Mrae	Mse	Rmspe	Tanulási ráta	Neuronok száma	Tanítási iterációk száma
0,006733	0,132086	0,175117	53,54334	7,88125E-05	0,01	100	5000
0,008875	0,174506	0,291428	82,4637	0,000117912	0,01	50	4000
0,009155	0,181193	0,244017	85,50569	0,000121353	0,01	100	4000
0,009275	0,181565	0,261971	113,6323	0,000163543	0,01	500	4000
0,011446	0,230361	0,314942	123,3029	0,000165971	0,01	50	5000
0,009965	0,194426	0,317496	127,1665	0,000185505	0,01	50	4000
0,011231	0,22259	0,308942	133,1877	0,000187054	0,01	100	4000
0,012356	0,244951	0,392814	192,1115	0,00027411	0,01	500	5000
0,012927	0,257782	0,360218	199,6133	0,000279801	0,01	100	5000
0,013247	0,262012	0,496753	212,0227	0,000299821	0,01	100	3000
0,01488	0,297661	0,648425	249,3514	0,000340542	0,01	10	5000
0,016413	0,324133	0,539249	265,8549	0,000382134	0,01	100	3000
0,014391	0,288101	0,424507	271,0952	0,000384673	0,01	100	2000
0,018341	0,368119	0,587846	291,2737	0,000398539	0,01	500	3000

A legjobb modell előrejelzési eredményének és a tényleges adatoknak az összehasonlítását az alábbi huszonhetedik ábra mutatja be. Az ábrán jól látható, hogy a mesterséges intelligencia alkalmas volt a képességeinek köszönhetően egy általános képet alkotni az adatsor mechanizmusairól és előrejelezni annak jövőbeli alakulását.



27. ábra *Az első kísérletsorozat legjobb eredménye*

Annak érdekében, hogy az előrejelző modell pontosságáról meg lehessen győződni, a tényleges és prediktált adatokból származtatható reziduál sort fehérzaj tesztnek kell alávetni. A teszt eredményét az alábbi huszonnyolcadik ábra mutatja be. Jól látható a korrelogramon, hogy nincsenek szignifikáns értékek, tehát a modell által produkált hiba megfelel a fehérzaj kritériumnak.



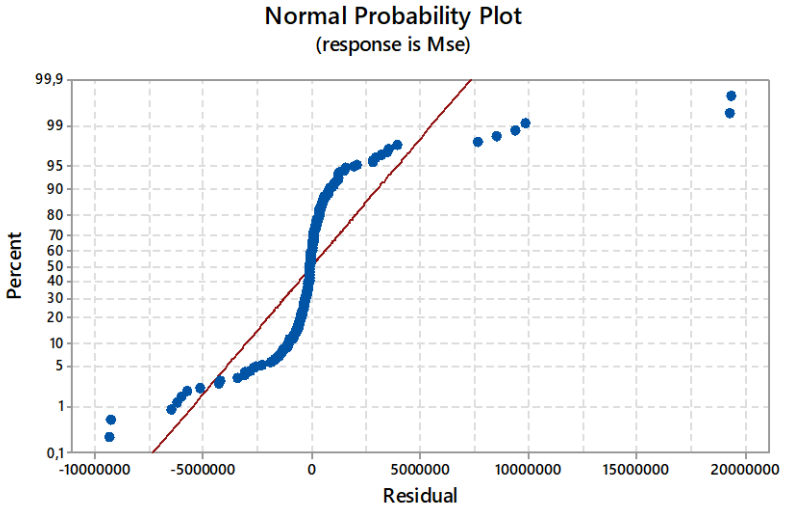
28. ábra *Az első kísérletsorozat legjobb eredményének fehérzaj tesztje*

A kereskedelmi jellegű adatsor elemzését követően a transzformált szinusz hullámra lefolytatott kísérletek elemzése következett. Hasonlóan, mint az előző elemzésben itt is a hibamutatók korrelációs vizsgálatával kezdem. Ahogyan azt a másik adatsor esetében is tapasztalhattuk, a mutatók korrelációja itt is kellően magas fokot mutat. Ezt mutatja be az alábbi tizenkettedik táblázat

12. táblázat *A hibák korrelációs vizsgálata II.*

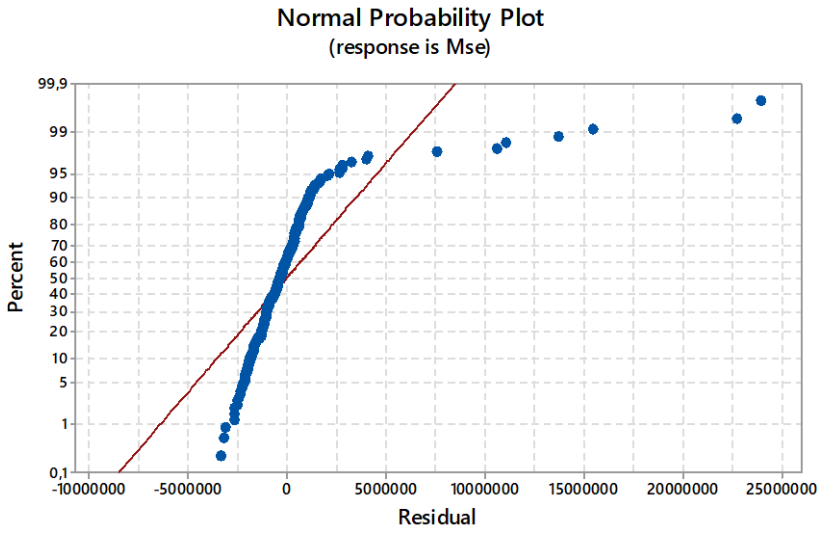
Hibamutatók	Mape	Mase	Mrae	Mse
Mase	0,991			
Mrae	0,986	0,996		
Mse	0,907	0,907	0,888	
Rmspe	0,893	0,874	0,854	0,987

Akárcsak az előző esetben, az erős korreláció miatt itt is a továbbiakban csak MSE mutatót vizsgálom, mivel a hibamutatók között továbbra is fennáll a nagyon erős korreláció, amit szintén a Pearson féle korreláció számítás alapján került kikalkulálásra. Sajnálatos módon a felállított regressziós modell és valós adatokból származtatott hiba a normalitás teszten jelentős mértékben megbukott, ahogy az alábbi logaritmikus skálán is lehet látni ezt az eredményt a huszonkilencedik ábrán.



29. ábra *A második kísérletsorozat normalitás tesztje paraméter interakciókkal*

A normalitás teszten való megbukásnak az okait valószínűleg a hasonló problémákra lehet visszavezetni, mint amiket az előző adatsor esetében tapasztaltam. Ezt alátámasztja, hogy az esetleges interakciókat eltávolítva szintén rossz eredményt kapunk, amit az alábbi harmincadik ábrán is láthatunk.



30. ábra *A második kísérletsorozat normalitás tesztje paraméter interakciók nélkül*

A fentiekből messzemenő statisztikai következtetéseket ugyan nem lehet levonni, de a gyakorlati tapasztalatok azt mutatják, hogy a paraméterek interakciók útján igenis befolyással vannak a modell pontosságára. Az egyes paraméterek és a paraméter interakciók modellre gyakorolt hatásait az alábbi tizenharmadik táblázatból lehet leszűrni. A táblázatból az olvasható ki, hogy a paraméterek egyéni hatása érvényesül inkább és ebben az esetben a rétegben használt neuronok hatása nem szignifikáns, szemben az előző esethez képest, ahol is a tanítási iterációk száma volt kevésbé fontos. A statisztikai mutatók ebben az esetben viszont azt sugallják, hogy a paraméterek interakciós hatása külön-külön nem olyan számottevő, mint az előző esetben. Az eltéréseket véleményem szerint egyértelműen az adatsorok jelleg és komplexitás béli különbségeivel lehet magyarázni. Emiatt úgy gondolom kijelenthető, hogy különböző tulajdonságokkal bíró idősorokhoz, eltérő paraméter beállításokkal bíró neurális háló vezethet eredményre.

13. táblázat *A második kísérletsorozat ANOVA analízisének eredménye*

Factors	DF	Adj SS	Adj MS	F-Value	P-Value
Model	59	9,85E+14	1,67E+13	2,39	0
Linear	9	4,15E+14	4,61E+13	6,6	0
Learning_rate:	2	2,40E+14	1,20E+14	17,16	0,002
Number_of_neurons:	3	1,11E+14	3,69E+13	5,28	0,058
Number_epochs:	4	6,46E+13	1,61E+13	2,31	0,004
2-Way interactions	26	3,61E+14	1,39E+13	1,99	0
Learning_rate:*Number_of_neurons:	6	2,32E+14	3,87E+13	5,54	0,298
Learning_rate:*Number_epochs:	8	6,72E+13	8,40E+12	1,2	0,715
Number_of_neurons:*Number_epochs:	12	6,17E+13	5,14E+12	0,74	0,203
3-Way interactions	24	2,09E+14	8,70E+12	1,25	0,203

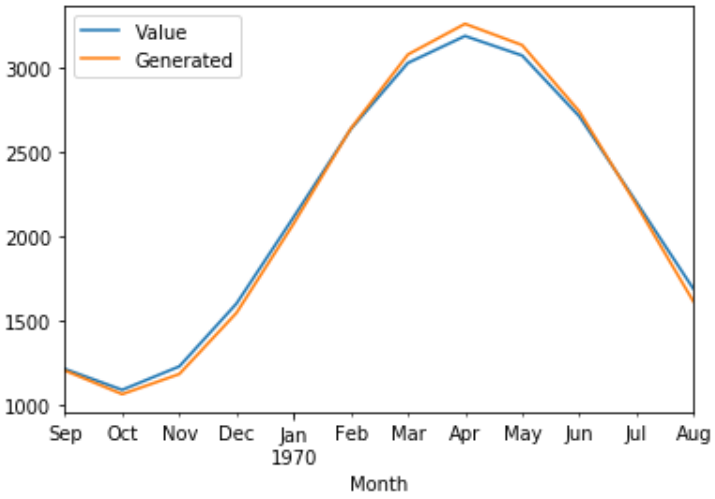
A továbbiakban érdemes lehet megvizsgálni, hogy melyek azok a paraméterek, amikkel futtatva a mesterséges intelligenciát a lehető legjobb tesztelési eredményeket kaptam. Ennek a

táblázatos összefoglalását mutatja be az alábbi tizennegyedik táblázat. Fontos észrevétel véleményem szerint, hogy itt a legjobb eredmények a paraméterek szempontjából rendkívül heterogén képet mutatnak, szemben az előző adatsoron tapasztaltakhoz. Továbbá meg kell állapítani, hogy az itt kapott hibaeredmények egy nagyságrenddel magasabb értékeket vesznek fel, még az arányosított mutatók esetében is. Véleményem szerint ez azért fordulhat elő, mivel a vizsgált paraméter beállítások között nincsen megfelelő kombináció, amivel biztosítani lehetne, hogy a mesterséges intelligencia megtanulja az adatsori sajátosságokat.

14. táblázat *A második kísérletsorozat legjobb eredményei mutatókkal*

Mape	Mase	Mrae	Mse	Rmspe	Tanulási ráta	Neuronok száma	Tanítási iterációk száma
0,020439	0,118889	0,194091	2158,3	0,000575	0,001	100	3000
0,016412	0,106812	0,149429	2815,4714	0,000705	0,001	10	5000
0,024175	0,14442	0,209532	3099,30261	0,000835	0,01	10	5000
0,024844	0,143395	0,163031	3297,03822	0,000872	0,001	50	5000
0,026447	0,152322	0,232912	4531,3614	0,001315	0,001	500	4000
0,034545	0,200316	0,305356	6510,00111	0,001934	0,1	50	3000
0,035302	0,187556	0,22258	6895,60847	0,002278	0,001	10	5000
0,028796	0,211764	0,344361	8016,15381	0,00107	0,01	10	2000
0,046111	0,24043	0,29729	9981,36552	0,003467	0,01	50	4000
0,039166	0,250037	0,391331	10093,9529	0,001994	0,001	10	3000
0,042535	0,225035	0,285238	10215,5958	0,003269	0,01	100	5000
0,043087	0,260484	0,300286	10417,5088	0,002501	0,001	100	4000
0,051474	0,282881	0,358813	11782,4135	0,003708	0,001	500	4000
0,045738	0,29245	0,46094	14219,9509	0,002698	0,01	50	4000

A legjobb modell előrejelzési eredményének és a tényleges adatoknak az összehasonlítását az alábbi harmincegyedik ábra mutatja be. Az ábrán jól látható, hogy a mesterséges intelligencia képes volt valamennyire megtanulni az adatsori sajátosságokat, azonban nem teljes egészében. Ez azért nem kielégítő, mivel az adatsor zajtól mentes és hasonlóan hibátlanul kellett volna megtanulnia a rendszernek az idősort, mint a szinusz hullám esetében.



31. ábra *A második kísérletsorozat legjobb eredménye*

3.3.1. Tézisek vizsgálata

A kísérletek fényében az alábbi következtetéskere jutottam a tézisekkel kapcsolatban:

- ***H1:*** A vizsgálat fókuszában álló paraméterek vannak a legnagyobb ráhatással a modellre.
  Eredmény: ***Elvetve***
  Indoklás: Nem sikerült az állítást igazolni, mivel a regresszió analízis által produkált hiba a normalitás teszten elbukott, így valószínűleg egy, vagy több magyarázó változó hiányozhat.
- ***H2:*** Az egy rejtett réteggel bíró LSTM háló képes megtanulni úgy az idősori sajátosságokat, hogy a kimeneti hiba csak fehérzaj legyen.
  Eredmény: ***Elfogadva***
  Indoklás: Az eredmények elemzése során láthattuk, hogy volt olyan neuronháló alapú előrejelzési modell, amely produkált olyan eredményt, amelynek esetében a generált előrejelzés által produkált hiba adatsor a fehérzaj tesztnek megfelelt.
- ***H3:*** Az epoch szám befolyásolja leginkább a modell pontosságát.
  Eredmény: ***Elvetve***
  Indoklás: Az ANOVA eredményei alapján kijelenthető, hogy ez az állítás nem igaz.
- ***H4:*** A tanulási ráta és a rejtett rétegben lévő LSTM blokkok számának együttes hatása határozza meg leginkább az elérhető pontosságot.
  Eredmény: ***Elfogadva***
  Indoklás: Az ANOVA eredményei alapján kijelenthető, hogy ez az állítás igaz.
- ***H5:*** A paraméterek önálló ráhatása a modell pontosságára számottevőbb, mint ahogy a paraméterek páronként befolyásolják a pontosságot.

  Eredmény: ***Elfogadva***
  Indoklás: Az ANOVA eredményei alapján kijelenthető, hogy ez az állítás igaz.
- ***H6:*** Különböző tulajdonságokkal bíró idősorokhoz, eltérő paraméter beállításokkal bíró neurális háló vezet eredményre .

Eredmény: ***További vizsgálatok szükségesek***
Indoklás: A kísérletek alapján begyűjtött információk alapján nem lehet egyértelműen kijelenteni, azonban az eredmények az állítás helyességét sejtetik.

3.4. További vizsgálati lehetőségek

Az eredményekből egyértelműen kivehető, hogy a terület feltárása korántsem ért a végére, még számos megoldandó kérdés van ezen a tudományos ágon. Ellenben az eredményekből tisztán kivehető, hogy egy rendkívül ígéretes és érdekes vizsgálatokat lehetne még lefolytatni a témával kapcsolatban. A kutatás fókuszában jelenleg a hosszútávú előrejelzések álltak, mivel nagy általánosságban erre van a legnagyobb igény a különböző logisztikához köthető iparágakban. Ezen a szakterületen belül feltétlenül érdemes tovább vizsgálni annak a lehetőségét, hogy komplexebb zajos havi aggregátsággal bíró adatsorok esetében milyen pontos eredményeket lehetne elérni az előrejelzések esetében mesterséges intelligenciákkal.

A további kutatásoknak véleményem szerint egy sarkalatos pontjának kéne lennie, hogy fel legyen tárva egy úgynevezett „neuro identifikáció" ami szigorúan a kereslettervezéshez lenne köthető. Ennek célja az lenne, hogy egy olyan tudásbázist hozzunk létre, amely irány mutatóként szolgálhatna annak tekintetében, hogy milyen adatsor esetében, milyen struktúrájú neurális hálót érdemes használni.

Fontos lenne továbbá feltárni annak a lehetőségét, hogy a jelenlegi eszközrendszer milyen opciókat biztosít a rövid távú kereslet előrejelzés terén. A heti szinten aggregát adatok esetében az idősori szisztematizmusokat sokkal jobban elfedi az idősorban tapasztalható zaj, ennél fogva sokkal nehezebb precíz előrejelzéseket készíteni, ellenben a vállalati szférában erre annál nagyobb igény mutatkozik.

Egy érdekes vizsgálati terület lehet továbbá, még az idősorok dekomponálása és azok vizsgálata. Az idősorok multiplikatívan, vagy additívan elemeikre bonthatóak. Elképzelhető, hogy a dekomponált elemeket külön-külön megtanítva egy mesterséges intelligenciának és az elemekre külön előrejelzéseket készítve, majd ezeket a megfelelő struktúra szerint összegezve jobb eredményt kapunk, mintha a teljes idősort próbálnánk megtanítani a hálónak.

Az utolsó és egyben legsarkalatosabb pontja a későbbi vizsgálatoknak véleményem szerint más LSTM cella típusok vizsgálata. Számos, a vizsgálat tárgyát képzőnél jóval komplexebb LSTM típus is létezik, amikkel más területeken, mint például a hang- és videófeldolgozás rendkívüli eredményeket értek el. Ilyen cellák például: Grid LSTM, vagy éppenséggel a Time Frequency LSTM cellák. Véleményem szerint ezeket a típusokat a keresleti idősorok kontextusában mindenféleképpen érdemes megvizsgálni.[37]

Összefoglalás

A dolgozatom első felébe feltártam, hogy milyen alapjai vannak a neurális hálóknak, hogy az egyes architektúrák milyen tulajdonságokkal bírnak és, hogy milyen típusú mesterséges intelligenciákkal érdemes kísérletezni amennyiben ilyen módon akarunk kereslettervező rendszereket konstruálni. A fejezetben továbbá bemutattam a munkám során használt könyvtárakat, valamint azok legfontosabb attribútumait.

A második fejezetben a rendelkezésre álló irodalom alapján megpróbáltam a kereslettervezés jelenhelyzetét feltárni, illetve jellemezni. A munka során ismertettem a hagyományos előrejelző módszerek lépéseit, kiemelve az egyes fázisok nehézségeit és felhívva a figyelmet arra, hogy a szakterületen alapjában véve nincsenek kőbe vésett iránymutató szabályok. A fejezet második részében megpróbáltam feltárni, hogy a mesterséges intelligenciák milyen lehetőséget kínálnak az egyes fázisok helyettesítésére, valamint támogatására.

A harmadik fejezetben az irodalomkutatás fényében megpróbáltam olyan mesterséges intelligenciákat alkotni, amelyek alkalmasak lehetnek az idősorok jövőjének előrejelzésére. A megfelelő eredményeket követően egy teljes faktoriális kísérlettervet raktam össze, amelynek eredményéül azt reméltem, hogy a mesterséges intelligenciák legfontosabb paraméter összefüggéseit megismerhetem az idősor előrejelzési feladatok esetén. A kísérletek lefolytatását után az eredmények statisztikai vizsgálata következett, amelynek fényében kijelöltem a véleményem szerint legfontosabb kutatási csapásirányokat a jövőre való tekintettel.

Az eredmények fényében kijelenthető, hogy a mesterséges intelligenciák segítségével lehet kereslet előrejelző rendszereket készíteni, bár az még további kutatások feladata megválaszolni, hogy a hagyományos módszerekkel, komplex feladatok esetében mennyire képesek felvenni a versenyt.

Ábrajegyzék

Táblázatjegyzék

Hivatkozások, felhasznált irodalom

[1] Sebastian Raschka - *Python Machine Learning* 2015 Packt Publishing

[2] Horváth Gábor – *Neurális hálózatok* Panem Könyvkiadó Kft., Budapest, 2006

[3] Gebhard Kirchgässner, Jürgen Wolters, Uwe Hassler – *Introduction to modern time series analysis* Springer Heidelberg New York Dordrecht London 2012

[4] Daniel Graupe - *Principles Of Artificial Neural Networks 2nd edition* University of lllinois, Chicago, USA, 2007

[5] Diederik P. Kingma, Jimmy Lei Ba - *Adam: A Method For Stochastic Optimization* Published as a conference paper at ICLR 2015

[6] http://colah.github.io/posts/2015-08-Understanding-LSTMs/ 2017.09.03 21:00

[7] Simon Haykin - *Neural Networks - A Comprehensive Foundation* 1999 by Pearson Education

[8] Yoshua Bengio, Patrice Simard, Paolo Frasconi - *Leaming Long-Term Dependencies with Gradient Descent is Difficult* IEEE Transactions On Neural Networks, Vol. 5, No. 2, March 1994

[9] Sepp Hochreiter – *Long Short-Term Memory* Neural Computation 9(8) 1735-1780, 1997

[10] Zachary C. Lipton, John Berkowitz - *A Critical Review of Recurrent Neural Networks for Sequence Learning* arXiv:1506.00019v4, 2015

[11]https://www.forbes.com/sites/quora/2016/08/25/this-is-what-makes-keras-different according-to-its-author/#6153c4366cfa 2017.09.20 20:19

[12] Martin Abadi - *TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems* Preliminary White Paper, November 9, 2015

[13] Martin Abadi - *TensorFlow: A System for Large-Scale Machine Learning* November 2–4, 2016 • Savannah, GA, USA

[14] Dr. Bóna Krisztián – *Kereslet- és Készlettervezés előadás diasorozat* 2016, Budapest

[15] http://www.scholarpedia.org/article/Neural_Filtering 2017.10.15 10:17

[16] http://benanne.github.io/2015/03/17/plankton.html 10.23 19:50

[17] Kamer Kayaer - *Medical Diagnosis on Pima Indian Diabetes Using General Regression Neural Networks* Besiktas, Istanbul 2003

[18] Chris Chatfield - *Time series forecasting with neural networks a comparative study using the airline data* Appl. Statist. 1998

[19] Felix A. Gers - *Learning Precise Timing with LSTM Recurrent Networks* Journal of Machine Learning Research 3 (2002)

[20] Dongxu Shao, Tianyou Zhang - *Time Series Forecasting on Engineering Systems Using Recurrent Neural Networks* Advanced Data Mining and Applications 2016

[21] Daniel L. Marino - *Building Energy Load Forecasting using Deep Neural Networks* 2016 IEEE

[22]http://www.jakob-aungiers.com/articles/a/LSTM-Neural-Network-for-Time-Series Prediction 2017.09.01 9:38

[23] Felix A. Gers - *Applying LSTM to Time Series Predictable Through Time-Window Approaches* Neural Nets WIRN Vietri-01 pp 193-20

[24] Hardik Goel - *Multivariate Aviation Time Series Modeling: VARs vs. LSTMs*

[25] https://en.wikipedia.org/wiki/Principal_component_analysis 2017.10.20 15:42

[26] https://pvirie.wordpress.com/2016/03/29/linear-autoencoders-do-pca/ 2017.10.20 16:01

[27] http://karpathy.github.io/2015/05/21/rnn-effectiveness/ 2017.10.02 12:35

[28] Klaus Greff - *LSTM: A Search Space Odyssey* arXiv:1503.04069v2 [cs.NE] 4 Oct 2017

[29] Felix A. Gers – *Recurrent Nets that Time and Count* IDSIA

[30] Dr. Wenzel Klára – *Kísérlettervezés Taguchi Módszerrel* 2013

[31] Johanyák Zsolt Csaba – *Bevezetés a kísérletmódszertanba* 2002

[32] Dr. Fidy Judit, Dr. Makara Gábor *Biostatisztika* (2005) InforMed 2002 Kft.

[33]http://blog.minitab.com/blog/adventures-in-statistics-2/why-you-need-to-check-your-residual-plots-for-regression-analysis 2017.10.24 19:30

[34]http://blog.minitab.com/blog/adventures-in-statistics-2/how-important-are-normal-residuals-in-regression-analysis 2017.10.24 19:30

[35] https://hu.wikipedia.org/wiki/Varianciaanal%C3%ADzis 2017.10.24 19:30

[36]http://blog.minitab.com/blog/adventures-in-statistics-2/how-to-interpret-regression-analysis-results-p-values-and-coefficients 2017.10.24 19:30

[37] Yuzhou Liu - *Time And Frequency Domain Long Short-Term Memory For Noise Robust Pitch Tracking* The Ohio State University 2016