HO
GENT

Faculteit Bedrijf en Organisatie

Vergelijkende studie van voorspellingsmodellen voor tijdreeksen

Emiel Declercq

Scriptie voorgedragen tot het bekomen van de graad van
professionele bachelor in de toegepaste informatica

Promotor:
Johan Decorte
Co-promotor:
Stijn Lievens

Instelling:

Academiejaar: 2019-2020

Tweede examenperiode

Faculteit Bedrijf en Organisatie

Vergelijkende studie van voorspellingsmodellen voor tijdreeksen

Emiel Declercq

Scriptie voorgedragen tot het bekomen van de graad van
professionele bachelor in de toegepaste informatica

Promotor:
Johan Decorte
Co-promotor:
Stijn Lievens

Instelling:

Academiejaar: 2019-2020

Tweede examenperiode

Woord vooraf

Deze bachelorproef werd geschreven als eindwerk voor het afronden van de opleiding Toegepaste Informatica aan de Hogeschool Gent met specialisatie e-business.

Daarnaast wil ik ook nog ir. Johan Decorte bedanken voor de vlotte begeleiding van deze bachelorproef en Stijn Lievens voor het opnemen van het co-promotorschap.

Mijn ouders wil ook hartelijk ik bedanken voor alle steun, niet alleen tijdens deze scriptie maar gedurende mijn volledige studietraject. Zeker tijdens deze lockdown kan ik mij voorstellen ik dat dit op sommige momenten voor heel wat stress heeft veroorzaakt. Tenslotte verdient ook mijn zus, Edith Declercq een woordje van dank voor al het naleeswerk tot in de late uurtjes en de resem van aangereikte tips bij het schrijven van een paper.

Samenvatting

In deze paper zullen verschillende voorspellingstechnieken voor verschillende types tijdreeksen geanalyseerd worden. Er zal ook getest worden op verschillende types tijdreeksen. Het onderscheid tussen deze types wordt gemaakt op basis van 2 criteria namelijk seizoensgebondenheid en het aantal onafhankelijke variabelen. Deze zullen dan onderverdeeld worden in een al dan niet seizoensgebondenheid en een enkele of meerdere onafhankelijke variabelen. In totaal zullen er dus voorspellingen gemaakt worden voor 4 verschillende types tijdreeksen namelijk.

- Tijdreeksen met enkel de tijd als onafhankelijke variabele en 1 afhankelijke variabele zonder seizoensgebonden verband
- Tijdreeksen met enkel de tijd als onafhankelijke variabele en 1 afhankelijke variabele met een seizoensgebonden verband
- Tijdreeksen met 2 onafhankelijke variabelen waarvan 1 de tijd en 1 afhankelijke variabele zonder een seizoensgebonden verband
- Tijdreeksen met 2 onafhankelijke variabelen waarvan 1 de tijd en 1 afhankelijke variabele met een seizoensgebonden verband

De eerste voorspellingstechniek die onder de loep zal genomen is polynomiale regressie. De tweede techniek die zal toegepast worden is een ARIMA/VARMAX model. Tenslotte zal een recurrent neuraal netwerk van het type LSTM (Long Term Short Memory) op dezelfde data toegepast worden. Voor elk van deze 4 types tijdreeksen zal dan bepaald worden welk van de 3 technieken het beste resultaat zal opleveren.

Inhoudsopgave

Lijst van figuren

1. Inleiding

De hoeveelheid data die men kan verwerven in het digitale tijdperk waarin we de dag van vandaag leven is gigantisch zeker met de opkomst van het internet der dingen beter gekend onder de noemer *Internet of Things*. Deze data kan enorm uiteenlopend zijn, zo wordt onder andere de buitentemperatuur bijgehouden, maar ook de waarde van de aandelen van een bedrijf op de beurs, het aantal bezette plaatsen in een parking, het aantal mensen dat positief getest heeft op het coronavirus, ...

Zo kan het lijstje nog een hele tijd aangevuld worden, maar de net opgenoemde gegevens zijn niet enkel voorbeelden van data. Deze zaken variÃńren ook nog eens doorheen de tijd. Om dit in contrast te stellen met een ander voorbeeld zou een naam of een postcode van je geboorteplaats niet variÃńren en dus niet tijdsgebonden zijn. Een sequentie van data die tijdsgebonden is wordt benoemd als een tijdreeks ofwel een textitTime series.

Een voorbeeld hiervan zou het aantal dagelijks gewandelde kilometers van het afgelopen jaar zijn. Doordat dit een tijdreeks is zouden we het aantal dagelijks gewandelde kilometers van het volgende jaar kunnen voorspellen met behulp van bepaalde modellen. Daaruit zouden we dan bijvoorbeeld kunnen vaststellen dat er in de winter minder gewandeld zal worden. Het nut hiervan zou dan kunnen zijn dat men inziet dat men te weinig lichaamsbeweging zal hebben en daar dan zal op inspelen door een indoorsport te beoefenen zodat men toch nog voldoende lichaamsbeweging heeft. Bij dit voorbeeld is het verband zeer simpel en kan men zich afvragen waarvoor het opstellen van een model nuttig zou zijn aangezien dit vrij makkelijk af te leiden valt met het blote oog. Maar soms zal het verband een pak complexer zijn dan een seizoen waardoor het moeilijk te vatten valt voor het menselijk brein maar praktischer is om te identificeren door middel van een wiskundig model.

De modellen die onderzocht zullen worden zijn polynomiale regressie, ARIMA/VARMAX modellen en LSTM modellen. Daarnaast zal ook nog rekening gehouden met het aantal invoerparameters. Zo zullen modellen die gebruik maken van 1 invoerparameter benoemd worden als univariate modellen, modellen die gebruik maken van meerdere invoerparameters daarentegen zullen benoemd worden als multivariate modellen. Deze 2 types modellen kunnen dan nog eens onderverdeeld worden in seizoensgebonden data en niet-seizoensgebonden modellen. Seizoen wijst hier echter niet enkel op regelmatige afwijkingen binnen de tijdspanne van 1 jaar maar op cyclische regelmatige afwijkingen doorheen de volledige tijdreeks. Dit betekent dus dat een cyclus ook wekelijks herhaald zou kunnen worden.

1.1 Probleemstelling

Voorspelde tijdreeksen kunnen zeer belangerijke data zijn om de richting waarin bepaalde beslissingen genomen moeten worden aan te geven. Een brandend actueel voorbeeld hiervan zou een voorspellingsmodel van de coronacijfers zijn wanneer er zo'n model beschikbaar is met een betrouwbaarheid van 100% wat zou er in principe geen discutie meer mogen zijn over de ersnt van de situatie en het treffen van de correcte maatregelen zou een pak makkelijker te regulariseren vallen. Dit is echter vrij onwaarschijnlijk aangezien er in dit geval tal van factoren een invloed hebben op die cijfers waarvan er velen zeer moeilijk te registeren vallen. Gelukkig zijn er ook problemen waarvan de factoren makkelijker registreerbaar zijn zoals het verminderen van de ijsoppervlakten aan de polen. Zo zal onderzocht worden welk model de beste voorspellingen maakt voor univariate niet-seizoensgebonden tijdreeksen, univariate seizoensgebonden tijdreeksen, multivariate niet-seizoensgebonden tijdreeksen en multivariate seizoensgebonden tijdreeksen. De conclusies en de broncode van dit onderzoek zal kan kunnen dienen als basis voor het opstellen van voorspellingsmodellen van andere tijdreeksen.

1.2 Onderzoeksvraag

Vergelijkende studie van voorspellingsmodellen voor tijdreeksen

1.3 Onderzoeksdoelstelling

De doelstelling van deze bachelorproef is om te achterhalen welke modeltypes de beste resultaten halen voor elk type invoerdata. Dit onderzoek zal ook kunnen dienen als basis voor het opstellen van een voorspellingsmodel van andere tijdreeksen.

1.4 Opzet van deze bachelorproef

De rest van deze bachelorproef is als volgt opgebouwd:

In Hoofdstuk 2 wordt een overzicht gegeven van de stand van zaken binnen het onderzoeksdomein, op basis van een literatuurstudie.

In Hoofdstuk **??** wordt de methodologie toegelicht en worden de gebruikte onderzoekstechnieken besproken om een antwoord te kunnen formuleren op de onderzoeksvragen.

In Hoofdstuk **??**, tenslotte, wordt de conclusie gegeven en een antwoord geformuleerd op de onderzoeksvragen. Daarbij wordt ook een aanzet gegeven voor toekomstig onderzoek binnen dit domein.

2. Stand van zaken

In dit hoofdstuk zullen de 3 gebruikte voorspellingstechnieken toegelicht worden.

2.1 Regressieanalyse

Regressieanalyse is een voorspellende modelleringstechniek die de relatie tussen een afhankelijke en onafhankelijke variabele onderzoekt. Deze techniek kan ook toegepast worden indien er meerdere onafhankelijke variabelen beschikbaar zijn. Dit betekent dus dat er een relatie gezocht wordt tussen de afhankelijke en de onafhankelijke variabelen om om de best passende lijn of regressievergelijking op te stellen. Op basis van deze regressievergelijking waarbij de onafhankelijke variabele als invoerwaarde gebruikt zal worden, zal de afhankelijke variabele voorspeld worden.
In het geval van tijdreeksen zal het tijdsverloop altijd de onafhankelijke variabele zijn bij univariate modellen en een van de invoervariabelen bij multivariate modellen. Er zijn verschillende manieren om aan regressie te doen de voornaamste technieken zijn: lineaire regressie, polynomiale regressie en logaritmische regressie. Deze zullen dan ook verder toegelicht worden.

2.1.1 Lineaire Regressie

De meest eenvoudige vorm van regressie is lineaire regressie. Dit type regressie wordt vaak gebruikt en wordt best toegepast op continue data (**Pant2019**). Dit is data die gelijkmatig stijgt en zal grafisch weergegeven worden als een rechte. Bij realistische voorbeelden

zullen deze waarden vrijwel nooit op een rechte lijn liggen maar zullen er altijd enkele uitschieters zijn. Indien we dit grafisch zouden weergeven krijgen we een rechte te zien die de gegeven waarden optimaal zal benaderen.

Een voorbeeld van zo'n lineaire data is weergegeven op de tabel 2.1. Op basis van de cijfers valt dit niet echt makkelijk waar te nemen maar op figuur 2.1 is er duidelijk stijgende tendens zichtbaar in de prijs naargelang het oppervlak van de leefruimte groter is.

Hieronder staat de basisformule voor een lineaire regressie afgebeeld.

$$Y_i = \beta_0 + \beta_1 X_i + \varepsilon_i \tag{2.1}$$

Het eerste deel van deze formule meerbepaald de eerste en de tweede term van de optelling zal de lineaire component weergeven terwijl de resterende derde term rekening zal houden met de willekeurige fout.
De lineaire component is in principe een eerstegraadsfunctie die de basis zal vormen voor de formule. De Y_i zal de afhankelijke variabele weergeven, dit zal dus de te voorspellen waarden weergeven. β_0 zal het intercept weergeven ofwel de waarde van de afhankelijke variabele waarvoor de onafhankelijke variabele theoretisch gezien nul zou zijn. Of eenvoudigweg de waarde van de functie waarbij de curve de y-as snijdt. β_1 wordt benoemd als de richtingscoëfficient, deze zal bepalen hoe sterk de rechte stijgt of daalt. Een hogere richtingscoëfficient zal tot een steilere curve leiden terwijl een richtingscoëfficient die 0 benaderd vlakker zal zijn, wanneer deze waarde effectief 0 is zal de afhankelijke waarde constant zijn de onafhankelijke variabele die weergegeven wordt door χ_i heeft hier dus geen invloed op aangezien deze geneutraliseerd wordt door de 0. Indien de richtingscoëfficient negatief is zal deze curve ook dalen in plaats van stijgen.
De Random Error component zal ervoor zorgen dat er ook rekening gehouden wordt met meetfouten die aanwezig geweest zouden kunnen zijn bij het verwerven van de data deze wordt weergegeven door ε_i.

Door de parameters β_0 en β_1 te variëren namelijk en hiervan telkens het foutpercentage te nemen zal bepaald kunnen worden welke waarden voor deze parameters voor de beste benadering van de reëele data zorgen. De foutmarge zal weergegeven worden doormiddel van de som van de gekwadrateerde verschillen tussen de waarden voorspelde en de effectieve waarden. Het berekenen van deze foutmarge zal berekend worden zoals weergegeven bij formule 2.2.

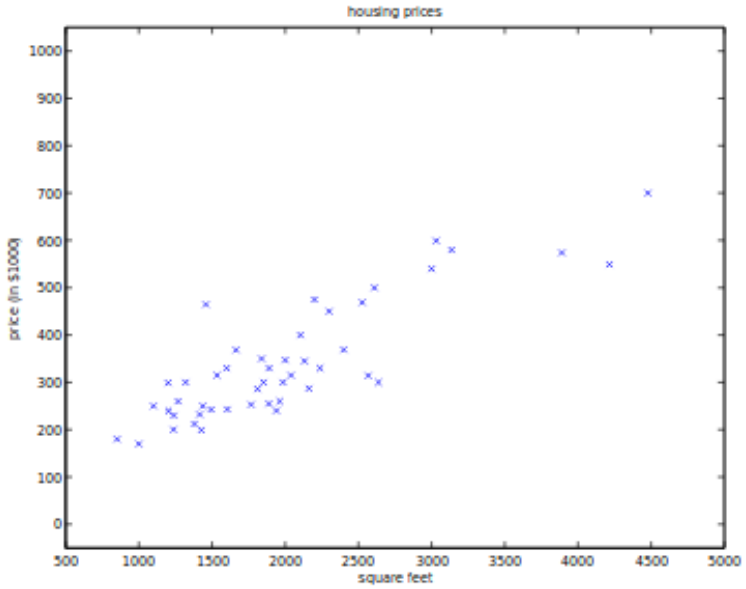
$$Error = \sum (actual\ output - predicted\ output)^2 \tag{2.2}$$

Nu rijst de vraag hoe we die fout kunnen verkleinen en de parameters zodanig kunnen doen evolueren zodat ze de kleinst mogelijke foutmarge benaderen. Dit wordt gerealiseerd aan de hand van gradient descent een uitgebreide uitleg van gradient descent komt aan bod bij het bespreken van de LSTM techniek maar het komt er op neer dat gradient descent er zal voor zorgen dat de optimale parameters zo nauwkeurig mogelijk bepaald zullen kunnen met een zo laag gebruik van middelen.

Tabel 2.1: Voorbeelddata voor lineaire regressie

Living area (feet^2)	Price (1000$)
2104	400
1600	330
2400	369
1416	232
3000	540

Figuur 2.1: Grafische weergaven voorbeelddata voor lineaire regressie



2.1.2 Polynomiale Regressie

Soms zal de foutmarge bij data hoog blijven bij lineaire regressie ook al is er een verband zichtbaar, dit verband zal dan waarschijnlijk niet te vatten vallen onder een gewone rechte. In zo'n geval kan polynomiale regressie hulp bieden, dit zal een complexere functie aan de data proberen fitten. Formule 2.3 is dan formule van een 2^{de} graadsfunctie

$$Y = \theta_0 + \theta_1 X_1 + \theta_2 X_2^2 \tag{2.3}$$

Dit zorgt ervoor dat de algemene formule herleid wordt naar de formule 2.4 indien we een polynomiale vergelijken wensen toe te passen bij de regressieanalyse.

$$Y = \theta_0 + \theta_1 X + \theta_2 X^2 + ... + \theta_m X^m + residual\ error \tag{2.4}$$

Ook bij polynomiale regressie wordt gebruik gemaakt van gradient descent om de minimale kost te berekenen aangezien dit in principe een uitbreiding is op lineaire regressie.

Dus polynomiale regressie zal de beste gemiddelde inschatting kunnen maken van de relatie tussen de afhankelijke en onafhankelijke variabelen. De curve die gefit moet worden zal zeer complexere vormen dan een simpele rechte kunnen aannemen en meer dan enkel lineaire verbanden kunnen deduceren uit de data. Het grootste nadeel van polynomiale regressie is dat enkele uitschieters de resultaten sterk kunnen beïnvloeden terwijl er bij lineaire regressie meer validatietechnieken beschikbaar zijn.

2.1.3 Logistieke Regressie

Logistieke regressie is gelijkaardig aan lineaire regressie maar in tegenstelling tot lineaire regressie zal dit voornamelijk gebruikt worden voor classificatieproblemen, waarbij dus geen continue waarden dienen voorspeld te worden maar nominale waarden. Onder zijn meest simpele vorm zal dit type algoritme nuttig zijn bij het voorspellen van binaire waarden bijvoorbeeld indien men al dan niet aan een aandoening zal lijden. Aangezien in deze paper enkel continue waarden voorspeld zullen worden, zal hier niet dieper op in gegaan worden. (Starmer, 2018)

2.1.4 Meervoudige lineaire regressie

Indien er meerdere onafhankelijke variabelen beschikbaar zijn die een lineaire invloed zouden hebben op de afhankelijke variabele kunnen deze ook in rekening gebracht worden. Dit model zal dan opgesteld worden aan de hand van meervoudige lineaire regressie ofwel multiple linear regression. Formule 2.5 geeft weer hoe de onafhankelijke waarden bij meervoudige lineaire regressie berekend zullen worden.

$$Y_i = \beta_0 + \beta_1 X_{i1} + \beta_2 X_{i2} + ... + \beta_n X_{in} + \varepsilon_i \tag{2.5}$$

Net zoals bij enkelvoudige lineaire regressie zal Y_i de afhankelijke variabele weergeven en β_0 het intercept. Het verschil zit hem in de termen die volgen want aangezien er rekening gehouden wordt met meerdere onafhankelijke variabelen, zullen ook deze verwerkt worden in de formule door middel van een waarde te voorzien voor elke onafhankelijke variabele weergegeven door χ_i met bijhorend intercept weergegeven door β. De n in de formule geeft dan het aantal onafhankelijke variabelen weer. De ε_i zal ook in dit geval een factor zijn die rekening zal houden met de meetfouten. (Kenton, 2020)

2.2 ARIMA

ARIMA is een acroniem gevormd door AutoRegressive Integrated Moving Average wat vrij vertaald kan worden als het autoregressie geïntegreerde voortschrijdende gemiddelde. Deze benaming geeft ook onmiddelijk de sleutelaspecten van dit model weer.

Zo staat autoregressief voor het modelleren van een volgende stap in een sequentie als een lineaire functie op basis van de voorgaande waarden in deze sequentie.
Daarnaast is ARIMA ook een geïntegreerd model wat inhoudt dat niet de cumulatieve waarden gebruikt worden om voorspellingen te maken maar er gekeken wordt naar het verschil tussen die cumulatieve waarden.
Ten slotte wordt ook gekeken naar het voortschrijden gemiddelde in plaats van naar de ruwe data zelf dit zal ervoor zorgen dat extreme waarden veroorzaakt door anomaliteiten geneutraliseerd worden. Hierdoor zal de globale trend beter waarneembaar zijn.

Elk van deze componenten is geparameteriseerd en kan dus aangepast worden deze parameters worden benoemd als (p ,d & q)

p: Het aantal verlate observaties die worden opgenomen in het model. Deze worden in het engels benoemd als lag observations of lag order. d: Zal bepalen hoeveel keer ruwe observaties van elkaar afgetrokken worden dit kan ook benoemd worden als de mate van differentiatie of the degree of differencing in het engels. q: Deze parameter zal het aantal dagen waarvan het voortschrijdend gemiddelde genomen wordt aangeven deze wordt in het engels benoemd als the order of moving average.

Een lineair regressiemodel wordt geconstrueerd met data waarbij er een graad van differentieatie is zodat deze stationair is. Dit houdt in dat trend en seizoensstructuren die het regressiemodel op een negatieve manier beïnvloeden verwijderd worden.

Bij elke parameter kan ook de waarde 0 opgegeven worden zodat dit aspect horende bij die parameter buiten beschouwing gelaten wordt en dit model effectief gebruikt kan worden als gelijk welke combinatie zoals bijvoorbeeld een autoregressief en een geïntegreerd model al dan niet gebruikmakend van het voortschrijdend gemiddelde. (Brownlee, 2018a)

SARIMA

In sommige gevallen blijkt er een seizoensgebondenheid aanwezig te zijn in de data. Deze seizoensgebondenheid zal de data dan op een cyclische manier beïnvloeden en dit is iets waar bij een gewoon ARIMA model geen rekening mee wordt gehouden. Daarom is er een variant op het ARIMA model genaamd SARIMA waarbij de toegevoegde S zal staan voor de seizoensgebondenheid. Dit zal in rekening gebracht worden door extra parameters toe te voegen. Zo zullen de parameters bij ARIMA (p,d,q) zijn, terwijl de parameters bij SARIMA (p, d, q)(P, D, Q)m zijn, waarbij m zal staan voor het aantal tijdsstappen binnen een seizoenscyclus. Zo zal bij een jaarlijkse cyclus m gelijk zijn aan 12 als de tijdspanne tussen de observaties 1 maand bedraagt. Daarnaast zal de P staan voor autoregressieve seizoensorde ofwel de seasonal autoregressive order, D voor de gedifferencïeerde seizoensorde ofwel de seasonal difference order en Q voor de seizoensorde van het voortschrijdend gemiddelde ofwel de seasonal moving average order. (Brownlee, 2018b)

VARMAX

Wanneer er meerdere tijdsreeksen voorspeld moeten worden waarbij er een verband mogelijk zou kunnen zijn tussen de verschillende tijdsreeksen kan de VARMAX techniek gebruikt worden. Hierbij zal de V staan voor vector wat duidt op het interpreteren en voorspellen van vectoren. Dit type model zal dan ook gebruikt worden bij het voorspellen van multivariate of meervoudige tijdreeksen AR wijst andermaal op autoregressief. MA slaat ook weer op het voortschrijdend gemiddelde. De X in VARMAX staat voor de invloed van exogene factoren op de factoren die voorspeld moeten worden.

2.3 Long Short Term Memory

Als tweede methode die gebruikt zal worden om tijdreeksen te voorspellen wordt gekozen voor een recurrent neuraal netwerk met gebruik van Long Short Term Memory modellen wat afgekort wordt als LSTM. Om de werking van een recurrent neuraal netwerk met LSTM te schetsen moet eerst de werking van een neuraal netwerk toegelicht worden.

2.3.1 Theoretische toelichting

Een neuraal netwerk zal inputwaarden omzetten in een score die bepaalt hoe waarschijnlijk het is dat deze gekoppeld kan worden aan een vooraf gedefinieerde outputwaarde. Dit valt makkelijkst te vergelijken met hoe een menselijk brein werkt. Zo zullen wij bijvoorbeeld het cijfer 1 herkennen aan zijn vormen. Door de verschillende pixels, gebruikt voor het weergeven van ons cijfer 1 als inputwaarden voor een neuraal netwerk te gebruiken en de outputwaarden gelijk te stellen met cijfers 0 tot en met 9 kan het netwerk aangeleerd worden deze cijfers te herkennen.

Dit gebeurt niet vanzelf. Zo bevinden zich tussen de input en de outputlaag verschillende lagen die benoemd worden als verborgen lagen. In deze lagen bevinden zich knooppunten (neuronen) waarin waarden worden berekend die een interpretatie geven van de inputwaarden waarmee ze verbonden zijn. Wanneer we dit toepassen op ons voorbeeld zou in een knooppunt bijvoorbeeld kunnen bekeken worden of er in het midden van de te interpreteren pixels een lijn staat. Wanneer dit het geval zou zijn zou er in dit neuron een grote waarde weergegeven worden en zal deze een sterk signaal sturen. Wanneer we voor dit voorbeeld met 1 verborgen laag zouden werken zou het neuron dat een sterk signaal zendt bij het herkennen van een centrale verticale lijn een belangrijke waarde zijn voor het herkennen van een 1 en een 4.De sterke van dit signaal zal dus een grote invloed hebben op de kans dat het resultaat een 1 of een 4 is. Wanneer we hier dieper op ingaan kunnen we aan elk neuron een formule toewijzen die er zo uitziet:

$$n = w_1 i_1 + w_2 i_2 \tag{2.6}$$

W staat voor de gewichten, i staat voor de inputwaarde. Stel dat bij dit neuron de bovenste

3 pixels genomen worden en we voor de eenvoudigheid werken met een 3x3 afbeelding, dan zal de formule er als volgt uitzien:

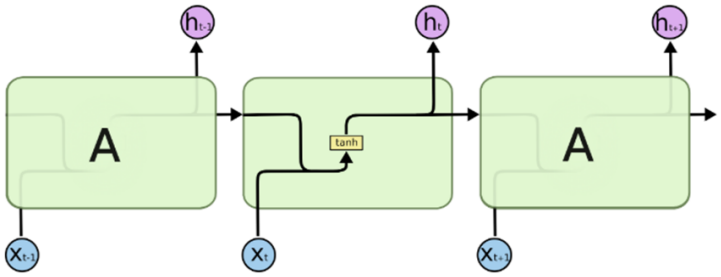
$$n = w_1 i_1 + w_2 i_2 + w_3 i_3 \tag{2.7}$$

Toepassing

Dit voorbeeld is gebaseerd op deze (Olah, 2015) toelichting van LSTM Bij dit voorbeeld zijn we nog steeds op zoek naar een 1 en kennen een hoge waarde toe aan donkere kleuren opdat het model de afbeelding zou kunnen interpreteren. Dit zou betekenen dat i_1 en i_3 een lage waarde moeten hebben en i_2 een hoge waarde. In dit geval zullen de gewichten voor w_1 en w_3 gelijkgesteld worden aan -1 en het gewicht voor w_2 aan 1. Daardoor zal n een hoge waarde krijgen als de centrale pixel zwart is en de buitenste pixels wit. Dit wordt de propagation function genoemd. Om deze waarden om te zetten naar een waarde die interpretabel is doorheen het volledige model wordt het resultaat van de propagation function nog eens gebruikt als invoerwaarde voor een activatiefunctie. In dit geval zou de hyperbolic tangent (tanh) een goede keuze zijn, maar dit hangt af van de toepassing. Bij dit voorbeeld zou dat signaal dan naar de outputlaag gestuurd worden die op zijn beurt ook weer diezelfde formule toepast maar dan niet met de invoerwaarden maar met de resultaten uit de verborgen laag. Hieruit zal dan een score zou komen die aangeeft hoe waarschijnlijk het is dat de waarden die de pixels weergeven uit de invoerlaag het cijfer vormen dat toegekend is aan dit neuron in de uitvoerlaag. Deze toepassing waarbij een cijfer wordt herkend is een voorbeeld van een toepassing van convelutionele neurale netwerken omdat de data slechts in 1 richting verloopt, een resultaat wordt niet beïnvloed door voorgaande inputwaarden. Bij recurrente neurale netwerken is dit wel het geval. Hierbij wordt rekening gehouden met de beoordelingen van voorgaande inputwaarden een voorbeeld hiervan is het beoordelen van zinnen de volgorde van de ingevoerde woorden zal een rol spelen. Zo zal ‘eet ik‘ geïnterpreteerd kunnen worden als een vraag aan de hand van de woordanalyse. Terwijl ’ik eet’ zal beoordeeld worden als een statement. Dit zal louter gebeuren op basis van de woordvolgorde waarbij connecties tussen verschillende cellen teruglopend kunnen zijn onder elkaar en de vorige inputwaarden dus een invloed zullen hebben op de beoordeling van de volgende inputwaarden (Lievens, 2018) pg.148. Dit is dus een gelijkaardige werking aan convelutionele neurale netwerken maar de signalen verlopen niet allemaal in dezelfde richting binnen het netwerk. Zo kan er een signaal van de 2de verborgen laag teruggestuurd worden naar de 1ste verborgen laag. Op die manier hebben de waarden van vorige iteraties een impact op de volgende resultaten en dit is wat we willen bereiken. Een lus van eenzelfde stuk uit het neuraal netwerk die informatie van vorige iteraties doorgeeft aan zichzelf dit wordt weergegeven op Figuur 2.2. Om het verloop van een tijdreeks te voorspellen moet ook rekening gehouden worden met het tijdsverloop. Zo zal het verloop van de vorige dagen een invloed hebben op het verloop van de volgende dagen. Dus voor deze toepassing zullen recurrente neurale netwerken gebruikt worden aangezien convelutionele neurale netwerken geen rekening houden met de voorgaande inputwaarden.

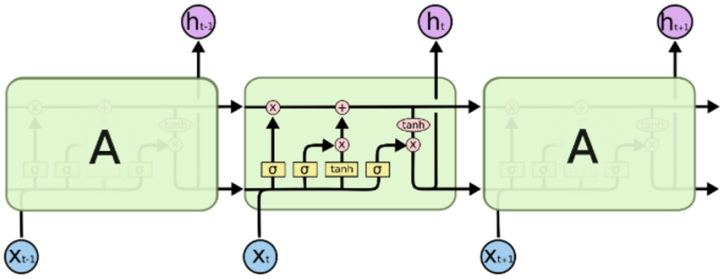
Het probleem bij gewone recurrente neurale netwerken is dat deze moeilijk patronen her-

Figuur 2.2: Cel in een neuraal netwerk dat gegevens van een andere cel gebruikt om een nieuw signaal uit te sturen



kennen op lange termijn. Hiervoor biedt de long short term memory variant van recurrente neurale netwerken een oplossing. Dit zorgt ervoor dat enkel de data die bijgehouden moet worden uit vorige iteraties een impact zal hebben op nieuwere resultaten. De werking van een LSTM wordt voorgesteld op figuur 2.3.

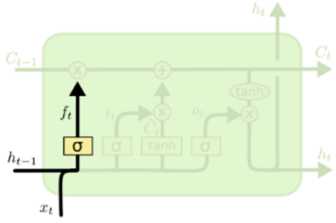
Figuur 2.3: Gedetaileerdere figuur van een cel uit een neuraal netwerk dat informatie interpreteert uit een andere cel van het neurale netwerk



Dit aan de hand van 4 stappen:

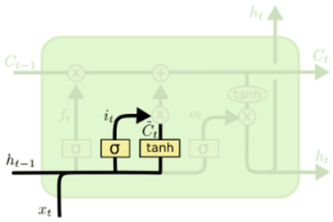
Stap 1 Eerst wordt bepaald welke info weggegooid mag worden in de âĂIJforget layerâĂİ. Deze zal een waarde tussen 0 en 1 uitvoeren waarbij een 0 zal betekenen dat deze waarde volledig genegeerd mag worden en waarbij een 1 zal betekenen dat alles behouden zal moeten worden.

Figuur 2.4: Grafische weergave van de forget layer binnen het neuron



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

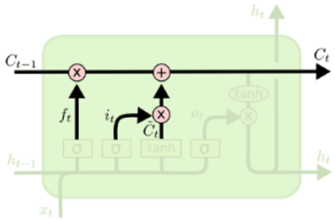
Figuur 2.5: Grafische weergave van de input gate layer binnen het neuron



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

[b]

Figuur 2.6: Grafische weergave van het 2^{de} deel van de input gateway



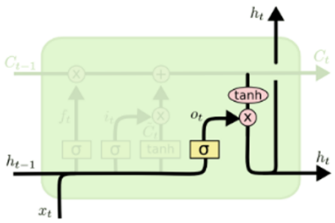
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Stap 2 Voor de volgende stap zal in de âĂIJinput gate layerâĂİ bepaald worden welke waarden geÃijpdatet moeten worden aan de hand van een sigmoïde funtie die hier zal resulteren in i_t. Daarna zal de vector geconstrueerd worden met de potentiële waarden C_t.

Stap 3 In deze stap zullen we de voorgaande celwaarde vernieuwen. Zo zullen door C_{t-1} te vermenigvuldigen met f_t de overbodige waarden vergeten worden. En door het product van i_t en C_t hierbij op te tellen zullen de achterhaalde waarden geÃijpdatet worden.

Stap 4 Als laatste zal de uitvoer bepaald worden. Dit is een gefilterde versie van de celtoestand. Die filtering gebeurt aan de hand van een sigmoïde functie. Het resultaat hiervan wordt dan vermenigvuldigd met de tanh van de celtoestand. Met deze formule verkrijgen we de uitvoerwaarde die de cyclus kan verderzetten.

Figuur 2.7: Grafische weergave van het onderdeel van de cel waarin de input vermenigvuldigd wordt met de tanh van de celtoestand om de outputwaarde te bekomen



$$o_t = \sigma\left(W_o \; [h_{t-1}, x_t] \; + \; b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

2.3.2 Praktische toelichting

Deze praktische toelichting is gebaseerd op het artikel van Jason Brownlee (Brownlee, 2018c) en zal dienen als basis voor het opstellen van een LSTM netwerk.

Univariate LSTM modellen

Univariate LSTM models maken voorspellingen voor data waarbij er slechts 1 afhankelijke variabele is. Hierbij bestaan de problemen uit een reeks observaties. Op basis van vorige waarden in een reeks zal de volgende waarde in diezelfde reeks voorspeld worden. In deze subsectie zullen de voorgestelde modellen toegepast worden op eenstaps univariate tijdreeksen maar deze kunnen makkelijk aangepast worden om gebruikt te worden bij andere soorten tijdreeksen.

Datavoorbereiding Een tijdreeks zelf bestaat uit een sequentie van waarden zoals hieronder bijvoorbeeld.

Listing 2.1: Originele datasequentie

```
[10, 20, 30, 40, 50, 60, 70, 80, 90]
```

Om deze waarden te gebruiken voor het trainen van het neuraal netwerk zal deze tijdreeks omgevormd moeten worden naar samples. De dimensies van deze samples zullen bepalen hoeveel inputwaarden en outputwaarden zullen gebruikt worden bij het model. Aangezien we bij dit voorbeeld met eenstapsvoorspellingen werken zal elk sample over 1 outputwaarde beschikken. Bij dit voorbeeld worden 3 inputwaarden genomen. Dit zal als gevolg hebben dat de samples van de gegeven datasequentie er als volgt zullen uitzien.

Listing 2.2: Samples van de gegeven datasequentie

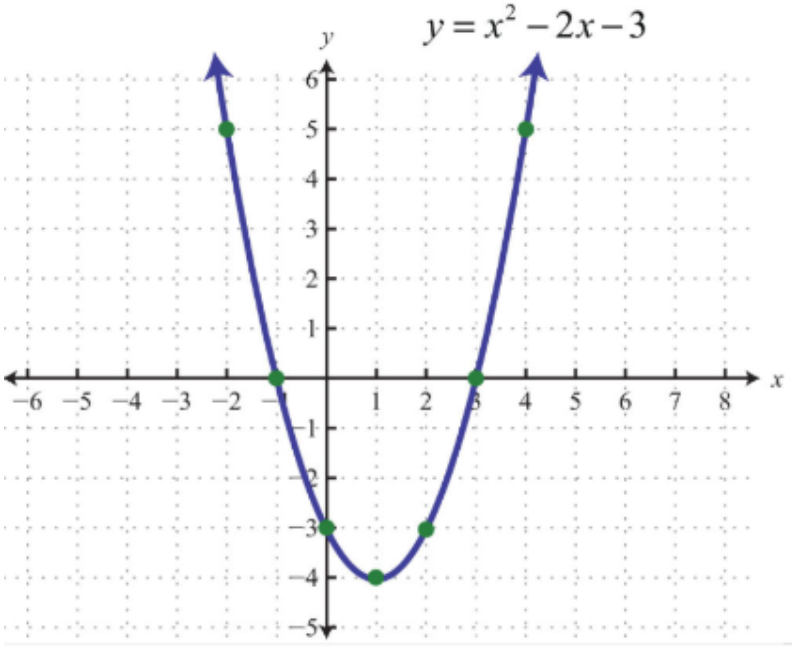
```
X,                  y
10, 20, 30          40
20, 30, 40          50
30, 40, 50          60
...
```

Vanilla LSTM

Een Vanilla LSTM is een LSTM model dat over een enkele verborgen laag beschikt en een uitvoerlaag die een voorspelde waarde zal aangeven. In deze verborgen laag zullen zich 50 nodes bevinden. De hoeveelheid te kiezen nodes is afhankelijk van de data. Zo zullen meer nodes de foutmarge verkleinen maar een kleiner aantal nodes zal meer gaan generalizeren, wat het eindresultaat ook zou kunnen verbeteren. Uiteindelijk moet hier een evenwicht in gevonden worden.

Om te bepalen of een signaal binnen een neuron effectief sterk genoeg zal zijn om te activeren en er zelf een te sturen zal de relu activatiefunctie gebruikt worden. Dit is de

Figuur 2.8: Parabolische functie



[b]

meest gebruikte activiatiefunctie in neurale netwerken (Liu, 2020) en heeft als voordeel dat het er weinig complexe wiskunde aan te pas komt bij de berekening waardoor het performant is. Daarnaast convergeert het ook sneller waardoor de helling niet zal stagneren en blijft gelijkmatig stijgen naargelang x groter wordt. Tenslotte is deze functie ook spaars wat inhoudt dat er in geen gevallen geen activatie is, specifiek bij de relu functie alle waarden onder 0 liggen. Spaarsheid komt de efficiëntie van het model ten goede omdat hierdoor bepaalde neuronen enkel geactiveerd zullen worden bij specifieke prikkels. Zo zou een netwerk voor image recognition een bepaald neuron kunnen hebben dat zal activeren indien er een kattenoor waar te nemen valt, maar het zou niet wenselijk zijn moest dit neuron actief zijn indien er een afbeelding van een gebouw wordt getoond.

De gewichten van de verliesfunctie zullen bepaald worden aan de hand van de Adam versie van stochastische gradient descent. Om stochastische gradient descent te definiëren zal eerst toegelicht moeten worden wat gradient descent precies inhoudt. Gradient staat voor de afgeleide van de verlies functie en descent betekent afdaling. De combinatie van deze 2 begrippen houdt in dat men zal afdalen tot het laagste punt bereikt wordt. (Srinivasan, 2019)

We kunnen dit principe makkelijkst voorstellen aan de hand van een parabool. Deze wordt grafisch weergegeven op figuur 2.7. Het minimum van deze parabolische functie wordt bereikt wanneer x een waarde van 1 heeft. Als startpunt wordt een willekeurige waarde op de curve genomen waarbij dan nagegaan wordt in welke richting bewogen moet worden om dichter bij het minimum te komen. Wanneer het minimum gezocht wordt zou een stapgrootte gedefinieerd kunnen worden en dan telkens met even grote stappen

afgedaald kunnen worden. Wanneer deze stapgrootte bepaald moet worden zou bij een kleine stapgrootte het minimum nauwkeuriger gedefinieerd zijn maar zou dit veel meer berekeningen vergen. Bij een grote stapgrootte zou het minimum snel berekend kunnen worden, maar het zou onnauwkeurig zijn.

Bij gradient descent wordt deze stapgrootte dynamisch bepaald op basis van het product van de hellingsgraad en de learning rate, een vooraf bepaalde waarde die moet voorkomen dat de stapgrootte te groot wordt. Wanneer de hellingsgraad laag zal zijn betekent dit dat het minimum wordt benaderd en dus kleinere stappen genomen zullen worden om dit zo nauwkeurig mogelijk te bepalen. De kans dat 0 effectief wordt bereikt is zeer klein, daarom stopt het algoritme wanneer een vooraf bepaalde minimale stapgrootte wordt bereikt. Daarnaast kan ook een maximum aantal stappen ingesteld worden waarbij het algoritme stopt.

Gradient descent kan duidelijk toegelicht worden aan de hand van deze 5 stappen:

1. Neem de afgeleide van de verliesfunctie voor elke parameter in deze functie
2. Kies willekeurige waarden voor deze parameters
3. Voer de parameterwaarden in in de afgeleiden
4. Bereken de stapgroottes: Stapgrootte = helling * learning rate
5. Bereken de nieuwe parameters: Nieuwe parameter = oude parameter - stapgrootte

Herhaal stappen 3 tot 5 tot het minimum bereikt wordt.

Bij dit voorbeeld zou gewoon de afgeleide van deze parabool kunnen genomen worden om zo tot het minimum te komen, maar bij complexere functies is dit niet mogelijk en biedt gradient descent hiervoor een oplossing.

Bij gebruik van stochastic gradient descent wordt niet telkens voor alle data gebruikt wanneer de parameters herberekend worden, maar een willekeurige subset uit de dataset om het aantal berekeningen te reduceren wat deze methode performanter maakt zonder al te veel nauwkeurigheid te verliezen. (Starmer, 2019)

Dan rest enkel Adam nog verklaard te worden. Dit is een variant op de klassieke vorm van gradient descent waarbij de learning rate aangepast voor elk gewicht binnen het netwerk en doorheen het trainingsproces.

Deze aanpassingen worden door de auteurs van Adam omschreven als het combineren van 2 andere uitbreidingen op gradient descent namelijk:

Adaptive Gradient Algorithm (AdaGrad) die er voor zorgt dat een per-parameter learning rate die performantie zal verbeteren van de gradiënt.

Root Mean Square Propagation (RMSProp) deze uitbreiding op gradient descent zorgt voor aanpassingen in de learning rates op basis van de schaal van de laatste gradiënten kortom hoe snel deze veranderen. (Brownlee, 2017b)

En zal geoptimaliseerd worden op basis van de 'mse' verliesfunctie.

Listing 2.3: Samples van de gegeven datasequentie

```
# univariate lstm voorbeeld
from numpy import array
from keras.models import Sequential
from keras.layers import LSTM
from keras.layers import Dense

# Opslitsen van univariate sequentie in samples
def split_sequence(sequence, n_steps):
    X, y = list(), list()
    for i in range(len(sequence)):
        # vinden van het einde van dit patroon
        end_ix = i + n_steps
        # nagaan of we ons na de sequentie bevinden
        if end_ix > len(sequence)-1:
            break
        # verwerven van invoer en uitvoerdelen van het patroon
        seq_x, seq_y = sequence[i:end_ix], sequence[end_ix]
        X.append(seq_x)
        y.append(seq_y)
    return array(X), array(y)

# Bepalen van de sequentie
raw_seq = [10, 20, 30, 40, 50, 60, 70, 80, 90]
# bepalen van het aantal tijdsstappen
n_steps = 3
# onderverdelen in samples
X, y = split_sequence(raw_seq, n_steps)
# reshape from [samples, timesteps] into [samples, timesteps, features]
n_features = 1
X = X.reshape((X.shape[0], X.shape[1], n_features))

# definieer model
model = Sequential()
model.add(LSTM(50, activation='relu', input_shape=(n_steps, n_features)))
model.add(Dense(1))
model.compile(optimizer='adam', loss='mse')

# fit model
model.fit(X, y, epochs=200, verbose=0)

# demonstreer voorspelling
x_input = array([70, 80, 90])
x_input = x_input.reshape((1, n_steps, n_features))
```

```
yhat = model.predict(x_input, verbose=0)
print(yhat)
```

Stacked LSTM

Een stacked LSTM is een variant op het Vanilla LSTM waarbij er meerdere verborgen lagen na elkaar geplaatst worden. Het toevoegen van een extra laag zal voor een langere uitvoeringstijd zorgen. Het voordeel hiervan is dat hierdoor meerdere onderdelen makkelijker geregistreerd kunnen worden en in verband met elkaar gebracht kunnen worden. Zo zou de eerste laag bijvoorbeeld ronde vormen kunnen detecteren die dan door de volgende laag gedetecteerd kunnen worden als een groter geheel bijvoorbeeld een fiets. Dan zou dit uiteindelijk geclassificeerd kunnen worden als een wielerwedstrijd wanneer het nog grotere geheel bekeken wordt.

Bidirectionele LSTM

Bidirectionele LSTMs kunnen getrained worden gebruik makend van alle beschikbare invoerinformatie uit het verleden en de toekomst binnen een specifieke tijdspanne. Dit door middel van de toestandsneuronen op te splitsen van een normaal Recurrent Neuraal Netwerk in een deel dat verantwoordelijk is voor de positieve tijdsdirectie en een deel voor de negatieve tijdsdirectie. (Brownlee, 2017a)

Bidirectionele LSTMs worden voornamelijk gebruikt bij language processing omdat de definitie van een woord afgeleid kan worden uit de context. De begrippen die bepalend zijn voor de definitie kunnen zich zowel voor als achter het woord zelf bevinden dus moet de zin vanuit 2 richtingen geanalyseerd kunnen worden om de definitie te achterhalen.

CNN LSTM

Een convelutioneel neuraal netwerk ook wel een CNN genoemd wordt voornamelijk gebruikt bij het analyseren van tweedimensionale grafische data. Een CNN is ook sterk in het extraheren en aanleren van features van eendimensionele data. Een CNN model kan ook gecombineerd worden met een LSTM model om een CNN-LSTM te vormen. Hierbij wordt het CNN gebruikt om subreeksen te interpreteren die op zich dan weer zullen geïnterpreteerd worden als een reeks door het LSTM model.

ConvLSTM

ConvLSTM is ook een soort van CNN-LSTM waarbij het convelutionele deel rechtstreeks is ingebouwd binnen het LSTM gedeelte. Dus ook dit CNN-LSTM werd ontwikkeld voor het interpreteren van tweedimensionale grafische data maar kan aangepast worden voor gebruik bij univariabele tijdreeksvoorspellingen.

Multivariabele LSTM modellen

Bij multivariabele tijdreeksen zullen er in tegenstelling tot univariabele tijdsreeksen wel meerdere observaties zijn per tijdseenheid. Deze worden dan nog eens onderverdeeld in multiple input series en multiple parallel series.

Meerdere invoerreeksen

Bij de variant met meerdere invoerreeksen zullen er meerdere invoerwaarden zijn voor elk tijdstip. Hierbij zal slechts voor 1 van de invoerreeksen een voorspelling gemaakt worden.

Meerdere parallelle reeksen

Bij de LSTM variant waarbij er meerdere parallelle reeksen zijn zullen er meerdere invoerwaarden zijn voor elk tijdstip. Hier zal er voor elke reeks een waarde voorspeld worden.

Multi-Step LSTM modellen

Tot nu toe werden enkel maar one-step LSTM modellen toegelicht waarbij de invoerreeks slechts zal resulteren in 1 uitvoerwaarde. Bij multi-step LSTM modellen zal er meer dan 1 uitvoerwaarde voorspeld worden.

Vector output model

Bij dit type multi-step LSTM model wordt de output van het model retourneerd onder de vorm van een vector die de resultaten van de volgende tijdstappen zal weergeven en niet enkel van de volgende tijdstap.

Encoder-Decoder Model

Dit model is ontwikkeld om sequenties met variabele uitvoer te voorspellen (Brownlee, 2017a). Het is ook ontworpen om problemen waarbij er zowel invoer als uitvoersequenties zijn te behandelen. Deze problemen worden ook wel benoemd als sequence-to-sequence of seq2seq-problemen. Deze techniek wordt voornamelijk gebruikt bij vertalingstoepassingen maar kan ook toegepast worden op tijdreeksen. Het model bestaat uit 2 hoofdonderdelen de encoder en de decoder.

De encoder is verantwoordelijk voor het lezen en interpreteren van de invoerreeks. De encoder zal dan een vector met een vaste lengte uitvoeren die een interpretatie zal geven van de invoerreeks. Normaal zal de encoder een Vanilla LSTM model zijn, maar er kan even goed een stacked, bidirectioneel of CNN model gebruikt worden.

De decoder zal de uitvoer van de encoder dan gebruiken als invoer. Eerst zal de uitvoervector met vaste lengte van de encoder even veel keer met zichzelfzelf samengevoegd worden

als er tijdsstappen verwacht worden in de uitvoer. Zo zal een reeks die bestaat uit 1, 2 resulteren in 1, 2, 1, 2 indien er verwacht wordt dat er een uitvoer zal zijn van 2 waarden. Deze sequentie zal dan doorgegeven worden aan de decoder zelf. Die zal dan voor elke tijdsstap een uitvoerwaarde zal geven. Deze uitvoer zal dan nog eens geïnterpreteerd worden door een verborgen laag alvorens dit volledig model de multi-step uitvoerreeks zal teruggeven.

Multivariate Multi-Step LSTM Models LSTM modellen

Multiple input Multi-Step Output

Wanneer men over een dataset beschikt met meerdere invoerparameters maar slechts 1 enkele parameter voor meerdere tijdsstappen dient te voorspellen spreekt men over multiple input multi-step output.

Multiple Parallel Input and Multi-Step Output

Wanneer de dataset waarvan een voorspelling gemaakt moet worden meerdere invoerparameters heeft maar er ook meerdere voorspeld moeten worden zal dit benoemd worden als een multiple parallel input and multi-step output LSTM.

A. Onderzoeksvoorstel

A.1 Introductie

Artificiele intelligentie wordt steeds meer toegepast dus de optimale methodes bepalen om voorspellingen te maken is van vitaal belang. Verschillende datasets kunnen er volledig anders uitzien en deze kunnen dan ook op verschillende manieren ingedeeld worden. Voor dit onderzoek zal gefocust worden op data die tijdsgebonden is. Door het gebruik van dit type data zullen de modellen rekening moeten houden met de tijdsafhankelijkheid tussen de verschillende waarden.

A.2 Stand van zaken

Er zijn heel wat methoden die kunnen toegepast worden om een voorspelling te maken van tijdsgebonden data. Voor deze paper zullen enkel polynomiale vergelijkingen, ARIMA en LSTM getest worden.
De meest primitieve manier om een trend te voorspellen is het fitten van een polynomiale vergelijking op de trainingsdata en deze nadien toe te passen op de testdata. Daarnaast kan ook de ARIMA-methode (Brownlee, 2018a) gebruikt worden ofwel het Autoregressive Integrated Moving Average. Deze methode combineert autoregressie en voortschrijdend gemiddelde. Autoregressie modeleert de volgende stap in een sequentie als een lineaire functie van de waarden uit voorgaande tijdspannes. De methode van het voortschrijdend gemiddelde modeleert de volgende stap in de sequentie als een lineaire functie van de resterende fouten van een gemiddeld proces bij voorgaande tijdspannes. Er moet ook opgemerkt worden dat er een verschil is tussen een model met een voortschrijdend gemiddelde

en het voortschrijdend gemiddelde van de dataset zelf.
Ook neurale netwerken kunnen toegepast worden bij het maken van voorspellingen van tijdreeksen. LSTM (Long Short Term Memory) is een vaak gebruikt modeltype om tijdreeksen te voorspellen. Dit model zal het verloop van de volgende waarden voorspellen op basis van de ingevoerde waarden rekening houdend met de chronologie waarin ze voorkomen. Hierbij zal de invloed van oudere waarden minder relevant worden naargelang er meer waarden ingevoerd worden.
Op zowel de ARIMA als de LSTM modellen bestaan er varianten om multivariate tijdreeksen te voorspellen, bij ARIMA worden deze benoemd als VARMAX modellen. Ook bij polynomiale regressie kunnen multivariate times series voorspeld worden. Ook voor tijdreeksdata waar een duidelijk seizoenseffect zichtbaar is bestaat er een variant op het ARIMA model genaamd SARIMA.

A.3 Methodologie

Om na te gaan welke methodes de beste resultaten behalen zullen zowel polynomiale regressie, ARIMA en LSTM toegepast worden op 2 datasets, 1 waarbij een duidelijk seizoenseffect zichtbaar is en 1 waar geen duidelijke seizoensgebonden invloed aanwezig is. Daarnaast zullen ook al deze methodes of gespecialiseerdere varianten van deze methodes toegepast worden op datasets met en zonder seizoenseffect waarbij meerdere invoerparameters gebruikt zullen worden.
Om deze methodes te scoren zullen de laatste waarden weggelaten en voorspeld worden waardoor uit de foutmarge tussen de voorspellingen en de werkelijke waarden afgeleid zal kunnen worden welke methode de meest accurate voorspelling zal kunnen maken. Om deze methodes te quoteren zullen de r^2 en de RMPSE (Root Mean Square Percentage Error) scoringsmethodes benut worden.

A.4 Verwachte resultaten

Er valt te verwachten dat polynomiale regressie het zwakste resultaat zal behalen aangezien polynomiale technieken, door de aard van een veelterm, doorgaans minder goed zijn voor extrapolatie waarvoor ze in deze context benut zullen worden. Ik verwacht dat LSTM best zal scoren aangezien dit type model specifiek voor tijdreeksen is opgesteld gevolgd door ARIMA.

A.5 Verwachte conclusies

Er valt te verwachten dat de voorgestelde technieken goede resultaten zullen behalen. Vooral voor LSTM liggen mijn verwachtingen vrij hoog omdat ik reeds een artikel (**Siami-Namini2018**) heb gelezen waarbij de voorspellingen voor LSTM accurater zijn. Ik

heb een pak minder vertrouwen in polynomiale regressie aangezien deze techniek minder goed presteert bij extrapolatie en maar beter tot zijn recht komt bij interpolatie. ARIMA zal waarschijnlijk ook goede resultaten behalen.

Bibliografie

Brownlee, J. (2017a). Encoder-Decoder Long Short-Term Memory Networks. Verkregen 24 november 2020, van https://machinelearningmastery.com/encoder-decoder-long-short-term-memory-networks/

Brownlee, J. (2017b). Gentle Introduction to the Adam Optimization Algorithm for Deep Learning. Verkregen 24 november 2020, van https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/

Brownlee, J. (2018a). 11 Classical Time Series Forecasting Methods in Python (Cheat Sheet). Verkregen 24 november 2020, van https://machinelearningmastery.com/time-series-forecasting-methods-in-python-cheat-sheet/

Brownlee, J. (2018b). A Gentle Introduction to SARIMA for Time Series Forecasting in Python. Verkregen 24 november 2020, van https://machinelearningmastery.com/sarima-for-time-series-forecasting-in-python/

Brownlee, J. (2018c). How to Develop LSTM Models for Time Series Forecasting. Verkregen 20 mei 2020, van https://machinelearningmastery.com/how-to-develop-lstm-models-for-time-series-forecasting/

Kenton, W. (2020). Multiple Linear Regression (MLR). Verkregen 24 november 2020, van https://www.investopedia.com/terms/m/mlr.asp

Lievens, S. (2018). Artificiële Intelligentie, lesnota's HOGENT.

Liu, D. (2020). A Practical Guide to ReLU. Verkregen 24 november 2020, van https://medium.com/@danqing/a-practical-guide-to-relu-b83ca804f1f7

Olah, C. (2015). Understanding LSTM Networks. Verkregen 26 mei 2020, van https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Srinivasan, A. V. (2019). Stochastic Gradient Descent — Clearly Explained !! Verkregen 24 november 2020, van https://towardsdatascience.com/stochastic-gradient-descent-clearly-explained-53d239905d31

Starmer, J. (2018). StatQuest: Logistic Regression. Verkregen van https://www.youtube.com/watch?v=yIYKR4sgzI8&ab_channel=StatQuestwithJoshStarmer

Starmer, J. (2019). Gradient Descent, Step-by-Step. Verkregen van https://www.youtube.com/watch?v=sDv4f4s2SB8&ab_channel=StatQuestwithJoshStarmer