Predviđanje protoka vozila u mreži prometnica

Stanić, Andrea

SVEUČILIŠTE U ZAGREBU

PRIRODOSLOVNO–MATEMATIČKI FAKULTET

MATEMATIČKI ODSJEK

Andrea Stanić

PREDVIĐANJE PROTOKA VOZILA U MREŽI PROMETNICA

Diplomski rad

Voditelj rada:
dr. sc. Tomislav Šmuc

Zagreb, studeni, 2018.

Ovaj diplomski rad obranjen je dana _____ pred ispitnim povjerenstvom u sastavu:

1. _____, predsjednik
2. _____, član
3. _____, član

Povjerenstvo je rad ocijenilo ocjenom _____.

Potpisi članova povjerenstva:

1. _____
2. _____
3. _____

Sadržaj

Uvod

Broj vozila koja sudjeluju u prometu se brzo povećava i to dovodi do brojnih problema. Od gužvi, dodatnih troškova zbog kašnjenja, pa sve do zagađenja okoliša. S porastom prometa pojavila se potreba za modeliranjem protoka prometa i predviđanjem budućih stanja. Inteligentni transportni sustavi ITS (engl. *Intelligent Transportation System*) i napredni sustavi za upravljanje prometom ATMS (engl. *Advanced Traffic Management System*) su sustavi koji imaju integrirane informacijske i komunikacijske te druge tehnologije koje iskorištavaju za bolje reagiranje na incidente u prometu, obavještavanje putnika o stanju u prometu, predviđanje gužvi i zastoja. Od posebnog značaja za neke od navedenih problema je upravo predviđanje stanja u prometu s velikom preciznošću ([19]).

Prije nekoliko desetaka godina kada je započeto s modeliranjem protoka prometa računalni kapaciteti su bili značajno manji nego danas te je bilo manje povijesnih podataka koje je bilo moguće iskoristiti za izgradnju modela. S obzirom na to da se protok prometa može shvatiti kao vremenski niz neki od jednostavnih modela su upravo modeli vremenskih nizova. S povećanjem računalnih kapaciteta i količine dostupnih podataka se povećala i složenost modela, pa su danas dostupni različiti model koji koriste duboke neuronske mreže.

U ovom radu su uz teoretski uvod u vremenske nizove i duboko učenje navedeni i različiti pristupi za modeliranje protoka prometa. Na kraju su iskorišteni podaci koji simuliraju protok prometa na ulicama Varšave i napravljen je STARMA model te jednostavna LSTM neuronska mreža, čiji su rezultati uspoređeni s rezultatima dobivenim drugim metodama.

Poglavlje 1

Opis problema

1.1 Važnost rješavanja problema

Predviđanje protoka vozila je od velike važnosti za svakodnevni život građana, urbanističke planove i industriju.

Kroz godine broj prodanih vozila ima velik porast. Tako je između 1990. i 1999. godišnje u prosjeku bilo prodano 39.2 milijuna vozila, dok se predviđa da će se do kraja 2018. godine prodati 81.5 milijuna vozila[1]. Još jedan od problem u prometu su i velike gužve. Prema INRIX-ovom istraživanju za 2017. godinu[2] vozači u Los Angelesu godišnje u prosjeku provedu 102 sata u gužvama (12% ukupnog vremena koje provedu u vožnji), dok u Europi najviše vremena u gužvama provedu u Londonu, čak 74 sata (13% ukupnog vremena koje provedu u vožnji). Prema [17] zagađivači u prometu od interesa za zdravlje uključuju dušikov dioksid, ugljični monoksid, crni dim, benzen i metale (poput olova). Također navode da su istraživanja pokazala kako onečišćenje od prometa utječe na zdravstveno stanje stanovništva. Od toga da povećava smrtnost, utječe na razvoj alergijskih oboljenja, trudnoću, astmu i ostale bolesti vezane za respiratorni sistem. Kako ne bi došlo do stalnih gužvi, sporog prometa i velikog zagađenja zraka bitno je uložiti vrijeme, znanje i dostupne informacije u izgradnju prometne infrastrukture i urbanističkih planova, a tome uvelike pomaže ako znamo kako se promet odvija.

S obzirom na sve navedeno i činjenicu da promet ima bitnu ulogu u svakodnevnom životu važno je raditi na problemu predviđanja protoka prometa.

[1] https://www.statista.com/statistics/200002/international-car-sales-since-1990/

[2] http://inrix.com/press-releases/scorecard-2017/

1.2 Kompleksnost problema

Cilj je na osnovu povijesnih podataka i trenutnog stanja predvidjeti kakav će protok prometa biti u idućih nekoliko sekundi, minuta ili sati. Prilikom izrade model za predviđanje protoka vozila posebno je potrebno obratiti pažnju na izazove koji se prisutni u prometu. Prema [26] postoji 10 izazova koji se javljaju prilikom izrade modela, a ovdje ću ukratko navesti neke od njih.

Na promet utječu brojni faktori, poput prometnih nesreća, radova i vremenskih uvjeta. S obzirom na to, model na kojem radimo treba biti robustan na promjene. Jedan način kako dobiti robusniji model je da koristimo više izvora podataka (na primjer vremenske prognoze). Tu dolazimo do problema kako spojiti različite izvore podataka u jedan skup.

Postoje bitne razlike kada se rade predviđanja na autocestama, brzim cestama i u urbanim naseljima. U urbanim naseljima postoji znatno više prometne signalizacije i raskrižja, skretanja koja dodatno utječu na promet. Osim toga utjecaj ima i gustoća naseljenosti pojedenih dijelova naselja.

Prilikom izrade modela potrebno je odrediti i vremenski interval koji će se promatrati. U slučaju da su intervali jako mali, u podacima možemo imati jako puno šuma, pa se u takvim situacijama savjetuje agregacija podataka i/ili primjena algoritama za smanjenje šuma.

Pažnju treba posvetiti i vremensko-prostornim odnosima i načinu na koji će se oni modelirati. Treba voditi računa o tome da radili se o mreži prometnica te konstantom protoku podataka.

Rješavanju problema se može pristupiti na više načina i uobičajeno se za krajnji model uzima onaj koji ima najbolju točnost. Jedno od pitanja koje se javlja je i trebamo li različite modele usporediti ili kombinirati njihove rezultate. Često se kombinacijom različitih modela dobiju bolja rješenja, a o tome kako će se krajnji rezultat dobiti i je li kombinacija potrebna odlučuje osoba koja izrađuje model. Osim što krajnji model treba imati dobru točnost, ponekad je poželjno da je on i interpretabilan te da objašnjava pojedine fenomene koji se javljaju.

Različita rješenja je moguće dobiti korištenjem različitih modela. Problem na koji nailazimo je usporedba tih modela. Naime, različiti modeli imaju različitu terminologiju i koriste različite eksperimentalne postavke, pa nije uvijek moguće pronaći metriku koju bi koristili za njihovu usporedbu.

1.3 Prikupljanje podataka

Postoji više vrsta senzora pomoću kojih se mogu prikupiti podaci. Neki od njih su: induktivna petlja, magnetni senzori, laserski senzori, kamere, optički senzori, akustični senzori. Osim toga podaci se mogu dobiti i putem GPS.

Senzori se mogu podijeliti na one s fiksnom pozicijom i na senzore s promjenjivom pozicijom. Glavna prednost senzora s fiksnom pozicijom je da su takvi detektori pouzdani i da prate sav promet koji prolazi pored njih. Nedostatak takvih senzora je što ne znamo točnu putanju vozila i stoga može biti teško odrediti odnose između različitih segmenata promatrane prometnice ili mreže prometnica. Podaci prikupljeni sa senzora s promjenjivom pozicijom su prikupljeni korištenjem GPS-a ugrađenog u vozila i pametne uređaje. Podaci prikupljeni pomoću takvih senzora nam omogućuju da uočimo rute i povezanost između različitih segmenata prometnice ili mreže prometnica. Nedostatak kod ovakvog pristupa je to što nemamo podatke za sva vozila na određenoj lokaciji i postoji mogućnost da ono što imamo nije reprezentativno i da ne opisuje dovoljno dobro stanje u prometu.

Više detalja o senzorima možete pronaći u [19, pog. 2].

1.4 Mogući pristupi

Pristupe rješavanja problema možemo svrstati u dvije kategorije: pristupi vođen znanjem (engl. *knowledge-driven approach*) i pristupi vođen podacima (engl. *data-driven approach*) ([15]).

Kod pristupa vođenog znanjem glavnu ulogu imaju eksperti i cilj je iskoristi domensko znanje i znanje stečeno iskustvom. Jedan takav pristup je teorija čekanja (engl. *queuing theory*) koja se može primijeniti na redove čekanja na crvenim svijetlima, znakovima zaustavljanja. Kako ne bi došlo do velikih gužvi i zastoja potrebno je dobro razumijevanje od strane inženjera ([18]) i s obzirom na to za taj tip problema dobro je primjeniti pristup vođen znanjem. Zbog pravila koja se javljaju u takvim modelima oni su lakši za interpretaciju. U pristupu vođenom podacima glavnu ulogu imaju podaci. Tu se ubrajaju modeli vremenskih nizova, modeli dubokog učenja i upravo takvim modelima ću se baviti u nastavku.

Pristupe možemo razlikovati i po tome je li korišteni algoritam *offline* ili *online*. Za algoritam kažemo da je *offline* ako ima pristup svim podacima i njih koristi kako bi odredio parametre. Za razliku od toga, *online* algoritmi imaju stalni dotok podataka i kada dobiju novi primjer parametri se prilagođavaju.

Ovisno o problemu kojeg promatramo predviđanja mogu biti kratkoročna ili dugoročna. U slučaju kada nas zanima kakvo će stanje u prometu biti u idućih nekoliko sekundi ili minuta (do 30 minuta) koristit ćemo kratkoročna predviđanja. U suprotnom koristimo dugoročna predviđanja.

Ako nas zanima protok prometa na točno određenoj lokaciji, onda će se promatrati univarijatni modeli. S druge strane, ako nas zanima protok u mreži prometnica, onda ćemo se orijentirani na multivarijatne modele.

Poglavlje 2

Vremenski nizovi

2.1 Osnovne definicije i pojmovi

Kako bi objašnjenja modela vremenskih nizova bila jasnija prvo navodim neke osnovne definicije i svojstva vremenskih nizova. Više detalja možete pronaći u [4] i [24].

Definicija 2.1.1. ***Vremenski niz*** *je skup opažanja x_t, zabilježenih u trenutku t.*

Definicija 2.1.2. ***Model vremenskih nizova*** *za opažene podatke $\{x_t\}$ je specifikacija zajedničke distribucije (ili eventualno samo očekivanja i kovarijance) niza slučajnih varijabli $\{X_t\}$ koje mogu poprimiti vrijednosti $\{x_t\}$.*

Definicija 2.1.3. *Neka je $\{X_t\}$ vremenski niz za koji vrijedi $E(X_t) < \infty$.* ***Funkcija očekivanja*** *danog vremenskog niza je definirana s*

$$\mu_X(t) = E(X_t). \tag{2.1}$$

Kovarijacijska funkcija *od $\{X_t\}$ je*

$$\gamma_X(r, s) = Cov(X_r, X_s) = E[(X_r - E(X_r))(X_s - E(X_s)] \tag{2.2}$$

Definicija 2.1.4. *Vremenski niz $\{X_t\}$ je* ***(slabo) stacionaran*** *ako je:*

(i) $\mu_X(t)$ *nezavisan o* t

(ii) $\gamma_X(t, t + h)$ *nezavisno za svaki* t *i* h

Korolar 2.1.5. *Svojstva autokorelacijske funkcije:*

(i) $\gamma(0) \geq 0$

(ii) $|\gamma(h)| \leq \gamma(0)$*, za sve* h

(iii) $\gamma(h) = \gamma(-h)$*, za sve* h*.*

Dokaz. Dokaz pogledajte u [24, str. 41.]. □

Kada promatramo slabo stacionaran niz $\{X_t\}$ zbog (ii) iz prethodne definicije kovarijcijsku funkciju možemo prikazati kao funkciju jedne varijable

$$\gamma_X(h) = \gamma_X(h, 0) = \gamma_X(h, h + t)$$

Prilikom određivanja modela posebnu ulogu ima uzoračka autokorelacijska funkcija i uzoračka autokovarijacijska funkcija.

Definicija 2.1.6. *Neka su* $x_1, \ldots, x_n$ *opaženi uzorci vremenskog niza.* ***Uzoračko očekivanje*** *od* $x_1, \ldots, x_n$ *je*

$$\overline{x} = \frac{1}{n} \sum_{t=1}^{n} x_t. \tag{2.3}$$

Uzoračka autokovarijacijska funkcija *je dana s*

$$\hat{\gamma}(h) := \frac{1}{n} \sum_{t=1}^{n-|h|} (x_{t+|h|} - \overline{x})(x_t - \overline{x}), \qquad -n < h < n. \tag{2.4}$$

Uzoračka autokorelacijska funkcija *je dana s*

$$\hat{\rho}(h) := \frac{\hat{\gamma}(h)}{\hat{\gamma}(0)}, \qquad -n < h < n. \tag{2.5}$$

Definicija 2.1.7. ***Vremenski niz*** $\{X_t\}$ ***je jako stacionaran*** *ako:*

$$(X_1, \ldots, X_n)' \overset{D}{=} (X_{1+h}, \ldots, X_{n+h})' \tag{2.6}$$

za sve cjelobrojne h *i* $n \geq 1$*. (*$\overset{D}{=}$ *označava da su jednako distribuirani).*

Da bi vremenski niz bio jako stacionaran on mora zadovoljavati stroži uvjet nego za slabu stacionarnost. U nastavku kada piše da je vremenski niz stacionaran, podrazumijeva se da je slabo stacionaran.

Definicija 2.1.8. ***Bijeli šum (engl. white noise (WN))*** $\{X_t\}$ *je niz nekoreliranih slučajnih varijabli, s očekivanjem* 0 *i varijancom* σ^2*. Oznaka:* $\{X_t\} \sim WN(0, \sigma^2)$.

U nastavku kratko uvodim pojam najboljeg linearnog procjenitelja i Woldovu dekompoziciju koja će biti spomenuta u primjenama.

Ako imamo puno prošlih opservacija $X_m, \ldots, X_0, \ldots, X_n \quad (m < 0)$ dobro je iskoristi ih za pronalaženje najboljeg linearnog procjenitelja za X_{n+h} u terminima $1, X_m, \ldots, X_0, \ldots, X_n$. Taj prediktor možemo označiti s $P_{m,n}X_{n+h}$. U slučaju da je m velik dani procjenitelj možemo aproksimirati na slijedeći način:

$$\tilde{P}_n X_{n+h} = \lim_{m \to -\infty} P_{m,n} X_{n+h}. \tag{2.7}$$

$\tilde{P}_n$ nazivamo operator predviđanja s obzirom na beskonačnu prošlost, $\{X_t : -\infty < t \leq \infty\}$, dok P_n nazivamo operator predviđanja s obzirom na konačnu prošlost, $\{X_1, \ldots, X_n\}$. Više o ovim operatorima možete pročitati u [24, pog. 2.5].

Teorem 2.1.9. ***Woldova dekompozicija***
Ako je $\{X_t\}$ nedeterministički stacionarni vremenski niz, tada je

$$X_t = \sum_{j=0}^{\infty} \psi_j Z_{t-j} + V_j, \tag{2.8}$$

gdje je

(1) $\psi_0 = 1$ *i* $\sum_{j=0}^{\infty} \psi_j^2 \leq \infty$,

(2) $\{Z_t\} \sim WN(0, \sigma^2)$,

(3) $Cov(Z_s, V_t) = 0$ *za sve* s *i* t,

(4) $Z_t = \tilde{P}_t Z_t$ *za sve* t,

(5) $V_t = \tilde{P}_s V_t$ *za sve* s *i* t,

(6) $\{V_t\}$ *je deterministički.*

Definicija 2.1.10. ***Klasnična dekompozicija*** *modela je dana s*

$$X_t = m_t + s_t + Y_t, \qquad t = 1, \ldots, n \tag{2.9}$$

pri čemu je $EY_t = 0$, $s_t = s_{t+d}$, $\sum_{j=0}^{d} s_j = 0$ *i* d *je period.*

Trend je označen s m_t, s_t je sezonalna komponenta, a Y_t je slučajni stacionaran niz.

Cilj analize vremenskih nizova je shvatiti kako su podaci generirani i iskoristiti to za predviđanje budućih vrijednosti. Analiza vremenskih nizova se sastoji od nekoliko koraka. Prvi korak je da se podaci prikažu na grafu. Iz grafa treba uočiti kakav je trend, je li sezonalnost prisutna, postoje li nagle promjene u ponašanju i postoje li vrijednosti koje jako

odskaču. Zatim se iz podataka uklanjaju trend i sezonalnost, kako bi dobili stacionarne reziduale. Ponekad je potrebno napraviti transformaciju podatka prije nego se procjene trend i sezonalnost. Sljedeći korak je pomoću različitih statistika i uzoračke autokorelacijske funkcije odrediti model za reziduale. Predviđanje se dobije tako da predvidimo reziduale koje smo modelirali te primijenimo inverzne transformacije koje smo prethodno radili, s ciljem da dobijemo predviđanje originalnoga niza.

Trend m_t se može ukloniti pomoću nekoliko metoda. Jedna od njih je modeliranje trenda polinomom. Pretpostavimo da je trend oblika $m_t = a_0 + a_1 t + a_2 t^2$. Cilj je pronaći koeficijente a_0, a_1, a_2 koji minimiziraju sumu $\sum_{t=1}^{n}(x_t - m_t)^2$, a to, na primjer, možemo pomoću metode najmanji kvadrata. Druga metoda je diferenciranje. Operator diferenciranja $j-$tog reda je definiran rekurzivno s

$$\nabla X_t = X_t - X_{t-1}, \qquad \nabla^j X_t = \nabla(\nabla^{j-1} X_t). \tag{2.10}$$

Ako je trend polinom stupnja k, $m_t = \sum_{j=1}^{k} c_j t^j$, tada primjenom operatora diferenciranja $k-$tog reda dobijemo stacionarni niz $\nabla^k X_t = k!c_k + \nabla^k Y_t$ s očekivanjem $k!c_k$. Ako imamo podatke $\{x_t\}$ na njih primjenjujemo operator diferenciranja sve dok ne dobijemo niz koji može biti modeliran kao realizacija stacionarnog procesa.

Prilikom procjene sezonalne komponente imamo dva moguća slučaja. Prvi slučaj je kada je period d paran, odnosno $d = 2q$. U tom slučaju prvo aproksimativno određujemo trend pomoću sljedeće formule

$$\hat{m}_t = (0.5x_{t-q} + x_{t-q+1} + \cdots + x_{t+q-1} + 0.5x_{t+q})/d, \qquad q < t \leq n - q. \tag{2.11}$$

Drugi slučaj je kada je d neparan, $d = 2q + 1$. Tada trend aproksimiramo kao

$$\hat{m}_t = \frac{1}{2q+1} \sum_{j=-q}^{q} x_{t-j}, \qquad q + 1 \leq t \leq n - q. \tag{2.12}$$

Nadalje, neka je w_k prosjek skupa $\{(x_{k+jd} - \hat{m}_{k+jd}), q < k + jd \leq n - q\}$. S obzirom na to da suma prosječnih devijacija w_k nije nužno jednaka nuli, sezonalnu komponentu procjenjujemo kao

$$\hat{s}_k = w_k - \frac{1}{d} \sum_{i=1}^{d} w_i, \ k = 1, \ldots, d, \text{ i } \hat{s}_k = \hat{s}_{k-d}, \text{ za } \quad k > d. \tag{2.13}$$

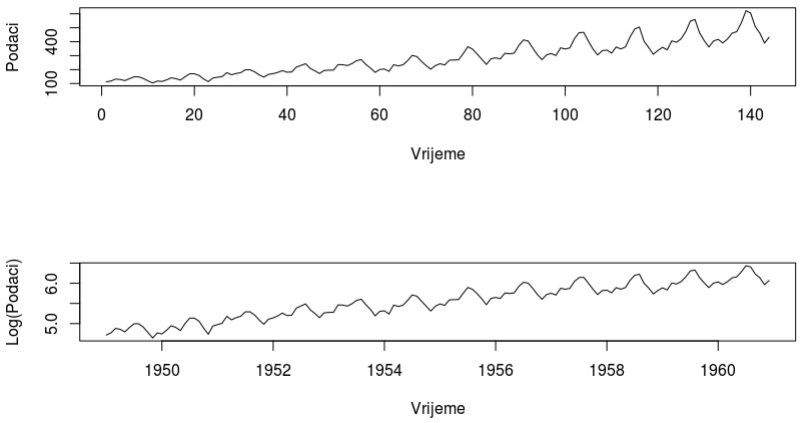
Kada smo napravili procjenu sezonalne komponente nju oduzimao od originalnih podataka i dobijemo desezonalizirani niz $d_t = x_t - \hat{s}_t, t = 1, \ldots, n$. Kako bi imali parametarsku reprezentaciju trenda koju onda možemo koristiti za predviđanje i simulacije, na desezonaliranom nizu, pomoću prethodno opisanih metoda, radimo procjenu trenda $\hat{m}_t$. Oduzimanjem trenda $\hat{m}_t$ i sezonalne komponente $\hat{s}_t$ dobijemo procijenjeni šum

$$\hat{Y}_t = x_t - \hat{m}_t - \hat{s}_t, \quad t = 1, \ldots, n. \tag{2.14}$$

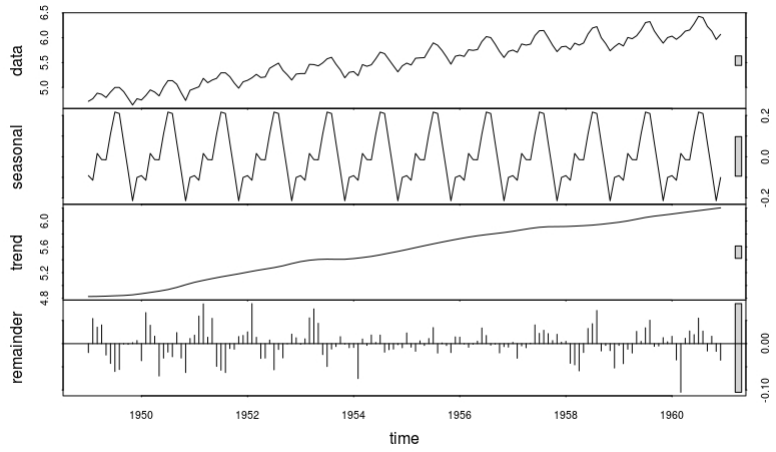
Sljedeći korak je provjeriti razlikuje li se dobiveni procijenjeni šum $\{\hat{Y}_t\}$ značajno od modela koji je reprezentiran pomoću nezavisnih i jednako distribuiranih slučajnih varijabli čije je očekivanje nula. Takav vremenski niz zovemo iid šum (engl. *independent and identically distributed*). Ako se razlikuju, onda $\{\hat{Y}_t\}$ modeliramo kao stacionarni vremenski niz.

U nastavku ću na primjeru koji je preuzet s [12] pokazati kako se u praksi provodi analiza vremenskog niza. Analiza primjera je napravljena u programskom jeziku R.

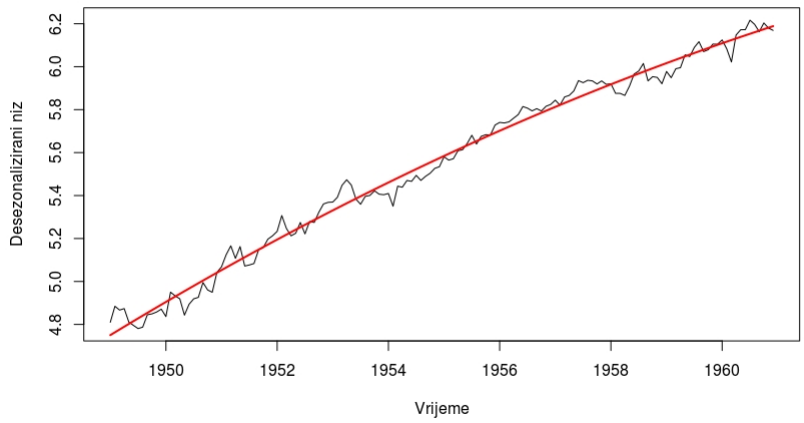
Primjer 2.1.11. *Imamo podatke o mjesečnom broju putnika u međunarodnom zračnom prometu u razdoblju od siječnja 1949. do prosinca 1960. godine. Prvo pomoću funkcije* $ts(pivo, start = 1949, frequency = 12)$ *napravimo objekt koji je vremenski niz. Kada prikažemo podatke vidimo da varijanca raste s vremenom i zato logaritmiramo podatke. Prikaz originalnih i logaritmiranih podataka ja na slici 2.1. Zatim za dekompoziciju vremenskog niza koristimo funkciju stl iz R-a. Kada prikažemo rezultat navedene funkcije na dobivenom grafu (slika 2.2) vidimo logaritmirane podatke, sezonalnu komponentu, trend i ostatak kroz vrijeme. Iz prikaza trenda možemo vidjeti da on nije linearan i da se možda može modelirati pomoću parabole 2. stupnja. Funkcija stl daje sezonalnu komponentu, koja se onda oduzima od originalnih podataka kako bi dobili desezonalizirani niz čiji prikaz je na slici 2.3. Sljedeći korak je modeliranje trenda pomoću polinoma. Pretpostavljamo da se može modelirati s polinomom 2. stupnja, ali krećemo s višim stupnjem i onda smanjujemo stupanj polinoma sve dok ne dobijemo da su nam svi koeficijenti statistički značajni. Pokazalo se da je polinom stupnja dva uistinu dobar. Da bi dobili koeficijente koristimo funkciju lm. Na slici 2.3 vidimo desezonalizirani niz s polinomom koji modelira trend. Zatim crtamo graf uzoračke autokorelacijske funkcije kako bi provjerili koreliranost reziduala. Ako su sve korelacije unutar* 95% *pouzdanog intervala, onda možemo pretpostaviti da se radi o bijelom šumu i idući korak je provođenje testova normalnosti. S druge strane, ako to nije slučaj, onda moramo iskorsiti modele kako bi opisali dobivene reziduale. Iz grafa 2.4 vidimo da su greške korelirane i sljedeći korak je modeliranje ostataka. Odabrat ćemo onaj model koji ima najmanji AIC. Pokazuje se da je to ARMA(5,6) model. Više o modelima i kako se oni biraju će biti objašnjeno kasnije.*



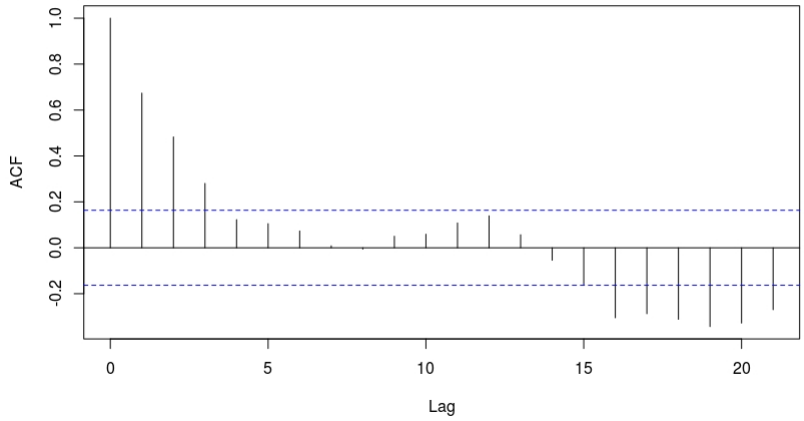
Slika 2.1: Originalni i logaritmirani podaci



Slika 2.2: Dekompozicija podataka



Slika 2.3: Desezonalizirani niz i procjenjeni trend



Slika 2.4: ACF graf

G. E. P. Box i G. M. Jenkins su 1976. godine izdali prvo izdanje knjige *Time Series Analysis: Forecasting and Control* ([4]) koja je odigrala značajnu ulogu u razvoju analize vremenskih nizova. To je prva knjiga koja je sadržavala sve postupke za analizu vremenskih nizova. Od specifikacije modela, prilagođavanja parametara, procjena i predviđanja.

Iako su AR(p) i MA(q) procesi bili i prije korišteni, Box i Jenkins su popularizirali upotrebu ARMA(p, q) procesa. Kako bi se upoznali s osnovnim modelima u nastavku navodim definiciju i osnovna svojstva ARMA(p, q) procesa. Nakon toga će biti navedeno još nekoliko procesa te će biti objašnjeno kako su oni korišteni u problemu predviđanja protoka vozila.

2.2 ARMA(p, q) proces

Definicija 2.2.1. *Vremenski niz* $\{X_t\}$ *je* ***ARMA(p,q) proces*** *ako je* $\{X_t\}$ *stacionaran i za svaki t vrijedi*

$$X_t - \phi_1 X_{t-1} - \cdots - \phi_p X_{t-p} = Z_t + \theta_1 Z_{t-1} + \cdots + \theta_q Z_{t-q}, \tag{2.15}$$

gdje je $\{Z_t\} \sim WN(0, \sigma^2)$ *i polinomi* $(1 - \phi_1 z - \cdots - \phi_p z^p)$ *i* $(1 + \theta_1 z + \cdots + \theta_q z^q)$ *nemaju zajedničkih faktora.*

Sažeto to možemo prikazati kao

$$\phi(B)X_t = \theta(B)Z_t, \tag{2.16}$$

$$\text{gdje su} \quad \phi(z) = 1 - \phi_1 z - \cdots - \phi_p z^p, \qquad \theta(z) = 1 + \theta_1 z + \cdots + \theta_q z^q$$

i B je operator pomaka unatrag ($B^j X_t = X_{t-j}, B^j Z_t = Z_{t-j}, j = 0, \pm 1, \pm 2, \dots$).

Vremenski niz $\{X_t\}$ je autoregresivni proces (engl. *autoregressive process*) reda p, AR(p), ako je $\theta(z) \equiv 1$, odnosno proces pomičnih prosjeka (engl. *moving-average process*) reda q, MA(q), ako je $\phi(z) \equiv 1$.

Stacionarno rješenje jednadžbe $\{X_t\}$ iz definicije ARMA procesa **postoji** i **jedinstveno** je ako i samo ako vrijedi $\phi(z) = 1 - \phi_1 z - \cdots - \phi_p z^p \neq 0$ za sve $|z| = 1$.

Definicija 2.2.2. *ARMA(p, q) proces* $\{X_t\}$*je* ***kauzalan ARMA(p,q) proces*** *ili* ***kauzalna funkcija od*** $\{Z_t\}$ *ako postoje konstante* $\{\psi_j\}$ *takve da je* $\sum_{j=0}^{\infty} |\psi_j| < \infty$ *i*

$$X_t = \sum_{j=0}^{\infty} \psi_j Z_{t-j} \quad \textit{za sve } t. \tag{2.17}$$

Kauzalnost je ekvivalentna uvjetu da je

$$\phi(z) = 1 - \phi_1 z - \cdots - \phi_p z^p \neq 0, \text{za sve} \quad |z| \leq 1.$$

graf/model	AR(p)	MA(q)	ARMA(p,q)
ACF	opada	prekine nakon q	opada
PACF	prekine nakon p	opada	opada

Tablica 2.1: Određivanje parametara p i q

Definicija 2.2.3. ***Parcijalna autokorelacija (PACF)*** *ARMA procesa $\{X_t\}$ je funkcija $\alpha(\cdot)$ definirana s*

$$\alpha(0) = 1, \tag{2.18}$$

$$\alpha(h) = \phi_{hh}, \quad h \geq 1, \tag{2.19}$$

gdje je ϕ_{hh} zadnja komponenta u $\boldsymbol{\phi}_h = \Gamma_h^{-1}\gamma_h, \Gamma_h = [\gamma(i-j)]_{i,j=1}^n$ i $\gamma_h = [\gamma(1), \gamma(2), \ldots, \gamma(h)]'$.

Pri određivanju parametara p i q procesa koristimo PACF i ACF graf, a u tablici 2.1 je navedeno kako izgledaju grafovi za pojedine procese.

S obzirom na to da se vrlo često događa da nije lako iz ACF i PACF grafa očitati vrijednosti p i q, onda za različite vrijednosti od p i q određujemo vrijednosti metrika koje nam govore koji je model bolji. Neke od metrika su AIC (engl. *Akaike's Information Criteria*) i BIC (engl. *Bayesian Information Criterion*) kriteriji koji su definirani na slijedeći način: neka je k broj parametara, n broj podataka i neka L maksimizira funkciju vjerodostojnosti, tada je

$$AIC = -2ln(L) + 2k \tag{2.20}$$

$$BIC = -2ln(L) + kln(n). \tag{2.21}$$

Odabrat ćemo onaj model koji ima najnižu vrijednost AIC, odnosno BIC kriterija.

2.3 ARIMA(p, d, q) proces

O procesu

ARMA procesi su važni za reprezentaciju stacionarnih vremenskih nizova, dok su ARIMA procesi veća klasa koja omogućuje reprezentaciju nestacionarnih vremenskih nizova. Modeliranje ARIMA procesa se sastoji od toga da se konačno mnogo puta diferencira niz i tako dani niz reduciraju do stacionarnog niza koji se može modelirati ARMA procesom.

Definicija 2.3.1. *Neka je d nenegativni cijeli broj. Za vremenski niz $\{X_t\}$ kažemo da je* ***ARIMA(p, d, q) proces*** *ako je $Y_t := (1-B)^d X_t$ kauzalan ARMA(p, q) proces.*

Definicija znači da vremenski niz $\{X_t\}$ zadovoljava sljedeću jednadžbu

$$\phi^*(B)X_t \equiv \phi(B)(1-B)^d X_t = \theta(B)Z_t, \qquad \{Z_t\} \sim WN(0, \sigma^2), \tag{2.22}$$

gdje su $\phi(z)$ i $\theta(z)$ polinomi reda p odnosno q i $\phi(z) \neq 0$, za $|z| \leq 1$. Polinom ϕ^* ima nultočku u $z = 1$ reda d. Proces $\{X_t\}$ je stacionaran ako i samo ako je $d = 0$ i u tom slučaju radi se o ARMA(p, q) procesu. Uočimo da ako u gornju jednadžbu X_t dodamo polinom stupnja $(d-1)$ da je jednakost i dalje zadovoljena. Upravo iz tog razloga ARIMA procesi su dobri za reprezentaciju podataka s trendom.

Primjena

U nastavku ću opisati kako su u dva rada koristili ARIMA proces za predviđanje protoka vozila, kakve podatke su imali te koji su njihovi zaključci.

Prvi primjer

U prvom radu ([2]) su se usredotočili na korištenje modela vremenskih nizova za predviđanje vremena putovanja na brzim pristupnim cestama. Od 26.10.2004. do 24.6.2005.godine na dijelu državne magistrale Minnesota 194 su skupljali podatke tijekom radnih dana od 15:30 do 17:00 sati. Za prikupljanje podatka su koristili vozila koja su sposobna za primanje GPS signala s informacijama o poziciji i vremenu. Bežičnom podatkovnom mrežom podaci se prenose do web poslužitelj. Ti podaci uključuju ID testnog vozila, geografsku duljinu i širinu, brzinu, smjer, vremensku oznaku, datum itd., a podaci o vremenskim oznakama koriste se za izračun vremena putovanja dionice. Duž promatranog koridora je ukupno 10 signaliziranih raskrižja.

U ovom radu su napravili model samo za segment ceste broj 7, dok je model za cijeli promatrani koridor moguće pronaći u [9]. Prvi korak u izradi modela im je bio da podacima o vremenu putovanja oduzmu prosječno vrijeme putovanja kako bi dobili podatke koji će imati očekivanje jednako nuli. Nakon što su podatke prikazali na grafu uočili su da taj proces nije stacionar. Podatke su jednom diferencirali i te podatke su nastavili koristiti. Sljedeći korak je bio prikazati ACF i PACF graf i zaključili su da će ARMA(p, q) model biti prikladan. Da bi odredili parametre p, q ARMA modela koristili su Yule-Walker metodu, Levison-Durbin algoritam, Burgov algoritam, Innovations algoritma i Hannan-Rissanen proceduru. Više o navedenim algoritmima potražite u [4] i [24]. Na promatranom segmentu dobili su da je prigodan ARIMA(6,1,3) model pri čemu su $\phi = [-0.339246, -0.778076, -0.116948, -0.0513862, 0.04, -0.158566]$ i $\theta = [-0.550511, 0.425886, -0.6653231]$. Kako bi provjerili da je model dobar napravili su analizu reziduala te *lack-of-fit* test. Obje provjere su potvrdile da je model prigodan.

Drugi primjer

U drugom radu ([14]) su koristili podskup ARIMA modela za kratkoročno predviđanje količine prometa na autocestama.

Podatke o količini prometa na autocestama u San Antoniu (Texsas) tijekom radnih dana su dobili iz TransGuide sistema. Sistem je napravljen tako da promatra uvjete u prometu, kontrolira prometnu signalizaciju te reagira na prometne nesreće. Broj vozila u intervalima od 5 minuta su prikupljali po 15 dana u periodu od 7 sati. Nasumično su odabrana dva područja na kojima su prikupili podatke. Prvo područje je imalo jedan detektor i on je skupljao podatke o prometu koji ide na zapad. Na tom dijelu su podatke prikupljali od veljače do ožujka 1996. godine između 7:00 i 14:00 sati. Na drugom području su četiri trake u jednom smjeru i tamo je detektor prikupljao podatke o količini prometa koja ide na istok. Podaci su prikupljani od lipnja do srpnja 1996. godine između 12:00 i 19:00 sati.

Podskup ARIMA modela reda (P, Q) je definiran kao reprezentacija sa samo nekoliko koeficijenata različitih od nule. To je proširena verzija ARIMA modela i dana je formulom

$$\sum_{k=0}^{P} \phi_k X_{t-k} = \sum_{m=0}^{Q} \theta_m Z_{t-m},$$

gdje su P i Q stupnjevi autoregresivnog procesa, odnosno procesa pomičnih prosjeka. $\{Z_t\} \sim WN(0, \sigma^2)$, $\phi = (\phi_1, \ldots, \phi_P)$ i $\theta = (\theta_1, \ldots, \theta_Q)$ su koeficijenti pri čemu su neki od njih jednaki nula. Za deskriptivni opis podataka su koristili ACF i PACF graf te standardizirana rezidualna odstupanja. Prilikom analize podataka uočili su da je količina prometa, na prvom području, za vrijeme jutarnjih gužvi dvostruko veća nego tijekom ostalog dijela promatranoga perioda te da dnevne fluktuacije nisu značajne. ACF i PACF graf su ukazali na jaku i linearnu korelaciju između susjednih podataka. Standardizirana rezidualna odstupanja (engl. *standardized residual variance (SRV)* su blizu 0.1, a pravilo je da ako je SVR manji od $8/n$ za neke indekse vremenskog zaostatka (engl. *lag*) ([20]) da se onda radi o podacima u kojima postoji ovisnost s mnogo prijašnjih podataka (podacima daleko u prošlosti). Zbog toga su diferencirali podatke, nakon čega je SVR bio 0.8 za indeks vremenskog zaostatka 40, što je veće od $8/n \approx 0.016$, pa u tim podacima nije bilo ovisnosti o podacima daleko u prošlosti.

Pri odabiru modela koristili su AIC kriterij, koji je definiran s $AIC(r) = n \log \hat{\sigma}_r^2 + 2r$ gdje su n i r broj primjera, odnosno broj parametara, a $\hat{\sigma}_r^2$ procjena varijance greške dobivene pomoću metode maksimalne vjerodostojnosti. Prvi dio AIC kriterija mjeri koliko dobro model opisuje podatke, dok je drugi dio kazna za prevelik broj parametara. Za oba područja su na osnovu AIC kriterija odabrali 4 modela: AR model, podskup AR modela, ARIMA model i podskup ARIMA modela, pri čemu su zadali da je maksimalan stupanj autoregresivnog procesa 30.

Kako bi provjerili adekvatnosti modela koristili su dva testa za provjeru jesu li reziduali bijeli šum. *Portmanteau* test je napravljen s $m = 30$ te je napravljen Bartlettov test i oba

testa su potvrdila adekvatnost modela. Za ocjenu greške predviđanja su korišteni prosječna apsolutna pogreška (engl. *mean absolute error (MAE)*) i korijen prosječne kvadratne pogreške (engl. *root mean squared error (RMSE)*). Zbog usporedbe korišteni su i rezultati predviđanja dobivenih pomoću metode eksponencijalnog zaglađivanja (engl. *exponential smoothing method (ESM)*). Pokazalo se da na oba promatrana područja, po obje mjere pogreške, podskup ARIMA modela ima najmanju grešku, dok je u prosjeku ESM bio bolji od ostalih modela. Osim toga kako bi uvidjeli uzorke u pogreškama prilikom predviđanja napravili su i analizu distribucije apsolutne postotne pogreške. Za svaki primjer je izračunata apsolutna postotna pogreška i ako je manja od 15% primjer je dodan u jednu kategoriju, inače u drugu. Pokazalo se da podskup ARIMA modela ima najviše primjera, 91.3% na prvom, odnosno 92.7% na drugom području, čija je apsolutna postotna pogreška manja od 15%.

2.4 SARIMA$(p, d, q)\times(P, D, Q)_s$ proces

O procesu

Definicija 2.4.1. *Neka su d i D nenegativni cijeli brojevi. Tada je* $\{X_t\}$ ***sezonalni ARIMA(*** $p, d, q)\times(P, D, Q)_s$ ***proces s periodom*** *s ako je diferencirani niz* $Y_t = (1 - B)^d(1 - B^s)^D X_t$ *kauzalni ARMA proces definiran s*

$$\phi(B)\Phi(B^s)Y_t = \theta(B)\Theta(B^s)Z_t, \qquad \{Z_t\} \sim WN(0, \sigma^2), \tag{2.23}$$

gdje su $\phi(z) = 1 - \phi_1 z - \cdots - \phi_p z^p, \Phi(z) = 1 - \Phi_1 z - \cdots - \Phi_P z^P, \theta(z) = 1 + \theta_1 z + \cdots + \theta_q z^q$ *i* $\Theta(z) = 1 + \Theta_1 z + \cdots + \Theta_Q z^Q$.

Uočimo da je proces $\{Y_t\}$ kauzalan ako i samo ako su $\phi(z) \neq 0$ i $\theta(z) \neq 0$ za $|z| \leq 1$. U primjenama je D najčešće jedan, dok su P i Q obično manju od tri.

Klasična dekompozicija $X_t = m_t + s_t + Y_t$ pretpostavlja da sezonalna komponenta s_t ima identična ponavljanja u svakom ciklusu, dok to nije slučaj kod sezonalnog ARIMA procesa. On dozvoljava da postoje neslučajnosti u sezonalnim uzorcima od jednog do drugog ciklusa.

Prvi korak pri identifikaciji SARIMA modela (izuzev transformacije podataka, ako je ona potrebna) je odrediti parametar d kako bi uklonili jedinični korijen iz nesezonalog dijela. Zatim iz ACF grafa određujemo s i nakon toga D koji je uobičajeno jednak jedan. Nakon toga dobijemo niz $Y_t = (1 - B)^d(1 - B^s)^D X_t$ koji nema jedinični korijen ni u nesezonalnom niti u sezonalnom dijelu. Zatim pomoću ACF i PACF grafa odredimo parametre P i Q sezonalnog dijela.

Primjena

U nastavku ću opisati kako su u dva rada koristili SARIMA proces za predviđanje protoka vozila, kakve podatke su imali te koji su njihovi zaključci.

Prvi primjer

U prvom radu ([28]) pokazuju kako je SARIMA model pogodan za kratkoročna predviđanja uvjeta na cestama na fiksnoj lokaciji u mreži prometnica.

Navode kako se teoretsko opravdanje za korištenje SARIMA modela za predviđanje protoka vozila nalazi u Woldovoj dekompoziciji koja se odnosi na podatke u diskretnom vremenu koji su stacionarni oko svog očekivanja i varijance. S obzirom na to potrebno je potkrijepiti tvrdnju da će sezonalno diferenciranje dovesti do stacionarnog niza. U praksi, Woldova dekompozicija nam govori da se proizvoljan stacionarni vremenski niz može prikazati kao suma determinističkog i stohastičkog niza. Štoviše, govori nam da se deterministički dio može prikazati kao linearna kombinacija prošlih vrijednosti, dok se stohastički dio može prikazati kao proces pomičnih prosjeka pomoću vremenski neovisnoga linearnog filtera $\Psi = \{\psi_0, \psi_1, \dots\}$. S obzirom na to, ako možemo pretpostaviti da su podaci o protoku prometa nakon diferenciranja stacionarni, onda je opravdano koristiti sezonalni ARIMA proces.

U Woldovoj dekompoziciji se zahtjeva da je niz stacionara, ali dovoljno je da je on skoro stacionaran. Naime, ako su vremenske ovisnosti u očekivanim vrijednostima i kovarijancama male relativno na nominalnu razinu niza, niz može biti dovoljno blizu stacionarnog da ga se efektivno može modelirati pomoću ARIMA modela. S obzirom na to, početne podatke želimo diferencirati do stacionarnog ili skoro stacionarnog niza.

U svom primjeru su koristili podatke s dvije autoceste. Jednu u Ujedinjenom Kraljevstvu i podaci su iz vanjske petlje na jugozapadnom kvadrantu autoceste oko Londona. Promatrana cesta ima četiri trake, a podatke o protoku su gledali na intervalu od 15 minuta. Druga autocesta je u Sjedinjenim Američkim Državama i podaci su iz sjeverozapadnog kvadranta autoceste oko Atlante. Podaci o protoku su također u intervalima od 15 minuta. Grafički prikaz podataka govori da će autokorelacija biti jako velika između svakog dana i svakog tjedna. Nakon što su napravili jednotjedno diferenciranje podataka iz ACF grafa za podatke s prve autoceste zaključuju da je niz stacionaran, dok je kod druge autoceste to nešto manje jasnije. Naime, dio podataka za tu autocestu je prikupljeno za vrijeme blagdana i praznika što je utjecalo na koreliranost podataka. Jedan tjedan ima 672 intervala i s obzirom na to $s = 672$. Williams je ([27]) pokazao da je ARIMA$(1, 0, 1) \times (0, 1, 1)_s$ konstantno najbolji model s obzirom na Schwarz-Bayesianov informacijski kriterij (SBC). Zbog toga su krenuli od tog modela i promotrili su modele gdje je p poprimao vrijednosti $1, 2, 3$ i za q s vrijednostima $1, 2$. Prema SBC kriteriju dobili su da je najbolji model ARIMA$(1, 0, 1) \times (0, 1, 1)_{672}$ i za prvu i za drugu autocestu. Kako bi mogli ocijeniti ko-

liko je model dobar prilikom previđanja usporedili su ih sa slučajnom šetnjom i povijesnim prosjekom. Greške su mjerili pomoću tri statistike: korijen prosječne kvadratne greške predikcije (RMSEP), prosječne apsolutne devijacije (MAD), prosječne apsolutne postotne greška (MAPE). Po svim kriterijima, na obje ceste, najmanju grešku pri predviđanju ima SARIMA model. Također, dani model ujedno ima i najmanju standardnu devijaciju pogreške koja je dana da se vidi kolika je pouzdanost intervala predviđanja.

Drugi primjer

U drugom radu ([5]) je primarni cilj odrediti SARIMA model, ali parametri nisu odredili pomoću klasičnih metoda (procjenitelj maksimalne vjerodostojnosti, metodom najmanjih kvadrata) nego pomoću Bayesove metode.

U Bayesovom pristupu sve nepoznate veličine se smatraju slučajnim varijablama i sve nesigurnosti vezane za te veličine su reprezentirane pomoću uvjetne vjerojatnosne distribucije na dostupnim podacima. Za procjenu distribucije parametara modela su koristili MCMC simulaciju (engl. *Markov Chain Monte Carlo simulation*). Više o Bayesovoj analizi možete pronaći u [13], a o MCMC simulacijama u [7]. Svaki procijenjeni parametar u Bayesovom pristupu ima svoju uvjetnu funkciju gustoće s obzirom na dane podatke. Slično, svako predviđanje ima krivulju s funkcijom gustoće, pri čemu točka s najvećom vjerojatnosti odgovara predviđanju.

Podaci o protoku prometa koji se korist su prikupljeni putem induktivnog *loop* detektora na jednom raskrižju u centru Dublina, koje se sastoji od jedne jednosmjerne ulice i dvije ulice koje se spajaju s njom. Podaci su prikupljani od 2. do 30. studenog 2004. godine. Duljina intervala u kojem su promatrali protok vozila je 15 minuta. Odlučili su napraviti model čiji će period biti jedan dan, zbog toga je $s = 96$. S obzirom na to da se promet vikendom značajno razlikuje od prometa tijekom radnih dana uklonili su podatke koje su prikupili tijekom vikenda. Predviđali su protok prometa između 6:30 i 00:00 tijekom 30. studenog 2004. godine. Apsolutna postotna pogreška model s parametrima aproksimiranim pomoću Bayesove metode je 5.4%, dok je s klasičnim odabirom parametara ona 5.1%. Uočavaju da se pomoću Bayesove metode bolje predviđa jutarnja gužva te da taj model ima veći raspon vrijednosti u usporedbi s klasičnim modelom i da taj model bolje odgovara brzoj varijabilnosti u stvarnim podacima.

2.5 STARIMA(p, d, q) proces

O procesu

U nastavku ću krenut sa STARMA modelom za stacionarni niz, a na kraju ću dodati kako izgleda STARIMA model.

Pretpostavimo da na N fiksnih lokacija u prostoru mjerimo slučajnu varijablu Z_t u T vremenskih perioda. STARMA opservaciju $z_i(t)$ na lokaciji i u trenutku t prikazuje kao linearnu kombinaciju prethodnih opservacija na lokaciji i i u okolnim lokacijama. Uvodimo operator prostornog zaostatka L, prostornog reda l tako da je

$$L^{(0)} z_i(t) = z_i(t) \tag{2.24}$$

$$L^{(l)} z_i(t)) = \sum_{j=1}^{N} w_{ij}^{(l)} z_j(t), \tag{2.25}$$

gdje su $w_{ij}^{(l)}$ težine takve da je $\sum_{j=1}^{N} w_{ij}^{(l)} = 1$, za sve i i različite su od nula ako su i i j susjedi reda l. Matrično to možemo prikazati kao

$$L^{(0)} \boldsymbol{z}(t) = W^{(0)} \boldsymbol{z}(t) = I_N \boldsymbol{z}(t), \tag{2.26}$$

$$L^{(l)} \boldsymbol{z}(t) = W^{(l)} \boldsymbol{z}(t), \quad l > 0, \tag{2.27}$$

gdje je $\boldsymbol{z}(t)$ vektor opservacija $z_i(t)$ i I_N je $N \times N$ jedinična matrica. Matricu težina određuje osoba koja gradi model i ona ovisi o prirodi problema. Težine moraju odražavati hijerarhijski poredak prostornih susjeda, što znači da će susjedi prvog reda imati veće težine od susjeda nekog veće reda. STARMA model u vektorskoj formi možemo prikazati kao

$$\boldsymbol{z}_t = \sum_{k=1}^{p} \sum_{l=0}^{\lambda_k} \phi_{kl} W^{(l)} \boldsymbol{z}(t-k) - \sum_{k=1}^{q} \sum_{l=0}^{m_k} \theta_{kl} W^{(l)} \boldsymbol{\varepsilon}(t-k) + \boldsymbol{\varepsilon}(t), \tag{2.28}$$

gdje je $\boldsymbol{\varepsilon}(t)$ normalno distribuiran s očekivanjem nula i

$$E[\boldsymbol{\varepsilon}(t)\boldsymbol{\varepsilon}(t+s)'] = \begin{cases} \sigma^2 I_N & s = 0, \\ 0 & \text{inače.} \end{cases} \tag{2.29}$$

Za identifikaciju modela trebaju nam autokovarijacijska i autokrelacijska funkcija koje su u vektorskom obliku definirane u nastavku. *Funkcija vremensko-prostorne kovarijance* je definirana s

$$\gamma_{lk}(s) = E\left\{ \frac{[\boldsymbol{W}^{(l)} \boldsymbol{z}(t)]'[\boldsymbol{W}^{(k)} \boldsymbol{z}(t+s)]}{N} \right\} \tag{2.30}$$

Ako je $\boldsymbol{\Gamma}(s) = E[\boldsymbol{z}(t)\boldsymbol{z}(t+s)']$, tada je $\gamma_{lk} = \frac{1}{N} tr(\boldsymbol{W}^{(k)} \boldsymbol{W}^{(l)} \boldsymbol{\Gamma}(s))$. Uzoračku autokovarijacijsku funkciju $\hat{\gamma}_{lk}$ dobijemo tako da umjesto $\boldsymbol{\Gamma}(s)$ uvrstimo

$$\hat{\boldsymbol{\Gamma}}(s) = \sum_{t=1}^{T-s} \frac{\boldsymbol{z}(t)\boldsymbol{z}(t+s)'}{T-s}. \tag{2.31}$$

Vremenskto-prostorna autokorelacija između susjeda l-tog i k-tog reda koji su odvojeni za s je

$$\rho_{lk}(s) = \frac{\gamma_{lk}(s)}{[\gamma_{ll}(0)\gamma_{kk}(0)]^{1/2}}. \tag{2.32}$$

Za uzoračku vremensko-prostornu autokorelacijsku funkciju samo u prethodnoj definiciji γ_{lk} zamijenimo s $\hat{\gamma}_{lk}$. Nakon što smo izračunali $\hat{\rho}_{lk}$ treba odrediti kandidate za model. Parametar λ, prostorni stupanj u vremensko-prostornoj parcijalnoj autokorelacijskoj funkciji, je ostavljen na odabir osobi koja izrađuje model. Pri odabiru treba voditi računa da λ treba biti barem kao maksimalni prostorni stupanj u bilo kojem modelu koji se provjerava. S druge strane, ako uzmemo preveliku vrijednost to može biti skupo za računanje. Ako parcijalna i autokorelacijska funkcija obje imaju eksponencijalni pad prema nuli i oboje u vremenu i prostoru, tada možemo reći da se radi o STARMA modelu. Parametri za autoregresivni proces i proces pomičnih zareza u vremensko-prostornom ARMA procesu se određuju analogno kao za ARMA proces. Parametri $\boldsymbol{\phi}$ i $\boldsymbol{\theta}$ se određuju metodom maksimalne uvjetovane vjerodostojnosti. Funkcija uvjetovane vjerodostojnosti je definirana s

$$L(\boldsymbol{\phi}, \boldsymbol{\theta}, \sigma^2|\boldsymbol{z}) = \frac{1}{(2\pi\sigma^2)^{-TN/2}} exp\left(- \frac{S_*(\boldsymbol{\phi}, \boldsymbol{\theta})}{2\sigma^2} \right), \tag{2.33}$$

gdje je

$$S_*(\boldsymbol{\phi}, \boldsymbol{\theta}) = \hat{\varepsilon}\hat{\varepsilon}', \tag{2.34}$$

$$\hat{\varepsilon}(t) = \boldsymbol{z}(t) - \sum_{k=1}^{p} \sum_{l=0}^{\lambda_k} \phi_{kl} W^{(l)} \boldsymbol{z}(t-k) + \sum_{k=1}^{q} \sum_{l=0}^{m_k} \theta_{kl} W^{(l)} \boldsymbol{\varepsilon}(t-k), \tag{2.35}$$

za $t = 1, 2, \ldots, T$, dok su za $t < 1$ $\hat{\varepsilon}(t)$ i $\boldsymbol{z}(t)$ jednaki nuli. Može se pokazati da je procjenitelj maksimalne uvjetovane vjerodostojnosti upravo

$$\hat{\sigma}^2 = \frac{S_*(\hat{\boldsymbol{\phi}}, \hat{\boldsymbol{\theta}})}{TN}, \tag{2.36}$$

gdje su $\hat{\boldsymbol{\phi}}, \hat{\boldsymbol{\theta}}$ takvi da minimiziraju $S_*(\hat{\boldsymbol{\phi}}, \hat{\boldsymbol{\theta}})$.

Ako je $q \neq 0$, onda je STARMA model nelinearan, pa se parametri mogu procijeniti s više nelinearnih optimizacijskih algoritama. Opširniji opis tih metoda možete pronaći u [25]. Nakon što se odabere model slijedi provjera je li taj model dobar za dane podatke. Postoje dva načina na koja model može biti loš. Prvi je da model ne predstavlja dovoljno dobro koreliranost procesa i takva pogreška se može uočiti ako su korelacije između reziduala velike. Drugi problem je da je model neopravdano kompleksan što je vidljivo ako postoje parametri koji nisu statistički značajni. Prva faza provjere modela je provjera jesu li reziduali bijeli šum i to možemo napraviti pomoću vremensko-prostorne autokorelacijske i parcijalne autokorelacijske funkcije. Ako reziduali nisu bijeli šum, oni možda imaju neki

uzorak ponašanja koji može biti reprezentiran pomoću STARMA modela, i on se može ukomponirati s početnim modelom kako bi u konačnici dobili bolji model i reziduale koji su bijeli šum. Idući korak je provjera statističke značajnosti koeficijenata modela. U slučaju da neki od koeficijenata nisu statistički značajni oni se trebaju ukloniti i jednostavniji model se sada treba uzeti kao kandidat za model.

Kod STARMA model početni niz koji promatramo je stacionaran, dok kod STARIMA model niz prvo moramo difrencirati kako bi dobili stacionaran niz. S obzirom na to, dodatni korak kod STARIMA modela je da se početni niz diferencirati dok ne dobijemo stacionaran niz.

Više detalja o STARIMA procesu možete pronaći u [25].

Primjena

U ovom radu ([10]) predstavljaju kako modelirati protok prometa pomoću STARIMA modela.

Protok u mreži prometnice prikazuju kao stablo, pri čemu usmjereni vektori odgovaraju smjeru prometa, a čvorovi su lokacije na kojima se mjeri protok vozila. Neki čvorovi nisu povezani i stoga se ponašaju nezavisno jedan o drugome. Primjer takve situacije je kada ne postoji način da se iz jednog čvora dođe do drugog, pa zbog toga promet u prvom čvoru ne utječe na promet u drugom čvoru. Također, ako gledamo jednosmjerne ceste jasno je da čvor koji se nalazi niže neće utjecati na čvor koji je prije njega. Neka je N broj čvorova, odnosno broj lokacija na kojima se mjeri protok vozila. Tada je W_l kvadratna $N \times N$ matrica težina čiji je element $w_{i,j}^{(l)}$ različit od nule ako su čvorovi i i j susjedi l-tog reda. Težine su određene tako da susjedi l-tog reda jednog čvora imaju jednake težine i prema tome suma svakog retka matrice W_l jednaka jedan.

Podaci su prikupljeni sa 34 detektora na glavnim cestama koje vode do središta Atene. Detektori svakih 90 sekundi zapisuju količinu prometa, a oni su koristili prosjek pet uzastopnih intervala. Odnosno, gledali su intervale od 7.5 minuta i za svaki dan imaju 192 zapisa. Podatke prikupljene tijekom vikenda su izbacili, jer se protok vozila tada značajno razlikuje od prometa tijekom radnih dana. Nakon analize podataka odlučili su koristiti samo 25 detektora, jer je preostalih 9 imalo sumnjiva mjerenja. U svom modelu su promatrali relativnu brzinu koju su definirali kao količinu prometa podijeljenu s vremenom u kojem je dani dio ceste bio zauzet vozilima. Kako bi provjerili stabilnost određivanja parametra, odvojeni modeli su prilagođeni za dva vremenska perioda. Prvi dio je između srpnja i kolovoza, a drugi od veljače do ožujka. S obzirom na to da većina ljudi u Ateni ima godišnji tijekom kolovoza potvrđene su sumnje da će relativne brzine u tom periodu biti veće. Uklonili su dnevnu sezonalnost te su od podataka oduzeli očekivanje, kako bi dobili podatke čije će očekivanje biti nula. Kako bi provjerili postoji li jedinični korijen proveli su ADF test (engl. *augmented Dickey Fuller*). U slučaju da niz ima jedinični ko-

rijen to znači da on nije stacionaran. Rezultati testa su im rekli da podaci ne odstupaju od stacionarnosti. Za oba perioda vremensko-prostorna autokorelacije se smanjuje i pokazuje značajan porast u prostornom indeksu 0 i 192 indeksu vremenskog zaostatka , što ukazuje da parametar za pomične prosjeke u STARIMA modelu treba biti 192. Parcijalna autokorelacije nestaje u trećem indeksu vremenskog zaostatka za nulti indeksu prostornog zaostatka i na prvom indeksu vremenskog zaostatka za drugi indeks prostornog zaostatka. S obzirom na to kandidat za modele za oba perioda je dan s

$$Z_t = \phi_{10}Z_{t-1} + \phi_{20}Z_{t-2} + \phi_{30}Z_{t-3} + \phi_{11}W_1Z_{t-1} + \phi_{12}W_2Z_{t-1} - \theta_{10}a_{t-192} + a_t.$$

Znači da je svako mjerenje u vremenu t prikazano kao linearna kombinacija tri prethodna mjerenja plus težinski prosjek mjerenja od susjeda prvog reda u vremenu $(t-1)$ te težinski prosjek mjera susjeda drugog reda u vremenu $(t-1)$ plus predviđena pogreška koja je napravljena u isto vrijeme dan prije i plus slučajna greška. Nakon što su odredili parametre za predloženi model provjerili su koliko je taj model dobar. To između ostaloga uključuje i računanje vremensko-prostorne autokorelacije reziduala koji su pokazali dobre rezultate za sve osim za indeks vremenskog zaostatka 192 i indeks prostornog zaostatka nula za oba perioda i s obzirom na to ponovno su definirali modele kao

$$Z_t = \phi_{10}Z_{t-1} + \phi_{20}Z_{t-2} + \phi_{30}Z_{t-3} + \phi_{11}W_1Z_{t-1} + \phi_{12}W_2Z_{t-1} - \theta_{20}a_{t-192} - \theta_{10}a_{t-1} + a_t,$$

te su odredili vrijednost parametara. Pokazalo se da parametri koji odgovaraju opadajućim vremenskim stupnjevima također opadaju te da su oni manje statistički značajni. Parametri koji odgovaraju susjedima drugog reda su se pokazali značajnim od onih koji odgovaraju susjedima prvog reda. Kao uzrok tome navode da su promatrani vremenski intervali između opservacija relativno dugi. Vremensko-prostorna autokorelacijska i parcijalna autokorelacijska funkcija za nove reziduale pokazuju da je ovaj model prikladan.

Za razliku od prethodnih modela i pristupa, pomoću STARIMA procesa se može napraviti jedan model koji će pri predviđanju u obzir uzimati i vremensko-prostorne odnose detektora te se može koristiti ne samo za jednu cestu, nego za cijelu mrežu prometnica.

Poglavlje 3

Duboko učenje

Prvi računalni programi temeljili su svoja predviđanja na osnovu pravila zadanih od strane čovjeka, no razvojem računala, te mogućnosti skupljanja velikih količina povijesnih podataka razvile su se i metode za automatsko stvaranje modela ili baza znanja izvlačenjem informacija iz dostupnih podataka, prije svega metodama strojnog učenja. Uspjeh jednostavnih modela strojnog učenja uvelike ovisi o reprezentaciji danih podataka. Svaka informacija uključena u reprezentaciju se naziva *atribut* (engl. *feature*). Mnogi problemi se mogu riješiti pomoću jednostavnih modela strojnog učenja, uz uvjet da imamo atribute koji dobro opisuju problem. S druge strane, za mnoge probleme je teško odrediti koji atributi se trebaju koristiti. Jedno rješenje je da se metode koriste ne samo za izračunavanje izlazne vrijednosti, nego i za otkrivanje pogodne reprezentacije i tu dolazimo do dubokog učenja. Duboko učenje je posebna vrsta strojnog učenja koje postiže veliku moć i fleksibilnost učeći reprezentacije problema kao ugniježđenu hijerarhiju koncepata, pri čemu je svaki koncept u relaciji s jednostavnijim konceptima, t.j. apstraktnije reprezentacije izvode se kombiniranjem manje apstraktnih.

Za rješavanje problema promatranom u ovom radu se osim modela vremenskih nizova, koriste i metode dubokog učenja. Poseban naglasak će biti na *rekurentnim neuronskim mrežama*, a kao uvod u njih prvo dajem kratak opis neuronskih i dubokih neuronskih mreža.

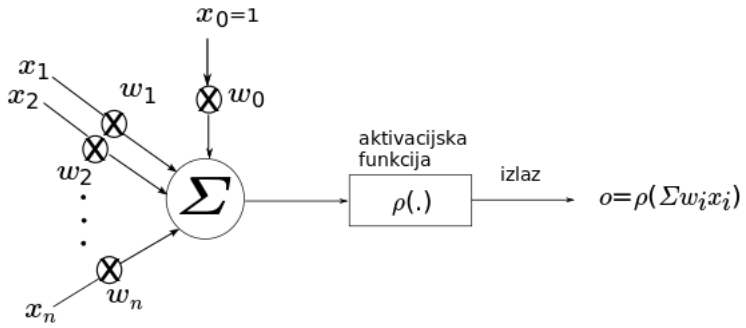
Detaljnija objašnjenja možete pronaći u [1] te u [6].

3.1 Neuronske mreže

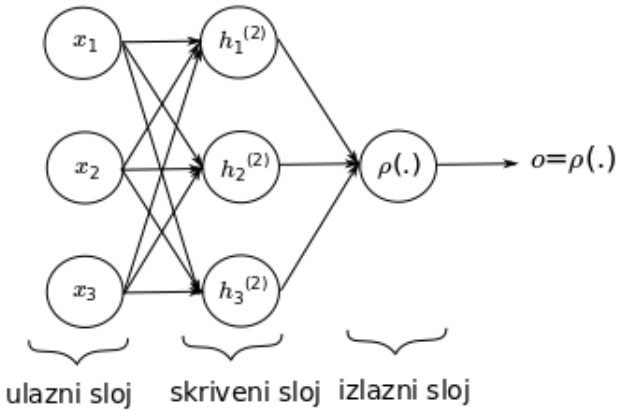
Osnovna ideja neuronskih mreža se može objasniti pomoću bioloških neurona. Vrlo pojednostavljeno objašnjeno je da biološki neuroni na jednom kraju imaju dendrite koji primaju signale od drugih neurona, ti signali se obrađuju u tijelu neurona i ovisno o jačini impulsa on se preko aksona prenosi dalje. Umjetni neuron baziran je na sličnim principima: ulazni podaci se procesiraju u tijelu neurona, zatim se sumiraju u aktivacijskoj funkciju

koja daje izlaz, odnosno povratnu informaciju (vidi sliku 3.1). Na slici 3.1 je prikazan jedan neuron, a neuronska mreža se sastoji od više takvih neurona, koji mogu biti u više slojeva (vidi sliku 3.2). Takve neuronske mreže su jednostavne, dok složene neuronske mreže imaju više čvorova i slojeva.

S $\{x_1, x_2, \ldots, x_n\}$ označavamo ulazni skup podataka, a s $\{w_1, w_2, \ldots, w_n\}$ pripadajuće težine. U nastavku ću za pojam *umjetni neuron* ravnopravno koristiti i pojmove *neuron*, *čvor* i *jedinica*. Najjednostavnija neuronska mreža se zove *perceptron* i ona za aktivacijsku funkciju ima *funkciju praga*, dok je izlaz jednak jedan ili nula, ovisno je li suma $\sum w_i x_i$ veća ili manja od nule.



Slika 3.1: Umjetni neuron



Slika 3.2: Neuronska mreža s jednim skrivenim slojem

Za aktivacijsku funkciju se najčešće uzimaju funkcije tangens hiperbolni (*tanh*), sigomoidalna funkcija (σ) i tzv. *rectified linear unit (ReLU)* koje su su definirane s:

$$tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} \tag{3.1}$$

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{3.2}$$

$$ReLU(z) = \begin{cases} z & z > 0 \\ 0 & z \leq 0. \end{cases} \tag{3.3}$$

Cilj je naučiti težine veza između čvorova mreže, $\{w_1, w_2, \ldots, w_n\}$, tako da je razlika između stvarnih vrijednosti i vrijednosti dobivenih pomoću neuronske mreže što je moguće manja. U tu svrhu potrebno je definirati i *funkciju gubitka L*, koja mjeri koliko su dobre vrijednosti dobivene pomoću neuronske mreže. Sama definicija funkcije gubitka L ovisi o promatranom problemu. Kako bi pronašli minimalnu vrijednost funkcije gubitka potrebno je iterativno ažurirati težine w_i, a za to koristimo gradijente funkcije L u odnosu na težine w_i. *Backpropagation* je algoritam koji se najčešće koristi za računanje gradijenata i on pri računanju koristi lančano pravilo za deriviranje. Iako nam *backpropagation* ne jamči da ćemo doći u globalni minimum, u praksi se pokazao kao izrazito dobar za korekcije težina po slojevima mreže. Detalji algoritma mogu se pronaći u poglavlju 6.5. u [6]. Gradijentni spust i stohastički gradijentni spust su samo neki od algoritama koji se koriste za ažuriranje težina. Razlikuju se u tome što se u gradijentnom spustu sve za određivanje gradijenta koristi greška izračunata na svim podacima za učenje, dok stohastički gradijentni spust korekciju težina bazira na greški izmjerenoj na podskupu primjera iz skupa za učenje, odnosno na pojedinim primjerima. Detalje o navedenim algoritmima možete pročitati u [6, pogl. 4.3, 5.9].

Ukratko, za učenje neuronske mreže imamo dva prolaza kroz mrežu. Prvi put kad prolazimo kroz mrežu koristimo težine koje imamo te računamo izlaznu funkciju. Nakon toga odredimo vrijednost ukupnog gubitka. Drugi prolaz se sastoji od *backpropagation* algoritma koji računa gradijente, koji se onda koriste za ažuriranje težina. Taj postupak ponavljamo više puta za cijeli skup podataka za treniranje, a pojam *epoha* se odnosi na jedan prolaz kroz cijeli skup za treniranje.

3.2 Duboke neuronske mreže

Duboke neuronske mreže za razliku od običnih neuronskih mreža imaju više skrivenih slojeva. Svaki sloj za ulaz dobije atribute izračunate u prethodnom sloju, pa s obzirom na to koristeći veći broj skrivenih slojeva mreža može naučiti kompleksnije funkcije originalnih atributa. Veće dostupne količine podataka, razvoj moćnijih računala, te razvoj metoda

za treniranje su doveli do porasta i popularnosti korištenja metoda dubokog učenja. U nastavku objašnjavam pojmove koji će biti potrebni u ostatku rada.

Dostupni skup podataka dijelimo na tri manja skupa. Prvi je skup za treniranje na kojemu učimo parametre (težine), drugi je skup za validaciju na kojemu se provjerava kako trenutna mreža radi i treći skup je skup za testiranje. Na njemu provjeravamo točnost i preciznost odabranoga modela. Skup za validaciju koristimo jer pri učenju može doći do *pretreniranja*. Pretreniranje se događa kada mreža jako dobro radi na skupu za treniranje, ali ne postiže dobre rezultate na skupu za validaciju. U tom slučaju kažemo da mreža loše generalizira. S druge strane može se dogoditi da mreža daje loše rezultate i na skupu za treniranje i na skupu za validaciju. To upućuje da model nije dovoljno kompleksan. Taj problem se rješava dužim treniranjem, kompleksnijom mrežom s više skrivenih neurona. Prilikom učenja pod *iteracijom* podrazumijevamo jedan prolaz u kojem se ažuriraju težine, odnosno parametri modela. Ovisno o broju primjera koji se koriste u jednoj iteraciji razlikujemo nekoliko vrsta učenja:

- *Online* učenje u kojem se u jednoj iteraciji promatra samo jedan primjer.
- Grupno učenje (*batch training*) podrazumijeva da se u jednoj iteraciji koriste svi primjeri iz skupa za učenje. U ovom slučaju epoha i iteracija se podudaraju.
- *Mini-batch* učenje podrazumijeva da se u jednoj iteraciji koristi manji podskup primjera iz skupa za učenje.

Kako bi se smanjila greška na skupu za testiranje koriste se različite *metode regularizacije*. Tako na primjer imamo L^1 i L^2 regularizaciju kojom se penalizira veličina težina. *Dropout* regularizacijom se u pojedinačnim iteracijama, slučajno odabrani čvorovi i sve veze s tim čvorovima zanemaruju prilikom korekcije težina. Prednosti, nedostatke i implementacijske detalje navedenih regularizacijskih metoda možete pronaći u [6].

Treniranje, odnosno učenje parametara neuronskih mreža je vremenski zahtjevno i zbog toga je od iznimne važnosti korištenje efikasnih *optimizacijskih algoritama* za pronalaženje parametara koji će minimizirati ukupnu funkciju gubitka. Neke od metoda koje se mogu koristiti su stohastički gradijentni spust, stohastički gradijentni spust s momentom, AdaGrad, RMSProp, Adam. Kako bi se postiglo brže učenje i veća robusnost parametara duboke neuronske mreže koristi se *batch* normalizacija. Metoda podrazumijeva da se za sve atribute u danom *mini-batch*-u izračunaju uzoračko očekivanje i varijanca te da se atributi normaliziraju. Detalje o navedenim metodama potražite u [6, pogl. 8].

Na kraju, pri svakoj iteraciji algoritam računa vrijednost funkcije gubitka, a algoritam staje s treniranjem kada je funkcija gubitka postigla malu vrijednost, ili kada se njena vrijednost neznatno mijenja kroz niz iteracija.

Postoji više tipova dubokih neuronskih mreža, ali u ovom radu su nam od posebne važnosti rekurentne neuronske mreže. One se koriste kada imamo podatke u obliku nizova,

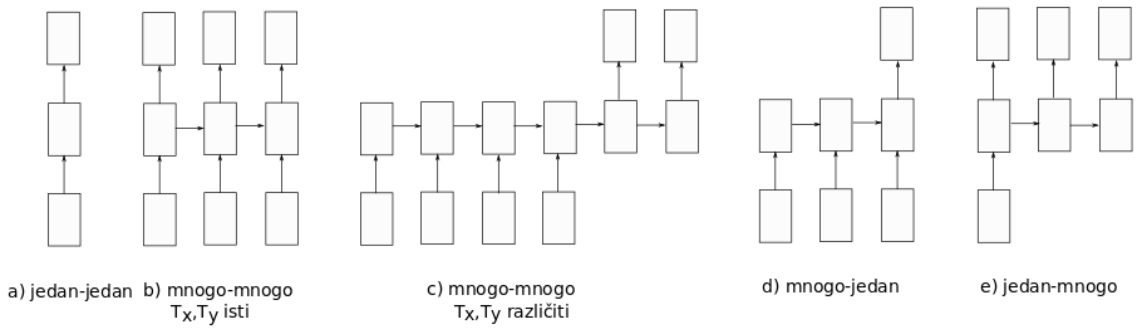
odnosno vremenskih serija (kod kojih postoji slijedna zavisnost, te koje nisu nužno iste duljine).

3.3 Rekurentne neuronske mreže

U ovom poglavlju izložit ćemo osnovne koncepte i funkcioniranje rekurentnih neuronskih mreža (engl. *Recurrent Neural Networks (RNN)*). Detaljniji prikaz se može pronaći u [6, pogl. 10] ili na web stranici [11].

Rekurentne neuronske mreže su familija neuronskih mreža koje služe za obrađivanje nizova podataka varijabilnih duljina. Kod rekurentnih neuronskih mreža težine se prenose u vremenu, odnosno iste su za sve dijelove niza. Dijeljenje težina se zasniva na pretpostavci da se iste težine mogu koristiti za različite pozicije u nizu. S obzirom na to, ulazni nizovi mogu biti različitih duljina i imamo manji broj parametara koji se trebaju naučiti. Rekurentne neuronske mreže dijele parametre tako da se na svaku izlaznu vrijednost primjeni isto pravilo ažuriranja kao i na prethodne vrijednosti.

Kao što je rečeno, rekurentne neuronske mreže omogućuju da ulazne i izlaze vrijednosti budu različitih duljina i zbog toga su primjenjive za različite probleme. Ulazni niz ćemo označavati s $(\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots, \boldsymbol{x}^{(T_x)})$, dok ćemo izlazne vrijednosti označavati s $(\boldsymbol{y}^{(1)}, \boldsymbol{y}^{(2)}, \ldots, \boldsymbol{y}^{(T_y)})$ pri čemu su T_x i T_y duljine nizova. Mogući oblici rekurentnih neuronskih mreža su prikazani na slici)3.3).

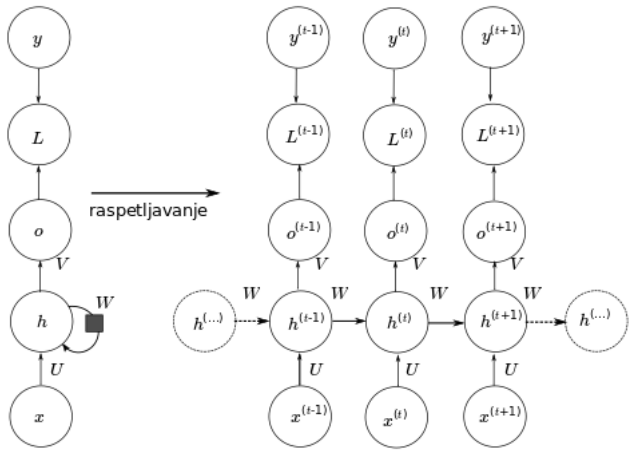


Slika 3.3: Oblici rekurentnih neuronskih mreža. Donji kvadrati označavaju ulazne vrijednosti, srednji su skriveni slojevi, a gornji su predviđanja. a) je zapravo obična neuronska mreža, u b) su ulazni i izlazni niz jednake duljine, u c) su ulazni i izlazni niz različitih duljina, u d) na ulazu imamo niz, dok je izlaz jedna vrijednost, a u e) je obrnuto.

Rekurentnu neuronsku mrežu možemo prikazati na dva načina. Pomoću jedne petlje ili pomoću raspretljane petlje u kojoj su vidljive kopije iste neuronske mreže uz dijeljenje

parametara (vidi slike 3.4 i 3.5). Nekoliko važnih oblika rekurentnih neuronskih mreža s obzirom na povratne povezanosti i izlazne vrijednosti su:

- RNN koje daju predviđanje u svakom vremenskom koraku i između skrivenih neurona je povratna povezanost. Vidi sliku (3.4).
- RNN koje daju predviđanje u svakom vremenskom koraku i imaju povratnu povezanost između izlazne vrijednosti skrivenoga neurona i trenutnog skrivenoga neurona. Vidi sliku (3.5).
- RNN s povratnom vezom između skrivenih neurona, koja prođe čitav niz i kao izlaz ima jednu vrijednost. Vidi sliku (3.6).



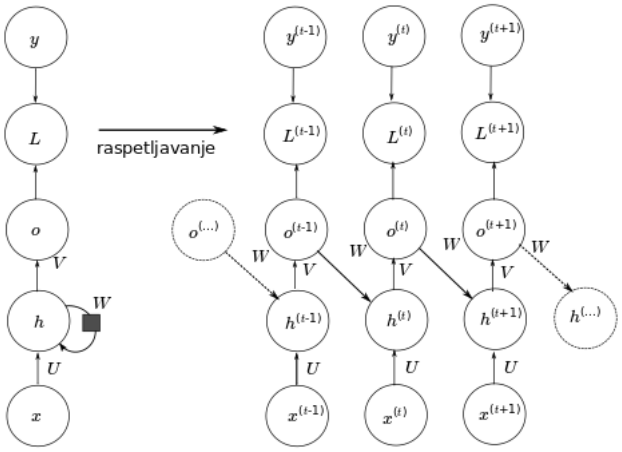
Slika 3.4: RNN koje daju predviđanje u svakom vremenskom koraku i između skrivenih neurona je povratna povezanost. Skriveni neuroni su označeni s $h^{(t)}$, izlazne vrijednosti skrivenoga neurona s $o^{(t)}$, funkcija gubitka u trenutku t s $L^{(t)}$, a predviđena vrijednost s $y^{(t)}$

Skriveni neuroni $\boldsymbol{h}^{(t)}$ kao ulaznu vrijednosti dobivaju stanje $\boldsymbol{x}^{(t)}$, ali i vrijednost skrivenih neurona u prethodnom trenutku $\boldsymbol{h}^{(t-1)}$, pa se vrijednost od $\boldsymbol{h}^{(t)}$ može odrediti iz jednadžbe

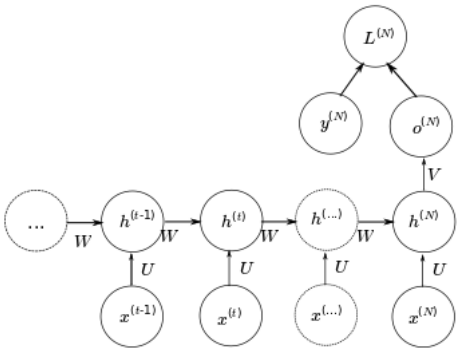
$$\boldsymbol{h}^{(t)} = g(\boldsymbol{W}_{hh}\boldsymbol{h}^{(t-1)} + W_{hx}\boldsymbol{x}^{(t)} + \boldsymbol{b}_h), \tag{3.4}$$

gdje je g aktivacijska funkcija, $\boldsymbol{W}_{hx}$ i $\boldsymbol{W}_{hh}$ su matrice težina između ulaznog i skrivenoga sloja, odnosno između dva susjedna skrivena sloja, a $\boldsymbol{b}_h$ je vektor pristranosti. Izlazna vrijednost, odnosno predviđanje se dobije kao

$$\boldsymbol{o}^{(t)} = \boldsymbol{W}_{yh}\boldsymbol{h}^{(t)} + \boldsymbol{b}_y \tag{3.5}$$



Slika 3.5: RNN koje daju predviđanje u svakom vremenskom koraku i imaju povratnu povezanost između izlazne vrijednosti skrivenoga neurona i trenutnog skrivenoga neurona.



Slika 3.6: RNN s povratnom vezom između skrivenih neurona, koja prođe čitav niz i kao izlaz ima jednu vrijednost.

$$\hat{\boldsymbol{y}}^{(t)} = f(\boldsymbol{o}^{(t)}), \tag{3.6}$$

gdje je $\boldsymbol{o}^{(t)}$ izlazna vrijednost, f je izlazna funkcija, $\boldsymbol{W}_{yh}$ matrica težina između skrivenoga i izlaznog sloja, a $\boldsymbol{b}_y$ je vektor pristranosti. Kao aktivacijska funkcija se najčešće koriste tangens hiperbolni ili *ReLu*, dok se za izlaznu funkciju najčešće koristi *softmax* funkcija.

Forward propagation počinje tako da $\boldsymbol{h}^{(0)}$ inicijalizira na nulu, a zatim za svaki vremenski korak t primjene gornje jednadžbe.

Za ulazni niz $\boldsymbol{x}$ i pripadni izlazni niz $\boldsymbol{y}$ duljine τ ukupan gubitak definiramo kao sumu

gubitaka po svim vremenskim koracima. Recimo da s $L^{(t)}$ označavamo gubitak kod predviđanja u trenutku t, tada je ukupni gubitak definiran kao

$$L(\{\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots, \boldsymbol{x}^{(\tau)}\}, \{\boldsymbol{y}^{(1)}, \boldsymbol{y}^{(2)}, \ldots, \boldsymbol{y}^{(\tau)}\}) = \sum_t L^{(t)}. \tag{3.7}$$

U primjenama se $L^{(t)}$, funkcija za gubitak u trenutku t, prilagođava problemu za koji se rekurentna neuronska mreža koristi.

Kako bi istrenirali mrežu treba nam *backpropagation* algoritam pomoću kojega ažuriramo težine, s ciljem minimiziranja funkcije ukupnog gubitka. Za rekurentne neuronske mreže koristimo modifikaciju standardnog *backpropagation* algoritma kojeg nazivamo *backpropagation through time (BPTT)*. Čvorovi u mreži uključuju parametre $L^{(t)}$, $\boldsymbol{h}^{(t)}$, $\boldsymbol{o}^{(t)}$, $\boldsymbol{W}_{hh}$, $\boldsymbol{W}_{hx}$, $\boldsymbol{W}_{yh}$, $\boldsymbol{b}_h$ i $\boldsymbol{b}_y$ i trebamo izračunati gradijent funkcije ukupnog gubitka s obzirom na navedene parametre. Pri računanju gradijenta se koristi lančano pravilo za derivacije, a u nastavku su izračunati navedeni gradijenti:

$$\nabla_{\boldsymbol{o}^{(t)}} L = \hat{\boldsymbol{y}}^{(t)} - 1 \tag{3.8}$$

$$\nabla_{\boldsymbol{h}^{(t)}} L = \boldsymbol{W}_{yh}^T \nabla_{\boldsymbol{o}^{(t)}} L = \boldsymbol{W}_{hh}^T (\nabla_{\boldsymbol{h}^{(t+1)}} L) \operatorname{diag}\left(1 - (\boldsymbol{h}^{(t+1)})^2\right) + \boldsymbol{W}_{yh}^T (\nabla_{\boldsymbol{o}^{(t)}} L) \tag{3.9}$$

$$\nabla_{\boldsymbol{b}_y} L = \sum_t \left(\frac{\partial \boldsymbol{o}^{(t)}}{\partial \boldsymbol{b}_y^{(t)}}\right)^T \nabla_{\boldsymbol{o}^{(t)}} L = \sum_t \nabla_{\boldsymbol{o}^{(t)}} L \tag{3.10}$$

$$\nabla_{\boldsymbol{b}_h} L = \sum_t \left(\frac{\partial \boldsymbol{h}^{(t)}}{\partial \boldsymbol{b}_h^{(t)}}\right)^T \nabla_{\boldsymbol{h}^{(t)}} L = \sum_t \operatorname{diag}\left(1 - (\boldsymbol{h}^{(t)})^2\right) \nabla_{\boldsymbol{h}^{(t)}} L \tag{3.11}$$

$$\nabla_{\boldsymbol{W}_{yh}} L = \sum_t \sum_i \frac{\partial L}{\partial \boldsymbol{o}_i^{(t)}} \nabla_{\boldsymbol{W}_{yh}} \boldsymbol{o}_i^{(t)} = \sum_t (\nabla_{\boldsymbol{o}^{(t)}} L) \boldsymbol{h}^{(t)T} \tag{3.12}$$

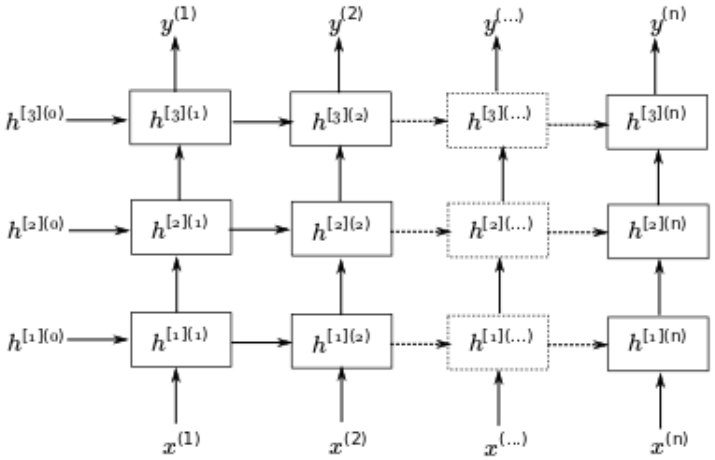
$$\nabla_{\boldsymbol{W}_{hh}} L = \sum_t \sum_i \frac{\partial L}{\partial \boldsymbol{h}_i^{(t)}} \nabla_{\boldsymbol{W}_{hh}} \boldsymbol{h}_i^{(t)} = \sum_t \operatorname{diag}\left(1 - (\boldsymbol{h}^{(t)})^2\right) (\nabla_{\boldsymbol{h}^{(t)}} L) \boldsymbol{h}^{(t-1)T} \tag{3.13}$$

$$\nabla_{\boldsymbol{W}_{hx}} L = \sum_t \sum_i \frac{\partial L}{\partial \boldsymbol{h}_i^{(t)}} \nabla_{\boldsymbol{W}_{hx}} \boldsymbol{h}_i^{(t)} = \sum_t \operatorname{diag}\left(1 - (\boldsymbol{h}^{(t)})^2\right) (\nabla_{\boldsymbol{h}^{(t)}} L) \boldsymbol{x}^{(t)T} \tag{3.14}$$

S obzirom na to da u svakom koraku izračun vrijednosti u trenutku t ovisi o vrijednosti u trenutku $(t-1)$ *BPTT* se ne može paralelizirati kako bi se ubrzalo vrijeme izvođenja. U slučaju da su ulazni i izlazni niz duljine τ vremenska složenost *BPTT* algoritma iznosi $O(\tau)$. Sva stanja koja su izračunata prilikom *forward propagation* su potrebna za *backpropagation* te se zbog toga moraju spremiti, zbog čega je prostorna složenosti također $O(\tau)$. Pomoću rekurentnih neuronskih mreža možemo prikazati veliki broj funkcija, ali s druge strane one su zahtjevne po pitanju vremenskog izvođenja i potrebne memorije.

U slučaju da u mreži ne postoji povratna veza između skrivenih slojeva pomoću metode *teacher forcing* se može izbjeći *BPTT*. U tom slučaju se treniranje može paralelizirati i učenje je brže. Više o spomenutoj metodi možete pronaći u [6, pogl. 10.2.1.].

Za učenje kompleksnih funkcija često je korisno spojiti nekoliko slojeva rekurentnih neuronskih mreža kako bi se dobili dublji modeli. Duboke rekurentne neuronske mreže možemo dobiti na više načina, a jedan od njih možete vidjeti na slici (3.7). Primjer je preuzet iz [21], a dodatna pojašnjenja i primjere možete pronaći u [6, pogl. 10.5.].



Slika 3.7: Duboka rekurentna neuronska mreža. Svaki sloj mreže ima svoje matrice težine W i vektore pristranosti b koji nisu posebno naznačeni na slici.

Problem učenja dugih veza

Problem učenja dugih veza je što čak i ako pretpostavimo da su parametri takvi da je mreža stabilna (nema problema s memorijom, niti problema s gradijentima) veza koja je daleko će imati eksponencijalno manju težinu od one kraće veze. Jedan od osnovnih problema koji se javlja je problem nestajućeg (engl. *vanishing*), odnosno eksplodirajućeg (engl. *exploding*) gradijenta.

Rekurentne neuronske mreže uključuju kompoziciju iste funkcije mnogo puta i te kompozicije mogu imati izrazito nelinearno ponašanje. Štoviše, neke od tih kompozicija uključuju matrična množenja. Promotrimo sljedeću jednadžbu koja se javlja u skrivenim slojevima i pretpostavimo da je parametar pristranosti jednak nuli

$$\boldsymbol{h}^{(t)} = \boldsymbol{W}^T \boldsymbol{h}^{(t-1)} = \cdots = (\boldsymbol{W}^t)^T \boldsymbol{h}^{(0)} \tag{3.15}$$

Pretpostavimo da matricu $\boldsymbol{W}$ možemo prikazati kao

$$\boldsymbol{W} = Q\Lambda Q^T, \tag{3.16}$$

gdje je Q ortogonalna matrica, a Λ je dijagonalna matrica, koja na dijagonali ima svojstvene vrijednosti matrice $\boldsymbol{W}$. Uvrštavanjem prethodne jednadžbe u izraz za $\boldsymbol{h}^{(t)}$ dobijemo

$$\boldsymbol{h}^{(t)} = Q\Lambda^t Q^T \boldsymbol{h}^{(0)}. \tag{3.17}$$

U slučaju da su svojstvene vrijednosti manje od jedan sa svakim množenjem one postaju manje i na kraju dobijemo matricu koja ima zanemarivo male elemente. S druge strane, ako su svojstvene vrijednosti veće od jedan svakim množenjem se te vrijednosti povećavaju i dobijemo matricu s jako velikim vrijednostima. Zbog toga može doći do eksplodiranja, odnosno nestajanja gradijenta.

Da bi RNN spremio prijašnja stanja tako da su ona otporna na manje perturbacije nužno je da RNN uđe u prostor gdje gradijent nestaje. Ako model može predstaviti duge veze, gradijent dugih veza će biti eksponencijalno manji od gradijenta bližih veza, a kako bi mreža naučila duge veze potrebno je puno vremena. Naime, informacija o ovisnosti s udaljenijim informacijama će često biti prekrivena s bližim informacijama.

U nastavku ću navesti jednu metodu koja pomaže kod eksplodirajućih gradijenata i jednu metodu za nestajuće gradijente. Više metoda i detalja možete pronaći u [6, pogl. 10.8–10.12].

Metoda odsijecanja gradijenta (engl. *gradient clipping*)

Metoda odsijecanja gradijenta sprječava da gradijent eksplodira. Ideja je da reskaliramo gradijent ako je on prevelik kako bi korak pri ažuriranju parametara bio manji. Jedan od načina kako da to napravimo je :

$$\text{ako je } \|g\| > v, \text{ tada } g \leftarrow \frac{g^v}{\|g\|}, \tag{3.18}$$

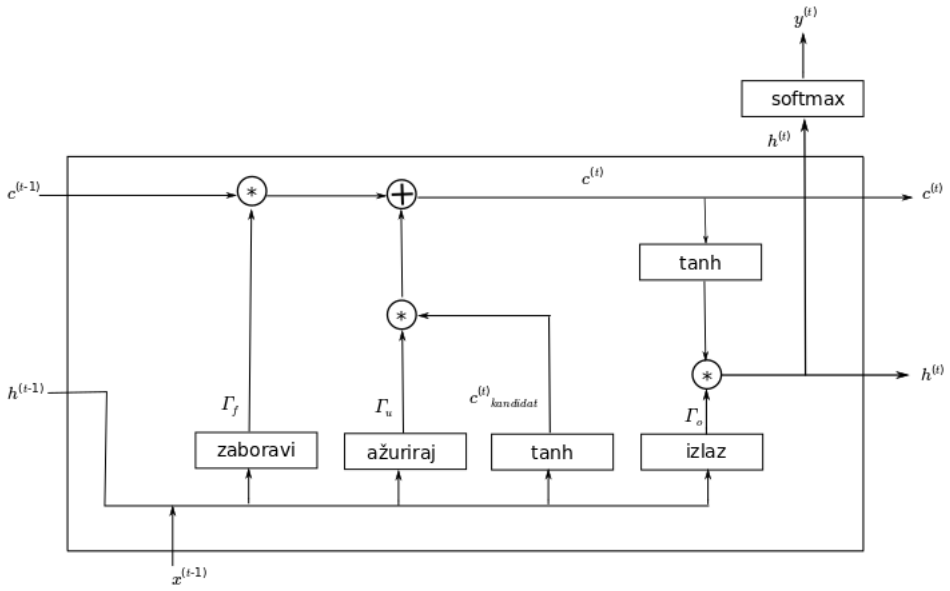
gdje je v unaprijed zadani prag (engl. *threshold*), a g je promatrani gradijent. Ako prilikom ažuriranja parametara koristimo reskalirane gradijente ti parametri će i dalje imati isti smjer kao i originalni gradijent. Razlika je u tome što je norma tih parametara ograničena, što onda sprječava da njihovim potenciranjem dođe do eksplodiranja, odnosno naglog rasta gradijenta i samim time brze promjene parametara.

Long-short term memory (LSTM)

Long-short term memory (LSTM) je posebna vrsta rekurentnih neuronskih mreža. Korištenjem *LSTM* rekurentne mreže se rješava problem nestajućih gradijenata i za njih je

specifično da nemaju problem s pamćenjem informacija tijekom dužeg vremenskog perioda. *LSTM* jednako kao i obične rekurentne mreže ima niz ponavljajućih modula, a razlika je u tome što kod RNN imamo jednostavan rekurentni čvor u kojem se obično izračunava tangens hiperbolni. Kod *LSTM* taj čvor uključuje više operacija i običajno ga nazivamo memorijska jedinica.

U nastavku ću proći kroz sve dijelove memorijske jedinice koju možete vidjeti i na slici 3.8. Za početak uvodimo oznake $\Gamma_f, \Gamma_u, \Gamma_o$ za propusnice zaboravljanja (engl. *forget*),



Slika 3.8: Memorijska jedinica LSTM

ažuriranja (engl. *update*) i izlaza (engl. *output*) te oznaku za potencijalnu novu vrijednost $\tilde{c}^{(t)}$.

Propusnica zaboravljanja postavlja težinu između 0 i 1 te određuje hoćemo li trenutno stanje zaboraviti i zamijeniti ga novim stanjem ili ćemo zadržati trenutno stanje. Ona se računa kao

$$\Gamma_f = \sigma(W_f[h^{(t-1)}, x^{(t)}] + b_f), \tag{3.19}$$

gdje je σ sigmoid funkcija, W_f matrica težina i b_f vektor pristranosti. Slijedeći korak je izračunavanje propusnice ažuriranja koja određuje koji parametri će se ažurirati. Ona se

računa kao

$$\Gamma_u = \sigma(W_u[h^{(t-1)}, x^{(t)}] + b_u), \tag{3.20}$$

gdje je σ sigmoid funkcija, W_u matrica težina i b_u vektor pristranosti. Nakon toga pomoću tangensa hiperbolnog se određuje potencijalna nova vrijednost formulom

$$\tilde{c}^{(t)} = \tanh(W_c[h^{(t-1)}, x^{(t)}] + b_c), \tag{3.21}$$

gdje je je W_c matrica težina i b_c vektor pristranosti. Pri ažuriranju stanja u trenutku t prvo stanje u trenutku $(t-1)$ pomnožimo s propusnicom zaboravljanja. Zatim potencijalnu novu vrijednost množimo s propusnicom ažuriranja da odredimo nove vrijednosti i na kraju to zbrojimo kako bi dobili vrijednost stanja u trenutku t

$$c^{(t)} = \Gamma_u * \tilde{c}^{(t)} + \Gamma_f * c^{(t-1)}. \tag{3.22}$$

Uočimo da ako je vrijednost propusnice zaboravljanja blizu 0 da tada zaboravljamo trenutno stanje, a ako je blizu jedan da tada ostaje veliki utjecaj trenutnog stanja. Slično vrijedi i za propusnicu ažuriranja. Ako je blizu 0, ažuriranje će malo utjecati na novo stanje, a ako je bliže 1, tada će vrijednosti od $\tilde{c}(t)$ imati značajan utjecaj.

Naposljetku trebamo odlučit što će biti izlazna vrijednost i za to koristimo sigmoid funkciju koja nam kaže koji dio stanja će ostati te primijenimo tangens hiperbolni na $c^{(t)}$ koji nam vrati vrijednosti između -1 i 1 i na kraju te dvije vrijednosti pomnožimo

$$\Gamma_o^{(t)} = \sigma(W_o[h^{(t-1)}, x^{(t)}] + b_o) \tag{3.23}$$

$$h^{(t)} = \Gamma_o^{(t)} * tanh(c^{(t)}), \tag{3.24}$$

gdje je je W_o matrica težina i b_o vektor pristranosti.

Napominjem kako znak $*$ u prethodnim jednadžbama označava množenje po elementima, dok je znak za obično množenje izostavljen. Detaljan opis *LSTM*-a možete pronaći u [8], dok jednostavno i vizualno objašnjenje možete pronaći na blogu [22].

3.4 Primjena

U nastavku ću ukratko objasniti kako su u dva rada koristili različite varijante rekurentne neuronske mreže za te kakve su rezultate postigli.

Prvi primjer

U prvom primjeru ([15]) predlažu da se protok vozila modelira kao difuzijski proces na usmjerenom grafu i predstavljaju difuzijske konvolucijske rekurentne neuronske mreže (engl. *diffusion convolutional recurrent neural networks (DCRNN))*.

DCRNN integrira difuzijsku konvoluciju, te slijed u slijed (engl. *sequence to sequence*) arhitekturu i tehniku planiranoga uzorkovanja (engl. *scheduled sampling*). Za modeliranje prostorne ovisnosti koristili su dvosmjerni graf slučajne šetnje, a vremensku ovisnost su modelirali pomoću rekurentnih neuronskih mreža.

Mrežu prometnica predstavljaju kao težinski usmjereni graf $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{W})$, gdje je $\mathcal{V}$ N-člani skup čvorova (senzora), $\mathcal{E}$ je skup bridova između čvorova i $\mathbf{W} \in \mathbb{R}^{N \times N}$ matrica težina, odnosno stvarne udaljenosti između čvorova. Protok prometa na grafu $\mathcal{G}$ označava se kao signal $\mathbf{X} \in \mathbb{R}^{N \times P}$, gdje je P broj atributa koji se opaža u svakom senzoru, a signal u trenutku t se označava s $\mathbf{X}^{(t)}$.

Difuzijski proces je karakteriziran slučajnom šetnjom na grafu $\mathcal{G}$ i s matricom promjene stanja $D_o^{-1}\mathbf{W}$, gdje je $D_o = diag(\mathbf{W1})$. Koriste i difuzijski proces u obrnutom smjeru, jer je dvosmjerna difuzija fleksibilnija u opisivanju uzvodnog i nizvodnog protoka prometa.

Difuzijska konvolucija primijenjena na signal $\mathbf{X} \in \mathbb{R}^{N \times P}$ i filter f_θ je definirana s:

$$\mathbf{X}_{:,p \star \mathcal{G}} f_\theta = \sum_{k=0}^{K-1} \Big(\theta_{k,1}(D_o^{-1}\mathbf{W})^k + \theta_{k,2}(D_I^{-1}\mathbf{W})^k)\Big)\mathbf{X}_{:,p}, \text{ za } p \in \{1, \ldots, P\}, \tag{3.25}$$

gdje su $\theta_{k,1}, \theta_{k,2}$ parametri filtera, a $D_o^{-1}\mathbf{W}$, $D_I^{-1}\mathbf{W}$ su matrice promjene stanja.

Pri definiranju rekurentne neuronske mreže su koristili *gated recurrent units (GRU)*, o kojima više možete pročitati u [6, str. 411]. Matrično množenje u definiciji GRU su zamjenili s difuzijskom konvolucijom i tako došli do *diffusion convolutional gated recurrent unit (DCGRU)* definiranu s jednadžbama:

$$\mathbf{r}^{(t)} = \sigma(\Theta_{r \star \mathcal{G}}[\mathbf{X}^{(t)}, \mathbf{H}^{(t-1)}] + b_r) \tag{3.26}$$

$$\mathbf{u}^{(t)} = \sigma(\Theta_{u \star \mathcal{G}}[\mathbf{X}^{(t)}, \mathbf{H}^{(t-1)}] + b_u) \tag{3.27}$$

$$\mathbf{C}^{(t)} = \tanh(\Theta_{C \star \mathcal{G}}[\mathbf{X}^{(t)}, (\mathbf{r}^{(t)} \odot \mathbf{H}^{(t-1)})] + b_C) \tag{3.28}$$

$$\mathbf{H}^{(t)} = \mathbf{u}^{(t)} \odot \mathbf{H}^{(t-1)} + (1 - \mathbf{u}^{(t)}) \odot \mathbf{C}^{(t)}, \tag{3.29}$$

gdje $\odot$ označava množenje po elementima, a $\mathbf{X}^{(t)}$ i $\mathbf{H}^{(t)}$ su ulazne i izlazne vrijednosti u trenutku t, $\mathbf{r}^{(t)}, \mathbf{u}^{(t)}$ su propusnice zaboravljanja, odnosno ažuriranja u trenutku t. $\Theta_r, \Theta_u, \Theta_C$ su parametri, a $\star\mathcal{G}$ označava difuzijsku konvoluciju definiranu u jednadžbi (3.25).

Model se sastoji od enkodera i dekodera, koji su oboje rekurentne neuronske mreže s DCGRU jedinicama. Ulazna vrijednost za enkoder su povijesni podaci, a njegov izlaz se koristi za inicijalizaciju dekodera. Dekoder radi predviđanja na osnovu pravih vrijednosti ili izlaza modela. Mreža je trenirana tako da se minimizira funkcija vjerodostojnosti koristeći BPTT. Više o enkoderu i dekoderu možete pročitati u [6, str. 396].

Za eksperimentalni dio su koristili dva skupa podataka. Prvi je METR-LA koji sadrži informacije prikupljene s 207 senzora tijekom 4 mjeseca u Los Angelesu, a drugi je PEMS-BAY koji sadrži informacije prikupljene s 325 senzora tijekom 6 mjeseci u Bay Area-u.

Podaci su agregirani u intervale od 5 minuta, a matricu susjedstva je definirana s $W_{i,j} = \exp\Big(- \frac{\text{dist}(v_i, v_j)^2}{\sigma^2}\Big)$, ako je $\text{dist}(v_i, v_j) \leq \kappa$, inače je 0. $W_{i,j}$ predstavlja težinu brida između vrhova v_i i v_j, σ je standardna devijacija udaljenosti i κ je zadani prag.

U radu su detaljnije istražili utjecaj vremenske i prostorne ovisnosti pomoću različitih varijanti DCRNN, a detalje potražite u radu na 6. i 7. stranici. Pokazali su da DCRNN ima glatka predviđanja i uz male oscilacije u brzini, što pokazuje robusnost modela. Dobro predviđa nagle promjene u brzini prometa te početak i kraj gužvi.

Drugi primjer

U ovom radu ([3]) su napravili arhitekturu od spojene dvosmjerne i jednosmjerne LSTM (SBU-LSTM) neuronske mreže za predviđanje brzine na mreži prometnica. Dvosmjerni LSTM sloj je korišten da bi se obuhvatila prostorna svojstva i dvosmjerna vremenska ovisnost u povijesnim podacima.

Ideja dvosmjerne LSTM neuronske mreže je zasnovana na dvosmjernoj RNN neuronskoj mreži o kojoj više možete pročitati u [6, str. 394.]. Ona obrađuje nizove podataka od početka prema kraju te od kraja prema početku pomoću dva različita skrivena sloja. Izlazna vrijednost BDLSTM sloja je y_t koji se dobije kao $y_t = \sigma\Big(\overrightarrow{h}, \overleftarrow{h}\Big)$, gdje su $\overrightarrow{h}, \overleftarrow{h}$ vrijednosti dobivene prolazom unaprijed, odnosno unatrag, a σ je funkcija koja kombinira te dvije vrijednosti.

Ponekad se, iz različitih razloga, dogodi da senzori koji prikupljaju informacije u nekom trenutku ne rade i zbog toga u skupovima podataka mogu nedostajati pojedine vrijednosti. U ovom radu su tom problemu doskočili na sljedeći način: ako u nekom trenutku t nedostaje informacija X_t, odnosno $X_t = \emptyset$, onda se u procesu učenja taj korak preskače i izračunato stanje u $(t-1)$ koraku se prosljeđuje kao ulazna vrijednost u $(t+1)$ koraku, a izlazna vrijednost u koraku t je $\emptyset$.

Duboke LSTM arhitekture se sastoje od više LSTM slojeva, pri čemu se izlazna vrijednost jednog LSTM sloja koristi kao ulazna vrijednost drugog sloja. U predloženoj arhitekturi za prvi sloj uzimaju BDLSTM kako bi pomoću njega naučili korisne informacije o prostornim ovisnostima. Pri predviđanju budućih vrijednosti brzine, na posljednji sloj arhitekture treba koristiti samo naučene značajke, pa je LSTM sloj dobar izbor za posljednji sloj.

U eksperimentalnom dijelu su koristili dva skupa podataka. Prvi je sadržavao informacije prikupljene s četri povezane autoceste u Seattle-u i ukupno 323 senzora. Podaci su prikupljeni tijekom 2015. godine u intervalima od 5 minuta. Drugi skup sadrži podatke za cijelu 2012. godinu, u intervalima od 5 minuta, a podaci su prikupljeni na 1014 senzora. Cilj je na osnovu niza od 10 vremenskih koraka predvidjeti vrijednosti brzina u idućem trenutku. Za početak su predloženu metodu usporedili s klasičnim metodama koje nisu u mogućnosti ukomponirati prostornu ovisnost. S obzirom na to rezultate su usporedili na

podacima sa samo jednog segmenta. Predložena metoda je postigla bolje rezultate od SVR (engl. *Support Vector Regression*) metode, slučajnih šuma, rekurentne neuronske mreže s GRU jedinicama. Zatim su SBU-LSTM usporedili s modelima koju imaju ukomponiranu prostornu komponentu. Rezultati su uspoređeni s LSTM mrežom, s mrežom od n LSTM slojeva i jednim potpuno povezanim slojem, modelom od n slojeva BDLSTM i višeslojnom LSTM mrežom koja u obzir uzima dan i vrijeme. Svi promatrani modeli najmanju *MAE* vrijednost postižu kada imaju dva sloja. Pokazalo se da SBU-LSTM postiže bolje rezultate u odnosu na navedene metode.

Dodatno su istraživali kako se mijenja točnost modela ovisno o broju promatranih vremenskih koraka i zaključili su da je točnost kada se promatra šest vremenskih koraka znatno varijabilnija i manja, nego kada se promatra više koraka. Zatim su promatrali kako se rezultati mijenjaju ako se mijenja veličina težinske matrice u posljednjem sloju i došli do zaključka da nema značajno velike promjene. Promatrali su utječe li red prostorne ovisnosti ulaznih podataka na učenje prostornih svojstava i zaključili su da nema utjecaja. Zatim su za ulazne podatke uzeli različite kombinacije brzine, količine prometa i okupiranosti promatranog segmenta i zaključili su da kada se kombiniraju sve tri veličine da su rezultati malo bolji.

Svi do sad navedeni rezultati su dobiveni koristeći skup podataka koji sadrži četiri autoceste. Da bi provjerili skalabilnost modela testirali su ga na drugom skupu podataka koji sadrži više različitih prometnica. Pokazalo se da model i na takvoj mreži prometnica postiže dobre rezultate. Pokazalo se da odabrani model dobro detektira gužve koje se periodično ponavljaju (jutarnje, popodnevne gužve), dok neočekivane gužve ne predviđa dobro.

Poglavlje 4

Primjer

U prethodnim dijelovima ovog rada je objašnjeno kako se različiti modeli vremenskih nizova i različite varijante rekurentne neuronske mreže mogu primijeniti za rješavanje problema protoka vozila. U nastavku je cilj prikazati kako pomoću STARMA modela i jednostavne LSTM neuronske mreže možemo predvidjeti broj vozila na različitim lokacijama u mreži prometnica.

4.1 Opis skupa podataka

Podaci su preuzeti s natjecanja koje je bilo dio *IEEE International Conference on Data Mining 2010*, u Sydney, Australija. Jedan od podzadataka je bilo predviđanje broja vozila u određenom periodu i podatke korištene za taj zadatak ću iskoristi za svoja rješenja.

Podaci su simulirani pomoću *Traffic Simulation Framework (TSF)* na ulicama Varšave u Poljskoj. Segmente na kojima su podaci simulirani možete vidjeti na slici 4.1. Simulacije su bazirane na modelu protoka prometa kojeg su dali Nagel i Schreckenberg. Ideja je da se cesta podjeli na blokove, pri čemu je svaki blok prazan ili zauzet, odnosno u bloku se nalazi ili se ne nalazi vozilo. Svako vozilo je autonomni agent, koji ima svoju početnu i krajnju točku, kao i svoju rutu. Neki od promatranih segmenata ceste se nalaze na velikim i prometnim cestama, dok su drugi na manjim, sporednim cestama.

U skupu za treniranje se nalazi 1 000 sati simulacija koje su podijeljene u 100 desetsatnih, nezavisnih ciklusa. Svaki red sadrži 20 vrijednosti, dvije vrijednosti za 10 promatranih segmenata, po jedna vrijednost za svaki smjer. Jedan red u skupu za treniranje odgovara jednoj minuti simulacije. S obzirom na to za kreiranje modela je dostupno 60 000 primjera. Najmanja uočena vrijednost na svim segmentima je nula, dok se maksimalne vrijednosti kreću od 44 na osmom segmentu do 101 na šestom segmentu. Prosječne vrijednosti broja automobila u jednoj minuti se kreću od 5.16 do 23.77.

Skup za testiranje sadrži drugih 1 000 sati simulacije, podijeljenih u blokove od 60 minuta, pri čemu je prvih 30 minuta svakog sata dostupno, a cilj je predvidjeti ukupan broj vozila u periodu od 41. do 50. minute. Simulirano je 100 desetsatnih, nezavisnih ciklusa, ali blokovi od 30 minuta su nasumično permutirani, tako da se u skupu za testiranje ne može pretpostaviti ovisnost između dva ili više susjednih blokova.

Na natjecanju su za evaluaciju rješenja koristili RMSE (engl. *root mean squared error*) metriku koja je definirana s

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}, \tag{4.1}$$

gdje je n broj primjera u skupu za testiranje, y_i je prava vrijednost, a $\hat{y}_i$ je predviđena vrijednost. Radi usporedbe u radu je korištena ista metrika.

4.2 Pobjednička rješenja

Podaci su preuzeti s natjecanja, a u nastavku su ukratko opisani najbolji pristupi i navedeni njihovi rezultate, a više detalja može se pronaći na blogu [23].

Ideja rješenja s najmanjom vrijednosti RMSE metrike je model koji je težinska linearna kombinacija prediktora. Sastoji se od 20 prediktora koji su dobiveni na osnovu tri različita algoritma s različitim parametrima: LLS (engl. *linear least squares*), SVD dekompozicije (engl. *singulare value decomposition*) i ograničenog Boltzmannovog stroja (engl. *restricted Boltzmann Machine* (RBM)).

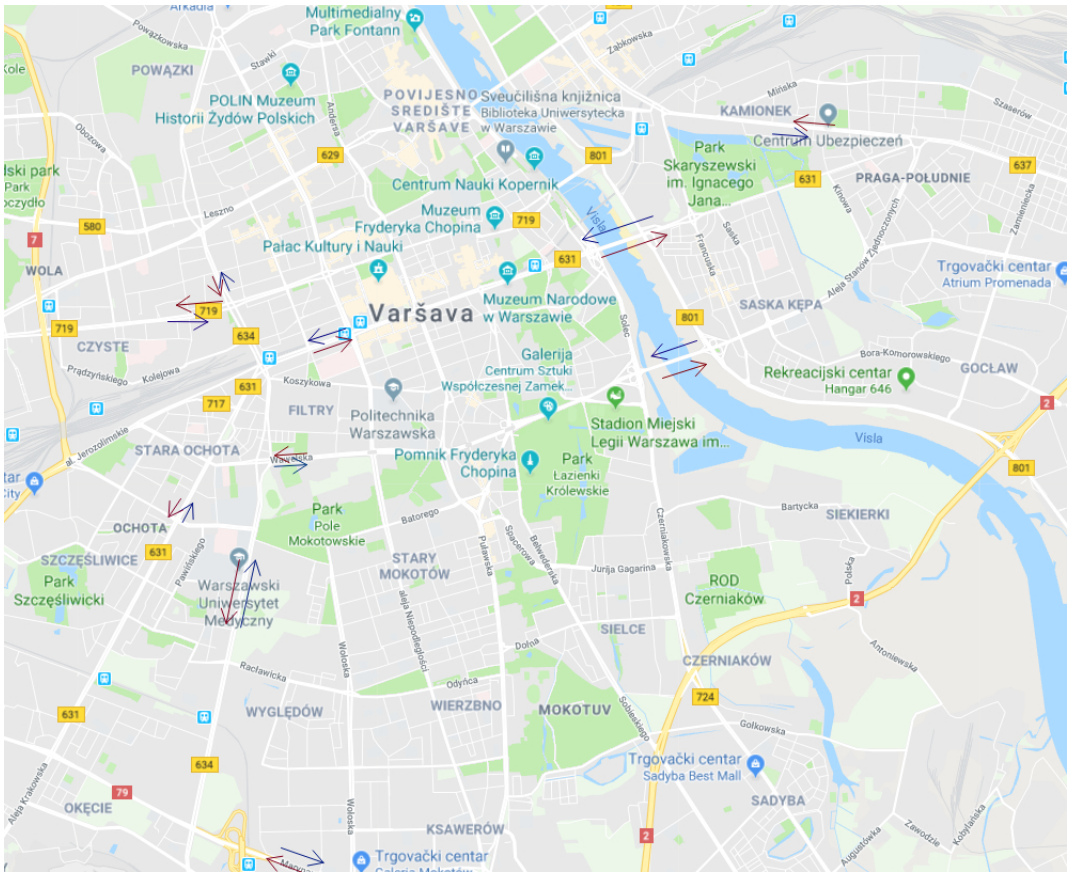
U drugom najboljem rješenju je za svaki segment napravljen poseban model pomoću neke jednostavne metode strojnog učenja poput slučajnih šuma ili k-najbližih susjeda. Zaključeno je da različiti modeli iskorištavaju različite informacije iz skupa podataka i zbog toga je za krajnji model napravljena konveksna kombinacija koja minimizira globalnu grešku.

Treće mjesto je osvojilo rješenje u koje je tri slučajne šume, pri čemu je svaka slučajna šuma definirana na drugačijem skupu podataka. Prvi skup se sastoji od 1 000 točaka i svaka točka odgovara prvoj polovici sata koja je dostupna. Drugi skup se sastoji od prvog skupa i primjera nastalih pomacima od jedne minute tijekom deset minuta. Treći skup se sastoji od drugog skupa i primjera nastalih pomoću svih mogućih pomaka.

RMSE vrijednosti koje su navedena rješenja postigla nalaze se u tablici 4.1.

4.3 Rješenje pomoću STARMA modela

Dekompozicija vremenskog niza $\{X_t\}$ je dana kao suma trenda, sezonalne komponente te ostatka i treba procijeniti svaku od komponenata. Osim toga, za STARMA model je



Slika 4.1: Prikaz promatranih segmenata

Mjesto	RMSE
1.	25.2327
2.	25.4167
3.	25.4337

Tablica 4.1: Vrijednosti RMSE metrike pobjedničkih rješenja

potrebno odrediti i matrice susjedstva. U ovom primjeru su određene matrice susjedstva prvog W_1 i drugog reda W_2. Susjedi prvog reda su oni koji su blizu promatranog segmenta i promet može doći do njega putem koji ne prolazi kroz druge segmente. Susjedi drugog reda su udaljeniji segmenti iz kojih promet može doći, a da pripadna putanja ne uključuje druge segmente ili putanje koje prolaze kroz mnogo ulica koje nisu među promatranim segmentima. Karta s označenim segmentima prikazana je na slici 4.1, dok susjede prvog i

Segment	Smjer	Susjed 1.reda	Susjed 2.reda
Most Poniatowskiego 1	E->W	Grochowska 1	Most Łazienkowski 2
Most Poniatowskiego 2	W->E	Aleje Jerozolimskie 2	Most Łazienkowski 1
Grójecka 1	S->N	Zwirki i Wigury 1	Marynarska 2
Grójecka 2	N->S	Wawelska 2	Aleje Jerozolimskie 1
Most Łazienkowski 1	E->W	Grochowska 1	Most Poniatowskiego 2
Most Łazienkowski 2	W->E	Wawelska 1	Most Poniatowskiego 1
Aleje Jerozolimskie 1	E->W	Most Poniatowskiego 1	Most Łazienkowski 1, Wawelska 2
Aleje Jerozolimskie 2	W->E	Towarowa 2	Wawelska 2 , Grójecka 1, Prosta 1
Marynarska 1	W->E	Zwirki i Wigury 2	Grójecka 2
Marynarska 2	E->W	Most Łazienkowski 1	Most Poniatowskiego 1
Zwirki i Wigury 1	S->N	Marynarska 2	Grójecka 2
Zwirki i Wigury 2	N->S	Wawelska 2	Grójecka 2, Aleje Jerozolimskie 1
Prosta 1	W->E	Towarowa 1	Aleje Jerozolimskie 1, Grochowska 2
Towarowa 1	S->N	Prosta 2	Aleje Jerozolimskie 1, Zwirki i Wigury 1, Wawelska 2
Grochowska 1	W->E	Prosta 1	Most Poniatowskiego 1,
Grochowska 2	E->W	Most Poniatowskiego 2	Most Łazienkowski 2
Wawelska 1	W->E	Zwirki i Wigury 1	Grójecka 1
Wawelska 2	E->W	Most Łazienkowski 1	Aleje Jerozolimskie 2
Towarowa 2	N->S	Prosta 1	Grochowska 2
Prosta 2	E->W	Towarowa 2	Grochowska 2, Aleje Jerozolimskie 1

Tablica 4.2: Tablica susjeda prvog i drugog reda.

drugog reda možete vidjeti u tablici 4.3.

Prvo sam promatrala podatke grupirane u intervale od 10 minuta. Svaki segment je analiziran zasebno i pokazalo se da u danim podacima nema trenda te je pomoću *Augmented Dickey-Fuller* testa utvrđeno da je dani niz stacionaran. S obzirom na to da su za svaki sat dostupna tri podatka, vrijednost u trenutku t možemo predvidjeti pomoću vrijednosti u trenutcima $(t-1)$, $(t-2)$, $(t-3)$. Prije nego su parametri procjenjeni podacima je oduzeto uzoračko očekivanje i podjeljeni su sa uzoračkom standardnom devijacijom. Dobiveni

model sa statistički značajnim koeficijentima, koji opisuje promatrani niz X_t je u nastavku

$$\begin{aligned} X_t =& 0.925505X_{t-1} + 0.025027X_{t-2} + 0.149330X_{t-1}W_1 - 0.152193X_{t-2}W_1 \\ &+ 0.158X_{t-1}W_2 - 0.173X_{t-2}W_2 - 0.316583\varepsilon_{t-1} - 0.175865\varepsilon_{t-2} \\ &- 0.131741\varepsilon_{t-3} - 0.074475W_1\varepsilon_{t-1} + 0.056770W_1\varepsilon_{t-2} - 0.060640W_1\varepsilon_{t-3} \\ &- 0.1564W_2\varepsilon_{t-1} + 0.0692W_2\varepsilon_{t-2} + 0.0557W_2\varepsilon_{t-3} + \varepsilon_t. \end{aligned} \tag{4.2}$$

Uočavamo da su koeficijenti uz X_{t-1} najveći, što nam govori da najviše utjecaja na količinu prometa u trenutku t ima vrijednost u prethodnom trenutku.

Traži se da na osnovu prvih trideset minuta predvidimo vrijednost od 41. do 50. minute što znači da moramo napraviti predviđanje dva koraka unaprijed. Recimo da imamo model opisan jednadžbom

$$Y_t = \alpha_1 Y_{t-1} + \alpha_2 Y_{t-2} + \alpha_3 Y_{t-3} + \alpha_4 Y_{t-4} + \beta_1 \varepsilon_{t-1} + \varepsilon_t \tag{4.3}$$

i da želimo odrediti predviđanje dva koraka unaprijed, a poznato nam je T vrijednosti. Kako bi dobili predviđenu vrijednost u koraku $T + 1$ u jednadžbi 4.3 zamijenimo t s $T + 1$, pa dobijemo

$$Y_{T+1} = \alpha_1 Y_T + \alpha_2 Y_{T-1} + \alpha_3 Y_{T-2} + \alpha_4 Y_{T-3} + \beta_1 \varepsilon_T + \varepsilon_{T+1}. \tag{4.4}$$

Vrijednosti Y_T, Y_{T-1}, Y_{T-2} su nam poznate, u trenutku T imamo grešku e_T, dok je buduća greška, ε_{T+1}, nepoznata i tu vrijednost zamijenimo s nulom. Dobijemo da je predviđena vrijednost u trenutku $T + 1$ na osnovu T dostupnih vrijednosti jednaka

$$\hat{Y}_{T+1} = \alpha_1 Y_T + \alpha_2 Y_{T-1} + \alpha_3 Y_{T-2} + \alpha_4 Y_{T-3} + \beta_1 e_T. \tag{4.5}$$

Da bi pomoću jednadžbe 4.3 dobili vrijednost u trenutku $T + 2$ potrebna nam je vrijednost u prethodnom trenutku, odnosno Y_{T+1}, i nju aproksimiramo s predviđenom vrijednošću $\hat{Y}_{T+1}$. Kada u jednadžbu 4.3 uvrstimo $T + 2$ umjesto t, svim budućim greškama pridružimo nulu i Y_{T+1} zamijenimo s $\hat{Y}_{T+1}$ dobijemo da je predviđena vrijednost u trenutku $T + 2$ jednaka

$$\hat{Y}_{T+2} = \alpha_1 \hat{Y}_{T+1} + \alpha_2 Y_T + \alpha_3 Y_{T-1} + \alpha_4 Y_{T-2}. \tag{4.6}$$

Navedeni princip za predviđanje je korišten i u ovom primjeru. Na kraju se na predviđene vrijednosti primjeni ista transformacija kao na ciljane vrijednosti na skupu koji koristimo za procjenu parametara te se računa greška modela. Greška modela pri predviđanju na skupu za treniranje je 175.9886, dok je na skupu za testiranje 218.8597.

Zatim sam napravila intervale od pet minuta i ponovila proceduru određivanja modela u kojemu za vrijednost u trenutku t možemo iskoristiti vrijednosti u trenutcima $(t - 1)$, $(t - 2)$, $(t - 3)$, $(t - 4)$, $(t - 5)$, $(t - 6)$. U ovom slučaju nisu bili prisutni trend i sezonalnost.

Nakon određivanja broja parametara i njihove procjene dobijemo da se niz može opisati slijedećom jednadžbom

$$\begin{aligned} X_t = &- 0.0246X_{t-1} + 0.9334X_{t-2} + 0.532903\varepsilon_{t-1} - 0.440664\varepsilon_{t-2} \\ &- 0.131359\varepsilon_{t-3} - 0.092240\varepsilon_{t-4} - 0.086149\varepsilon_{t-5} - 0.080255\varepsilon_{t-6} \\ &+ 0.146013W_1\varepsilon_{t-1} + 0.089412W_1\varepsilon_{t-2} - 0.052125W_1\varepsilon_{t-3} - 0.043828W_1\varepsilon_{t-5} \\ &- 0.083228W_1\varepsilon_{t-6} + 0.0353W_2\varepsilon_{t-1} + 0.0147W_2\varepsilon_{t-2} - 0.0091W_2\varepsilon_{t-3} \\ &+ 0.0077W_2\varepsilon_{t-4} + 0.0117W_2\varepsilon_{t-6} + \varepsilon_t. \end{aligned} \tag{4.7}$$

Kako bi predvidjeli ukupan broj vozila od 41. do 50. minute prvo moramo predvidjeti broj vozila u intervalima od pet minuta počevši od 31. minute pa sve do 40. minute. Za predviđanja je korišten postupak koji je prethodno opisan.

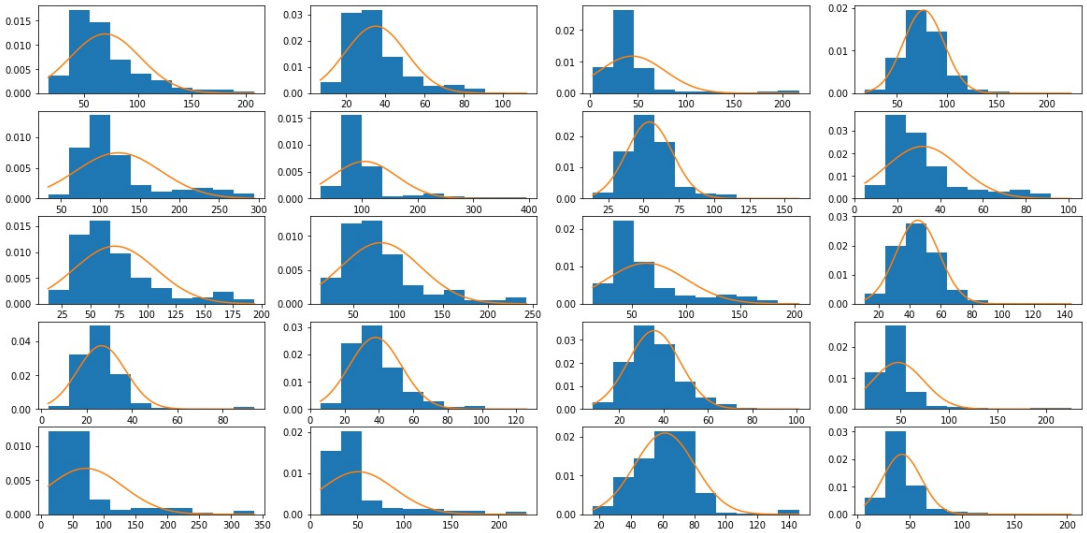
Greška na skupu za treniranje je 291.5537, a na skupu za testiranje je 306.1006, što su nešto veće greške nego kad promatramo intervale od 10 minuta. Jedan od mogućih razloga je i nakupljanje greške. Naime, da bi predvidjeli traženu vrijednost, moramo prije predvidjeti još četiri vrijednosti, a pri tome se pojavljuju buduće greške koje stavljamo na nulu, i to upravo nekoliko prvih grešaka u jednadžbi 4.7 čiji su koeficijenti veći u usporedbi s ostalim koeficijentima uz greške. Kada promotrimo koeficijente uz X_{t-1} i X_{t-2} vidimo da prethodnih pet minuta X_{t-1} ima manji utjecaj od pet minuta koje prethode X_{t-2}, odnosno da vrijednost u daljoj prošlosti ima veću utjecaj nego vrijednost u prethodnom trenutku. To upućuje da su promatrani intervali prekratki.

4.4 Rješenje pomoću LSTM neuronske mreže

Za drugo rješenje su odabrane jednostavne LSTM neuronske mreže. Podaci su grupirani u intervale od jedne, pet i deset minuta i za svaki tako dobiven skup podataka je određena jedna arhitektura. Zbog grupiranja podataka za predviđanje broja vozila od 41. do 50. minute je dostupno 30, šest odnosno tri podatka za intervale od jedne, pet odnosno deset minuta.

Prije učenja modela provjerena je normalnost podataka za svaki segment te su podaci normalizirani. Histogrami i pripadne funkcije gustoće za intervale od pet minuta, za povijesne podatke, su prikazani na slici 4.2. Osim povijesnih podataka normalizirani su i podaci koje želimo predvidjeti.

Glavna ideja je da se model napravi pomoću jednog ili više LSTM slojeva i jednog potpuno povezanoga sloja čija je aktivacijska funkcija linearna. Za optimizacijski algoritam je odabran RMSprop [6, str. 307]. Kao funkcija gubitka se koristi MSE (engl. *mean squared error*), a iz njenih vrijednosti ćemo izvaditi drugi korijen kako bi dobili RMSE i onda ćemo te rezultate usporediti s rezultatima pobjedničkih rješenja.



Slika 4.2: Histogram povijesnih podataka u intervalima od 5 minuta (skup za treniranje) i pripadna normalna funkcija gustoće. Jedan graf odgovara jednom promatranom segmentu.

Kada se vrijednosti od jedne minute agregiraju u intervale od pet, odnosno deset minuta, tada nam se skup za treniranje značajno smanjuje što utječe na učenje modela. Iako je uobičajeno se tijekom učenja modela nekoliko puta prođe po cijelom skupu podataka, odnosno da se napravi nekoliko epoha, u ovim primjerima se pokazalo se se greška znatno poveća ako je broj prolaza veći od jedan. Ako promatramo intervale od 10 minuta i arhitekturu LSTM(60,45)→LSTM(45,40)→LSTM(40,35) → LSTM(35,30) → Dense(30,20) , tada je u RMSE za 10 epoha gotovo tri puta veći u odnosu RMSE za jednu epohu.

U sva tri primjerima je isprobano nekoliko različitih arhitektura s različitim brojem primjera (engl. *batch size*) koje se koriste pri jednom ažuriranju težina te različitim veličinama skrivenih slojeva. S obzirom na to da složeni i veliki modeli imaju tendenciju da pretreniraju (engl. *overfitting*) za regularizaciju koristim *dropout*. Vjerojatnost da se određeni čvor izbaci je varirana i pokazalo se da se najbolji rezultati postižu kada je ta vjerojatnost jednaka 0.15.

Najbolji dobiveni modeli za sva tri primjera se nalaze u tablici 4.3. Za primjer gdje su korišteni intervali od jedne minute najbolji rezultati se postižu kada imamo jedan LSTM sloj i jedan potpuno povezani sloj. Za intervale od pet minuta najbolji rezultat se postiže s dva LSTM sloja i jednim potpuno povezanim slojem. U slučaju kada se koriste intervali od deset minuta najbolji rezultat je postigla kompliciranija mreža. Ona se sastoji od četiri LSTM sloja i jednog potpuno povezanoga sloja.

Uočavamo da su rezultati znatno lošiji od rezultata koji su postigli natjecatelji, na što

Agregacija	Arhitektura	RMSE
1 minuta	LSTM(600,120)→ Dense(120,20)	91.42
5 minuta	LSTM(120,110)→LSTM(110,100)→ Dense(100,20)	51.05
10 minuta	LSTM(60,50)→LSTM(50,30)→LSTM(30,30) → LSTM(30,30) → Dense(30,20)	56.39

Tablica 4.3: Vrijednosti RMSE metrike modela za podatke agregirane u intervale od jedne, pet i deset minuta. Prva vrijednost kod LSTM i Dense označava veličinu ulazna, a druga vrijednost veličinu izlaza.

sigurno utječe i mali broj primjera za učenje. Treniranje model s intervalima od jedne minute je vremenski zahtjevno, pa sam se ograničila na jednostavniju arhitekturu, ali postoji mogućnost da bi složenija arhitektura postigla bolje rezultate, iako su arhitekture s dva LSTM sloja dale lošije rezultate.

U skupu za treniranje su dostupne informacije za svih 60 minuta, za 1 000 sati, a u prethodnim primjerima je za učenje iskorišteno samo prvih 30 minuta svakog sata. Kako bi dobili više primjera za učenje iskorišteni su podaci i između 30. i 40. minute. Primjeri su dobiveni tako da se uzmu informacije od prve do 30. minute i predviđa se broj vozila od 41. do 50. minute, zatim se uzmu podaci od druge do 31. minute i predviđa se broj vozila od 42. do 51. minute i tako sve do primjera gdje se podaci od 30. do 40. minute koriste za predviđanje broja vozila od 51. do 60. minute. Tako za svaki sat, umjesto 30 primjera (samo od 1. do 30. minute) imamo 330 primjera. U prethodnom primjeru, prije agregiranja, je za odabir modela korišteno 30 000 primjera, a sad je dostupno 330 000 primjera. Tako dobiven skup podataka je ponovno agregiran u intervale od pet, odnosno deset minuta. Treniranje modela za intervale od jedne minute je i s manje primjera vremenski zahtjevno te ima najlošije rješenje, pa sam na augmentiranim podacima promatrala samo intervale od pet i deset minuta.

Kao i u prethodnom primjeru prvi korak je bila normalizacija podataka, nakon čega je uslijedio odabir arhitekture i prilagodba ostalih parametara. Već prve, jednostavne arhitektura od samo jednog LSTM sloja i jednog potpuno povezanoga sloja na obje duljine intervala, s deset epoha su postigle mnogo bolje rezultate nego što imaju modeli iz prethodnog primjera. Za intervale od pet minuta to je model LSTM(120,60)→Dense(60,20) s greškom 36.27, a za intervale od deset minuta arhitektura LSTM(60,60)→Dense(60,20) ima grešku 37.92.

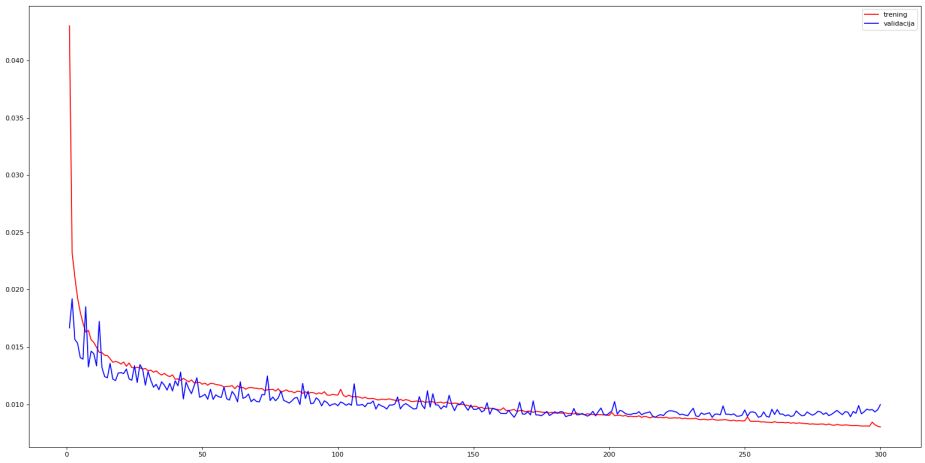
Kako bi se spriječilo pretreniranje ponovno se koristi *dropout* pri čemu je vjerojatnost da se čvor sa svim svojim vezama izbaci iz mreže varirana i ponovno je za taj parametar odabrana vrijednost 0.15. U tablici 4.4 je za obje duljine intervala, za najbolje dobivene modele prikazana korištena arhitektura, broj epoha i RMSE vrijednost.

Na slici (4.3) su prikazane krivulje koje predstavljaju grešku na skupu za treniranje,

Agregacija	Arhitektura	Broj epoha	RMSE
5 minuta	LSTM(120,90)→LSTM(90,60) →LSTM(60,60)→ Dense(60,20)	238	28.25
10 minuta	LSTM(60,40)→LSTM(40,30) → Dense(30,20)	270	29.54

Tablica 4.4: Modeli s većim skupom za treniranje. Prva vrijednost kod LSTM i Dense označava veličinu ulazna, a druga vrijednost veličinu izlaza.

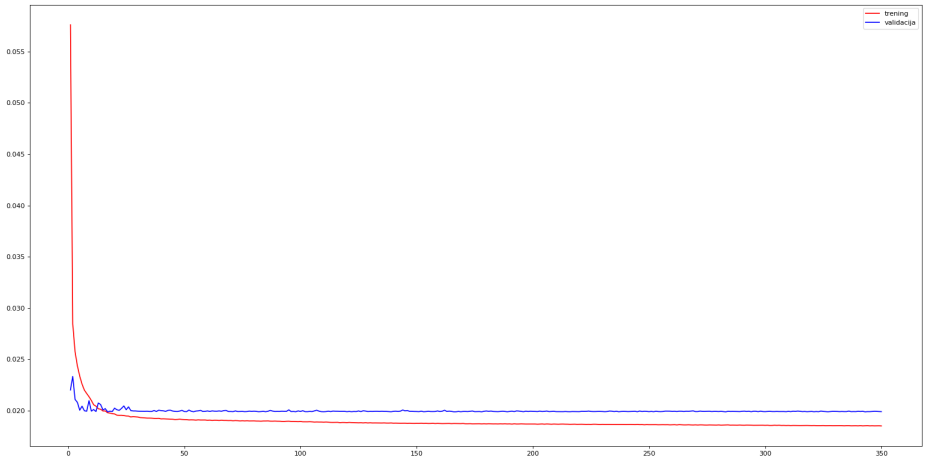
odnosno validaciju na kraju svake epohe pri treniranju modela za intervale od pet minuta. Uočimo da se greška na skupu za treniranje brzo približi greški na skupu za validaciju i da su onda blizu jedna druge, ali nakon dvjesto pedesete epohe greška na skupu za treniranje počinje rasti što upućuje na pretreniranje. S obzirom na to za krajnji model je korišteno manje epoha.



Slika 4.3: Greške na skupu za treniranje i validaciju tijekom 300 epoha modela za 5-minutne intervale na većem skupu podataka. Crvena krivulja je greška na skupu za treniranje, a plava na skupu za validaciju.

Na slici (4.4) su prikazane krivulje koje predstavljaju grešku na skupu za treniranje, odnosno validaciju na kraju svake epohe pri treniranju modela za intervale od deset minuta. Uočimo da greška na skupu za treniranje brzo postane manja od greške na skupu za validaciju te da su pripadne krivulje nakon stote epohe skoro paralele. Kako bi za krajnji

model dobili onaj s najmanjom greškom na skupu za testiranje broj iteracija je mijenjan te se pokazalo da model postiže najbolje rezultate s 270 epoha.



Slika 4.4: Greške na skupu za treniranje i validaciju tijekom 350 epoha modela za 10-minutne intervale na većem skupu podataka. Crvena krivulja je greška na skupu za treniranje, a plava na skupu za validaciju.

4.5 Zaključak

U prethodim dijelovima rada su opisane različite metode vremenskih nizova i dva pristupa s rekurentnim neuronskim mrežama za rješavanje problema protoka prometa. U ovom poglavlju cilj je bio prikazati kako iskoristiti osnovne ideje tih pristupa na drugom skupu podataka.

Svi modeli navedeni u prethodna dva potpoglavlja imaju veću grešku od pobjedničkih rješenja na natjecanju, pri čemu STARMA modeli imaju najveću grešku, zatim LSTM neuronske mreže s originalnim skupom podataka. Najbolje rezultate postižu LSTM neuronske mreže koje za treniranje koriste augmentirani skup podataka.

Kod STARMA modela za intervale od pet minuta smo vidjeli da je veći koeficijent uz informaciju dva trenutka prije, nego uz informaciju koja je neposredno prije. To nam sugerira da su intervali možda prekratki. S obzirom na to jedan potencijalni način kako

poboljšati točnost STARMA, ali i ostalih modela je da se promotre intervali drugačijih duljina.

Uočavamo da LSTM neuronske mreže doista imaju bolju točnost kada se treniraju na većem skupu, a taj skup se može još povećati. Naime, korišteni podaci su simulirani tako da jedan nezavisni ciklus čini sto sati informacija. Kod augmentiranja podataka u prethodnom primjeru su iskorišteni podaci od 30. do 40. minute svakoga sata i tako se povećao skup za treniranje. Augmentiranje podataka se može napraviti i na drugi način i to tako da se iskoristi činjenica da nezavisni ciklus čini sto sati. Način na koji se to može napraviti je da se uzmemo ciklus od sto sati, odnosno niz duljine 60 000, i s pomakom od jedne minute kao primjer, odnosno povijesne podatke, uzmemo trideset minuta, a njegovu ciljanu vrijednost odredimo kao sumu deset vrijednost, pri čemu je prva vrijednost udaljena deset koraka od posljednje vrijednosti povijesnih podataka. Navedeni postupak ponovimo za svih deset nezavisnih ciklusa i tako dođemo do još većeg skupa podataka. S obzirom na to da se u prethodnom primjeru pokazalo da na točnost LSTM neuronske mreže utjecaj ima veličina skupa za treniranje, druga mogućnost pomoću koje se može dobiti model s boljom točnošću je upravo augmentiranje podataka na način koji je prethodno opisan.

Bibliografija

[1] Bojana Dalbelo Bašić, Marko Čupić i Jan Šnajder, *Umjetne neuronske mreže*, `https://www.fer.hr/_download/repository/UmjetneNeuronskeMreze.pdf`, Umjetna inteligencija, Zavod za elektorniku, mikroelektroniku i inteligentne sustave Fakultet elektrotehnike i računarstva.

[2] D. Billings i J. Yang, *Application of the ARIMA Models to Urban Roadway Travel Time Prediction - A Case Study*, 2006 IEEE International Conference on Systems, Man and Cybernetics, sv. 3, 2006, str. 2529–2534, ISBN 1-4244-0099-6.

[3] Zhiyong Cui, Ruimin Ke i Yinhai Wang, *Deep Bidirectional and Unidirectional LSTM Recurrent Neural Network for Network-wide Traffic Speed Prediction*, (2018), arXiv:1801.02143.

[4] George E. P. Box, Gwilym M. Jenkins, Gregory C. Reinsel i Greta M. Ljung, *Time Series Analysis:Forecasting and Control (5th edition)*, John Wiley & Sons, Inc., 2015.

[5] Bidisha Ghosh, Biswajit Basu i Margaret O'Mahony, *Bayesian Time-Series Model for Short-Term Traffic Flow Forecasting*, Journal of Transportation Engineering **133 (3)** (2007).

[6] Ian Goodfellow, Yoshua Bengio i Aaron Courville, *Deep Learning*, MIT Press, 2016, `http://www.deeplearningbook.org`.

[7] W.K. Hastings, *Biometrika*, sv. 57, pogl. Monte Carlo sampling methods using Markov chains and their application, str. 97–109, 1970.

[8] Sepp Hochreiter i Jürgen Schmidhuber, *Long short-term memory*, Neural Computation **9(8)** (1997), 1735–1780.

[9] Yang Jiann-Shiou, *A Nonlinear State Space Approach to Arterial Travel Time Prediction*, (2006), `http://hdl.handle.net/11299/1020`, University of Minnesota Digital Conservancy.

[10] Yiannis I. Kamarianakis i Poulicos P. Prastacos, *Space–time modeling of traffic flow*, Computers & Geosciences **31** (2005), br. 2, 119 – 133, ISSN 0098-3004.

[11] Andrej Karpathy, *The Unreasonable Effectiveness of Recurrent Neural Networks*, http://karpathy.github.io/2015/05/21/rnn-effectiveness/, posjećeno 10. 10. 2018.

[12] Petra Lazić, *Statistički praktikum 2-vježbe*, https://web.math.pmf.unizg.hr/nastava/statpr2/materijali.html, vježbe iz kolegija Statistički praktikum 2 na diplomskom studiju PMF–MO, posjećeno 19. 9. 2018.

[13] Peter M. Lee, *Bayesian Statistics: An Introduction, 4th edition*, Wiley Publishing, 2012, ISBN 1118332571, 9781118332573.

[14] Sangsoo Lee i Daniel Fambro, *Application of Subset Autoregressive Integrated Moving Average Model for Short-Term Freeway Traffic Volume Forecasting*, Transportation Research Record: Journal of the Transportation Research Board **1678** (1999), 179–188.

[15] Yaguang Li, Rose Yu, Cyrus Shahabi i Yan Liu, *Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting*, (2018).

[16] Marco Lippi, Matteo Bertini i Paolo Frasconi, *Short-Term Traffic Flow Forecasting: An Experimental Comparison of Time-Series Analysis and Supervised Learning*, sv. 14(2), 2013, https://ieeexplore.ieee.org/document/6482260/.

[17] Krzyzanowski Michal, Kuna Dibbert Birgit i Schneider Jrgen, *Health effects of transport-related air pollution*, WHO Regional Office Europe, 2005.

[18] Dietmar P. F. Möller, *Introduction to Transportation Analysis, Modeling and Simulation: Computational Foundations and Multimodal Applications*, pogl. Transportation Models, str. 45–108, Springer London, 2014, ISBN 978-1-4471-5637-6, https://doi.org/10.1007/978-1-4471-5637-6_2.

[19] Attila M. Nagy i Vilmos Simon, *Survey on Traffic Prediction in Smart Cities*, Pervasive and Mobile Computing (2018), https://doi.org/10.1016/j.pmcj.2018.07.004.

[20] H. J. Newton, *TIMESLAB: A Time-series Analysis Laboratory*, Wadsworth and Brooks/Cole Publishing Company, 1996.

[21] Andrew Ng, Kian Katanforoosh i Younes Bensouda Mourri, *Deep RNNs, Deep Learning Specialization, Course: Sequence Models*, https://www.coursera.org/lecture/nlp-sequence-models/deep-rnns-ehs0S, posjećeno 10. 10. 2018.

[22] Christopher Olah, *Understanding LSTM Networks*, http://colah.github.io/posts/2015-08-Understanding-LSTMs/, posjećeno 10. 10. 2018.

[23] M. Pancewicz, *IEEE ICDM Contest-Overview of Top Solutions, part 1*, https://blog.tunedit.org/2010/10/26/ieee-icdm-contest-top-solutions-1/, posjećeno 31. 10. 2018.

[24] J. Brockwell Peter i A. Davis Richard, *Introduction to Time Series and Forecasting*, Springer, 2016, ISBN 9783319298528, https://doi.org/10.1007%2F978-3-319-29854-2.

[25] Phillip E. Pfeifer i Stuart Jay Deutsch, *A Three-Stage Iterative Procedure for Space-Time Modeling*, Technometrics **22** (1980), br. 1, 35–47, http://www.jstor.org/stable/1268381.

[26] Eleni Vlahogianni, Matthew Karlaftis i John Golias, *Short-term traffic forecasting: Where we are and where we're going*, Transportation Research Part C: Emerging Technologies **43** (2014), 3–19.

[27] Billy M. Williams, *Modeling and forecasting vehicular traffic flow as a seasonal stochastic time series process*, (1999), PhD dissertation, Univ. of Virginia, Charlottesville,Va.

[28] Billy M. Williams i Lester A. Hoel, *Modeling and Forecasting Vehicular Traffic Flow as a Seasonal ARIMA Process: Theoretical Basis and Empirical Results*, Journal of Transportation Engineering **129(6)** (2003), 664–672.

[29] Bing Yu, Haoteng Yin i Zhanxing Zhu, *Spatio-Temporal Graph Convolutional Networks: A Deep Learning Framework for Traffic Forecasting*, (2018), arXiv:1709.04875v4.

Sažetak

U ovom radu je istražen problem predviđanja protoka prometa u mreži prometnica. Složenost procesa koja je iza protoka prometa je iznimno velika i pri izgradnji modela treba voditi računa o mnogo čimbenika koji utječu na promet, a koji su navedeni u prvom poglavlju.

U drugom poglavlju je dan teoretski uvod u vremenske nizove te ARMA, ARIMA, SARIMA i STARIMA modele. Za svaki od navedenih modela je opisan neki od postojećih pristupa za modeliranje protoka prometa.

Treći dio rada je posvećen dubokom učenju te rekurentnim neuronskim mrežama. Opisana su dva različita pristupa, a svaki od njih koristi prilagođenu verziju rekurentnih neuronskih mreža. Prvi pristup koristi difuzijsku konvolucijsku rekurentnu neuronsku mrežu, a drugi dvosmjernu LSTM neuronsku mrežu.

Posljednje poglavlje sadrži primjenu STARMA modela i jednostavne LSTM neuronske mreže na podacima koji simuliraju protok prometa u Varšavi.

Summary

This thesis' emphasis has been on the problem of traffic flow forecasting in the road network. The process behind traffic flow is extremely complex, and when creating a model one should take into account many of the factors affecting traffic flow. The first chapter of the thesis explains those factors.

In the second chapter, there is a theoretical introduction to time series and ARMA, ARIMA, SARIMA, and STARIMA models. Some of the existing traffic flow modeling approaches have been described for the above-mentioned models.

The third part of the thesis includes an introduction to deep learning and recurrent neural networks. Two different approaches have been described, each of which uses a custom version of recurrent neural networks. The first approach uses a diffusion convolutional recurrent neural network, and the second bidirectional LSTM neural network.

The last chapter includes the use of STARMA model and simple LSTM neural networks on data simulating the traffic flow in Warsaw.

Životopis

Rođena sam 12.lipnja 1994. godine u Uslaru, Savezna Republika Njemačka. S nepune 4 godine s obitelji sam se doselila u Velimirovac gdje sam 2001. godine započela svoje osnovnoškolsko obrazovanje koje sam završila u Osnovnoj školi kralja Tomislava u Našicama. Išla sam u Prirodoslovno-matematičku gimnaziju u Našicama i nakon toga, 2013. godine sam upisala Preddiplomski sveučilišni studij Matematike na Prirodoslovno-matematičkom fakultetu u Zagrebu. Potom sam 2016. godine upisala Diplomski sveučilišni studij Matematična statistika na istom fakultetu. Tijekom četvrte i pete godine studija sam bila član neprofitne studentske organizacije eSTUDENT. Prvo kao član, a potom i kao voditeljica organizacijskog tima Mozgalo.