Beter voorspellen met Long Short-Term Memory

In het moderne verkeersmanagement neemt *voorspellen* een steeds belangrijkere plek in. Logisch, want wie in staat is om vooruit te kijken, kan anticiperen – en dus effectiever informeren, bijsturen en ingrijpen. Maar hoe kom je in real-time tot goede voorspellingen van bijvoorbeeld reistijden of de ontwikkeling van een file? In deze bijdragen duiken we in de intelligente modellen van het type *Long Short-Term Memory*, LSTM. Hoe werken ze? En wat kunnen we ermee?

Illustratie: Andrei Dzemidzenka

Aan verkeersdata geen gebrek. Van vrijwel alle rijkswegen, provinciale wegen en belangrijke gemeentelijke wegen krijgen we per wegvak en per minuut gegevens binnen als snelheid en intensiteit. Die data geven een scherp beeld van de toestand op de weg *nu*, maar zijn ook input voor voorspellende modellen.

Dat is belangrijk om bijvoorbeeld goede reistijdschattingen te maken: 'u komt zo laat op uw bestemming aan'. Of om tot de tekst op informatiepanelen te komen: 'Tot knooppunt Everdingen via A27: 7 min +1, via A2: 9 min +12'.

Wat voor methodieken en modellen zijn geschikt om dit soort voorspellingen te doen? Voor lineaire en stationaire reeksen volstaat *exponentiële afvlakking* met bijvoorbeeld het ARIMA-model.[1] Wanneer de dynamiek mechanistisch en (bijna) lineair is, zijn *toestandsruimtemodellen met Kalman-filtering* een mooie optie. Deze methoden zijn transparant en volstaan uitstekend voor min of meer gewone verkeersomstandigheden.

Het probleem is echter dat het op veel plaatsen zo druk is op de weg, dat van 'gewone omstandigheden' steeds minder sprake is. Vooral in moderne stedelijke netwerken speelt dit probleem. De laatste jaren grijpen modelspecialisten daarom naar zwaarder geschut: ze zijn gaan experimenteren met *machine-learningmethoden* die meer variabelen kunnen verwerken, zoals kalender- en weerskenmerken en gebeurtenisindicatoren. Dat is inderdaad een verbetering, maar met standaard *machine learning* blijven langetermijn- en contextafhankelijke effecten toch te vaak buiten beeld.

Recurrente neurale netwerken kunnen temporele afhankelijkheden leren. De 'gewone' variant van dit type modellen heeft nog moeite om informatie over langere perioden vast te houden, doordat de gradiënten geleidelijk vervagen. Maar de variant *Long Short-Term Memory* of LSTM lost dit probleem op met extra intelligentie die bepaalt wat belangrijk is om te onthouden en wat vergeten kan worden.

Dat maakt een LSTM uiterst geschikt voor het voorspellen van de verkeerstoestand – ook in minder gewone omstandigheden.

1 ARIMA staat voor *Autoregressive Integrated Moving Average*. Het is een veelgebruikte statistische techniek voor het analyseren én voorspellen van tijdreeksgegevens.

Technische toelichting

Om te begrijpen hoe deze aanpak werkt, staan we kort stil bij een praktisch voorbeeld: hoe een LSTM de snelheid op een corridor enkele minuten vooruit voorspelt.

Het model kunnen we ons het beste voorstellen als een reeks opeenvolgende 'geheugencellen'. Omdat we meetdata over de snelheid per minuut binnenkrijgen, zou een cel voor een minuut in de tijd staan. Figuur 1 laat zien welke processen zich in een enkele geheugencel afspelen en hoe de 'lijnen' zijn met de voorafgaande en de navolgende cel.

Elke minuut bestaat de invoer uit de actuele snelheid op het wegvak, enkele snelheden verder stroomopwaarts, de fase van de verkeerslichten, weersinformatie en een indicator voor bijzondere gebeurtenissen. Binnen de cel houdt het model op tijdstip *t* twee sporen bij. De *hidden state* (*h*) is een korte samenvatting die de cel meekrijgt van de voorafgaande cel (h_{t-1}) en die geüpdatet wordt doorgegeven aan de volgende cel (h_t). De *cell state* (*c*) is een contextueel geheugen dat belangrijke gebeurtenissen over de tijd heen meeneemt. Ook hier is er sprake van informatie uit de voorgaande cel (c_{t-1}) en van bewerkte informatie die naar de volgende cel gaat (c_t).

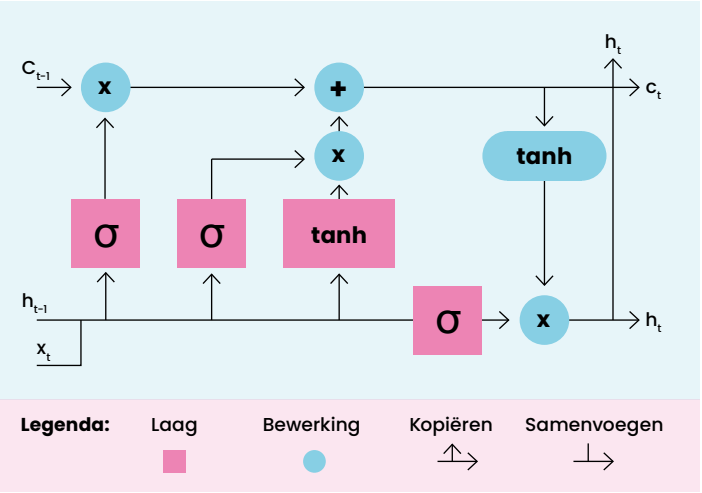
Drie beslispoorten

De bewerkingen die op *h* en *c* worden losgelaten, kunnen we het beste zien als drie 'beslissingen' die de cel neemt. In dit verband spreken we wel van *gates*, beslispoorten.

De forget-poort: welke oude context kunnen we vergeten en welke onthouden we? Via het bovenste spoor komt vanuit de voorgaande cel het contextuele geheugen c_{t-1} binnen. Een zogenaamde *sigmoid* (σ)-laag[2] fungeert als *forget*-poort en produceert een waarde f_t tussen nul en één. Deze waarde bepaalt hoeveel van het oude geheugen behouden blijft.
Wanneer bijvoorbeeld detectoren gedurende enkele minuten lage snelheden (= file) rapporteren, ligt f_t meestal rond 0,8-1,0. De cel zorgt er dan voor dat deze file niet uit het geheugen gewist wordt, maar behouden blijft. Zodra de snelheid weer terugkeert naar het gebruikelijke niveau, daalt f_t naar 0,1–0,3 en wordt de file 'vergeten'.

De input-poort: geven meetdata reden om het contextuele geheugen te updaten? Hier werkt een tweede sigmoid (σ)-laag, de *input*-poort, nauw samen met een eerste *tanh*-blok (roze in de figuur).[3] Dit blok stelt nieuw geheugen voor op basis van de meest recente meetdata.
Stel bijvoorbeeld dat de nieuwe meetdata laten zien dat het verkeer stroomopwaarts vertraagt. Het tanh-blok produceert dan een negatieve waarde, bijvoorbeeld -0,8: 'het systeem detecteert een daling'. De *input*-poort kent hieraan een waarde toe (i_t) van 0,9: 'deze verandering is belangrijk'. Het resultaat is dat de vertraging stroomopwaarts in het contextuele geheugen c_t wordt opgeslagen.

De output-poort: welke samenvatting geven we door? In dit proces werkt een derde sigmoid (σ)-laag, de *output*-poort, samen met een tweede tanh-blok (blauw in de figuur). Het doel van dit proces is om het geheugen *h*, de samenvatting, te updaten.
Tijdens een verstoring neigt de *output*-poort (o_t) ertoe meer van de opgeslagen context door te geven, zodat de voorspelling goed rekening houdt met de verstoring. Zodra de omstandigheden weer normaal worden, vermindert o_t deze doorgifte en keert de uitvoer terug naar het gebruikelijke gedrag.



Figuur 1:
Een LSTM-geheugencel. De cirkels geven een vermenigvuldiging of optelling aan. De blokken zijn getrainde lagen met een σ- of tanh-activatie.

De drie *poorten* werken dus als 'zachte' schakelaars die bij elke stap bepalen wat moet worden bewaard, wat moet worden overschreven en wat moet worden doorgegeven. Omdat de *cell state* informatie over de tijd heen doorgeeft met gecontroleerde updates, kan belangrijke verkeerscontext gedurende vele minuten behouden blijven. Deze opzet pakt het 'vervaagprobleem' aan dat de gewone recurrente modellen beperkt.

Architectuur

In de praktijk volstaat meestal een relatief eenvoudige architectuur. Voor korte voorspellingshorizonnen zijn één of twee LSTM-lagen voldoende. Voor langere horizonnen werken een *encoder* en *decoder* samen: de *encoder* vat de recente geschiedenis samen, terwijl de *decoder* stap voor stap de toekomst voorspelt op basis van die gecodeerde geschiedenis en zijn eigen eerdere voorspellingen.

Toepassingen

Nu de vraag wat we als verkeerskundigen met LSTM's kunnen. Om daar een beeld van te geven, bespreken we enkele werkende toepassingen uit de praktijk.[4]

Voorspelling van aankomsttijden. De geschatte aankomsttijd is cruciaal voor verkeersmanagement, reizigersinformatie en vlootbeheer. Een oplossing die in de praktijk al tot flink nauwkeurigere voorspellingen leidt, is die van een graafneuraal netwerk (GNN)[5] in combinatie met LSTM. Het GNN modelleert dan de weginfrastructuur en de actuele toestand ervan, terwijl de LSTM de temporele dynamiek vastlegt.

2 Een *sigmoid* (σ)-laag kunnen we zien als een 'zachte' schakelaar. Deze laag zet een bepaald invoergetal om in een waarde tussen 0 en 1. Bij bijvoorbeeld een *forget*-poort zou een nul 'compleet vergeten' betekenen, één 'alles onthouden'. De waarde kan echter ook 0,1 of 0,2 zijn of 0,9. Hiermee is een zachte beslissing mogelijk, zoals deels of geleidelijk vergeten.
3 *Tanh* staat voor hyperbolische tangens en is een functie die getallen omzet naar waarden tussen -1 en +1. Deze functie geeft dus een *richting* mee aan de waarde, zoals een dalende snelheid (tanh is negatief) of juist een stijgende (tanh is positief).
4 Zie de online versie van dit artikel op nm-magazine.nl voor de literatuurverwijzingen bij de toepassingen.
5 Een *graaf* is een netwerk van knooppunten (zoals sensoren, haltes of kruispunten) die met elkaar verbonden zijn via lijnen (wegen of routes). Modellen die *graafneurale netwerken* gebruiken, leren hoe verschijnselen als snelheidsveranderingen of reizigersstromen zich door dit netwerk verspreiden.

Kortetermijnverkeer en corridorsnelheden. Op een enkele corridor of een beperkt aantal wegvakken kan een LSTM de opbouw en afname van spitsverkeer vastleggen met minimale *feature engineering*. Naarmate de ruimtelijke omvang groeit tot tientallen of honderden sensoren, gaan netwerkeffecten overheersen en zijn hybride modellen met graaf- en recurrente componenten beter.

Reizigersaantallen, halteertijden en perronmanagement. Reizigersstromen en aankomsten hebben hun dagelijkse en wekelijkse ritmes. Maar tijdens feestdagen en grote evenementen wijken de waarden juist behoorlijk af. Een compacte LSTM leert hoelang recente omstandigheden van invloed blijven, en zet intervallen, instapvolumes en eenvoudige gebeurtenisindicatoren om in kortetermijnvoorspellingen voor halteertijd en perrondrukte.
Wanneer ruimtelijke effecten belangrijk zijn, kan de LSTM worden gecombineerd met een netwerkgraaf.

Verkeerslichten en fasevoorspelling. Adaptieve regelingen profiteren van kortetermijnvoorspellingen van cycluslengte en fasenverdeling. Een LSTM die gedetailleerde detectoraantallen en fasestatussen als invoer gebruikt, kan de tijd tot groen en de duur van komende cycli meerdere cycli vooruit voorspellen. Dat ondersteunt prioriteitslogica en snelheidsadvies met minder handmatige regels. Een kleine convolutionele *front-end* kan patronen binnen een cyclus samenvatten, terwijl de LSTM de toestand over meerdere cycli heen vasthoudt. Op echte kruispunten bleek deze opzet nauwkeurige cyclus- en fasevoorspellingen te leveren op basis van detectorgegevens.

Aandachtspunten

Ook bij LSTM's geldt uiteraard dat de resultaten afhankelijk zijn van een juiste opzet van het systeem. We noemen een aantal aandachtspunten.

Databereik en dekking. De frequentie van de gegevens moet aansluiten bij de beslissingen die genomen worden. Voor snelheden op wegvakken, vertrektijden en halteertijden volstaan meestal enkele maanden aan historische data met een resolutie van *één tot vijf minuten* om dagelijkse patronen en gangbare verstoringen vast te leggen. Voor bijvoorbeeld planningshorizonnen zijn jaren aan *dagelijkse of wekelijkse gegevens* nuttig.
Tijdstempels en tijdzones moeten consistent worden gehanteerd over alle databronnen heen – veel 'modelleerfouten' blijken uiteindelijk timingfouten te zijn. Ook ontbrekende waarden vragen om speciale aandacht. Kleine hiaten kunnen voorwaarts of achterwaarts worden ingevuld én gemarkeerd, zodat het model ze niet aanziet voor echte metingen. Langere hiaten worden beter opgevangen met modelgebaseerde imputatie die informatie 'leent' van verwante signalen. Als een sensor structureel onbetrouwbaar blijkt, is het beter die (tijdelijk) buiten gebruik te stellen.

Kenmerken en modelomvang. De basisopzet is doorgaans eenvoudig: een recent tijdvenster van de doelvariabele, kalenderkenmerken en een beperkte set weers- en gebeurtenisindicatoren. Wanneer een wegvak beïnvloed wordt door segmenten stroomopwaarts, worden die signalen expliciet meegenomen.
Als netwerkeffecten de dynamiek domineren, is een 'graafbewust model' (*graph-aware model*) beter: een standaard LSTM is er niet op ingericht om de topologie te leren. In risicogevoelige toepassingen is het bovendien beter om probabilistische uitkomsten te rapporteren – een centrale schatting met een onzekerheidsband in plaats van één enkel getal.

Drift en hertraining. Verkeers- en vervoerspatronen veranderen voortdurend door werkzaamheden, omleidingen, dienstregelingswijzigingen en seizoenseffecten. Continue prestatiemonitoring over een schuivend tijdvenster maakt veranderingen vroegtijdig zichtbaar, zodat hertraining kan plaatsvinden zodra bijvoorbeeld de betrouwbaarheid van de voorspelde intervallen afneemt.
Lichte drift kan vaak worden opgevangen door *fine-tuning* op basis van het laatste modelcheckpoint, in combinatie met geactualiseerde *scalers*, vakantiekalenders en gebeurtenisindicatoren. Structurelere breuken vragen om een volledige hertraining op bijgewerkte historische data. Het is wel belangrijk om van deze updates en dataveranderingen een *changelog* bij te houden, zodat prestatieverschillen traceerbaar blijven.

Onzekerheid die door de keten reist. Wanneer een voorspelling een vervolgactie aanstuurt, moet de onzekerheid in die voorspelling worden meegegeven. Zo kan bij het gebruik van snelheidsvoorspellingen voor aankomsttijdschattingen de hele verdeling worden doorgegeven, zodat bijvoorbeeld een logistieke partij een aankomsttijd met betrouwbaarheidsinterval ziet in plaats van een snelheid met foutmarge.

Conclusie

Klassieke voorspelmethoden zijn betrouwbaar zolang patronen stabiel zijn. LSTM's zijn juist bedoeld voor situaties waarin interacties sterk zijn en de invloed van de recente geschiedenis afhangt van de context.

LSTM's zijn niet bedoeld om de topologie van een netwerk te doorgronden. Is die topologie relevant, dan is het vaak het beste de LSTM te koppelen aan graaflagen.

Een verstandige aanpak is om te beginnen met de eenvoudigste methode die een beslissingsvraag beantwoordt. Indien nodig kan geleerd temporeel geheugen worden toegevoegd. Wanneer netwerkeffecten overheersen, kan ook nog ruimtelijke structuur worden ingebouwd. Echte implementaties zijn tot slot te gebruiken als leidraad voor de volgende verbetering. ●

De auteurs
Mingze Gong is onderzoeker van het DAIMoND Lab van TU Delft. Hij wordt begeleid door dr. Yanan Xin, dr. ir. Winnie Daamen en prof. dr. ir. Serge Hoogendoorn.
Willem-Jan Gieszen is onderzoeker van het TTS Lab van TU Delft. Zijn begeleiders zijn prof. dr. ir. Serge Hoogendoorn en dr. ir. Haneen Farah.